On Calibrating Enterprise Switch Measurements

Boris Nechaev
Helsinki Institute for
Information Technology HIIT /
Helsinki University of
Technology TKK
boris.nechaev@hiit.fi

Vern Paxson
International Computer
Science Institute
University of California,
Berkeley
vern@icir.org

Mark Allman
International Computer
Science Institute
mallman@icir.org

Andrei Gurtov
Helsinki Institute for
Information Technology HIIT /
Helsinki University of
Technology TKK
gurtov@hiit.fi

ABSTRACT

The complexity of modern enterprise networks is ever-increasing, and our understanding of these important networks is not keeping pace. Our insight into intra-subnet traffic (staying within a single LAN) is particularly limited, due to the widespread use of Ethernet switches that preclude ready LAN-wide monitoring. We have recently undertaken an approach to obtaining extensive intra-subnet visibility based on tapping sets of Ethernet switch ports simultaneously. However, doing so leads to a number of measurement calibration issues that require careful consideration to address. First, one must correctly account for redundant copies of packets that appear due to switch flooding, which if not accurately identified can greatly skew subsequent analysis results. We show that a simple, natural rule one might use for doing so in fact introduces systematic errors, but an altered version of the rule performs significantly better. We then employ this revised rule to aid with calibration issues concerning the fidelity of packet timestamps and the amount of measurement loss that our collection apparatus incurred. Additionally, we develop techniques to "map" the monitored network in terms of identifying key topological components, such as subnet boundaries, which hosts were directly monitored, and the presence of "hidden" switches and hubs. Finally, we present initial analyses demonstrating that the magnitude and diversity of traffic at the subnet level is in fact striking, highlighting the importance of obtaining and correctly calibrating switch-level enterprise traces.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Measurement

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.
Copyright 2009 ACM 978-1-60558-770-7/09/11 ...$10.00.

Keywords

Trace calibration, enterprise networks, network traces, switch-based packet capture

1. INTRODUCTION

The network traffic of enterprises can be measured, and thus characterized, from a number of vantage-point perspectives. In the past, a great deal of work has used measurements captured at an enterprise's access link, which allows characterization of network activity involving the external Internet, but does not shed any light on activity that stays confined within the enterprise. More recently, studies have drawn upon measurements made at an enterprise's core routers [5]. Doing so yields insight into the enterprise's broader network dynamics, i.e., how hosts in one subnet communicate with those in another, but does not give any insight into intra-subnet communication. Alternatively, studies have measured communication on the end-hosts themselves [2]. While this approach yields information about all of a host's traffic—including communication that occurs outside the enterprise in the case of monitoring on a laptop—the measurements (i) lack a broader context of what is happening in the surrounding network (e.g., network load) and (ii) can be difficult to setup and manage.

From the mid-1980s through the mid-1990s, researchers readily measured intra-subnet traffic by leveraging the near-ubiquitous deployment of 10base2 Ethernet and simple hubs. Due to the bus-based nature of these networks a single measurement tap could capture the activity for all of the hosts on the subnet (e.g., [3]). However, with the rise of 10baseT switched Ethernet, this capability was lost—there is no longer a single vantage point that can see all of the traffic transiting a single Ethernet broadcast domain.

An intermediary approach between the recent techniques of measuring at an enterprise's inter-subnet routers or its individual end systems is to capture traffic as seen at different Ethernet *switches*. Monitoring hardware exists for facilitating such measurement. For example, the system we used for recording traffic aggregates 5 full-duplex 100 Mbps Ethernet streams onto a Gbps Ethernet link. (In fact, it does this for two pairs of such streams, representing 10 switch ports in total. See § 2 for details.) Such a monitor provides the capability of simultaneously analyzing multiple ports connected to a given switch. If these ports link directly to end-systems, then the monitor can capture the same traffic as observable directly on the end system, but with significantly less effort to do so in aggregate since we can measure multiple end systems concurrently. On the other hand, by not running directly on the end system we lose the opportunity to relate a node's network activity with its system/process activity [2]. If the ports instead connect to other switches, then we can obtain a view of more aggregated intra-subnet traffic.

While such switch monitoring provides a fairly economical means for measuring intra-subnet enterprise traffic, analyzing data obtained in this fashion raises a number of subtle issues regarding measurement fidelity. In this paper we assess calibration issues that arise when doing so, which we explore in the context of a fairly extensive set of switch traces we gathered from different Ethernet subnets inside the Lawrence Berkeley National Laboratory (LBL). Our goal is to establish a foundation for understanding the quality of, and artifacts present in, these traces, as a first step towards then being able to build up a sound understanding of how intra-subnet traffic behaves.

A central premise of our work is that measurements such as those we conducted will see significant further employment by others in the future, raising for others the same issues that we explore in this paper. We also note that while some of the calibration techniques we present may in retrospect appear straight-forward, our team—which includes members with extensive experience in measurement and calibration—found the techniques required significant investigation to develop. Thus, we believe there is considerable contribution in framing the calibration *approaches* to aid in future studies based on subnet switch measurements.

Four basic properties of the measurements we wish to calibrate concern *timing*, *loss*, *gain*, and *layout*. The first two are already familiar from previous studies of calibrating packet trace measurements, such as [6]: timing refers to the accuracy of the timestamps associated with the recorded packets, and loss refers to *measurement loss*, i.e., packets erroneously missing from a trace because the monitor failed to capture or record them.

By "gain," we mean instances of the monitor recording packets that *did not exist*—or, at least, did not exist as a distinct network event. These can in principle occur, for example, due to bugs in the monitor software ([6] discusses a kernel packet filter that in some circumstances recorded two copies of each packet). However, in the context of Ethernet switch measurement we must deal with the much more common phenomenon of a switch *replicating* a packet when forwarding it, and thus if we measure multiple switch ports concurrently, each port may include an instance of the packet, leading to multiple copies in the aggregated trace.

We refer to the additional copies of a packet recorded multiple times as *phantoms*. In one sense, they do not reflect a distinct network event, because the source originally transmitted only one instance, not several. In another sense, however, they do reflect network events, as their appearance is expected and reflects the switch's correct functioning.

In general, Ethernet switches can replicate packets for one of three valid reasons. First, any packet destined for the Ethernet broadcast address is forwarded to all ports that represent edges of the Ethernet broadcast spanning tree. For simple topologies (which includes the LBL enterprise),[1] this will nominally mean "all" ports of the switch other than the one from which the broadcast packet arrives; for more complex topologies, the switch might replicate to only a subset of the ports. Here, we put quotes around "all" because we find that the switches we measured sense whether a port currently has an active system at the other end of the link, and do not flood packets to the port if it does not.

Second, a switch might replicate packets sent to Ethernet multicast addresses, depending on its knowledge of the location of listeners for the given address (e.g., IP-level multicast can be correspondingly mapped to Ethernet-level multicast, and some switches sniff IGMP traffic to prune forwarding for ports without listeners.)

Third, if a switch receives a unicast Ethernet packet, it might *flood* it to all switch ports if it does not find an entry for the destination MAC address in its forwarding table. In a simple Ethernet topology we would expect this last phenomenon to occur only rarely (roughly, no more than once per flow, and perhaps significantly less), since any two-way communication should induce the switch to quickly enter an entry into its forwarding table. It is possible, however, that unicast flooding might occur more often due to asymmetric forwarding within the Ethernet subnet, or because the number of active flows exceeds the size of the forwarding table, causing the repeated eviction of flow entries.

Thus, in Ethernet switch traces we expect a significant proportion of the recorded packets to in fact reflect a form of "phantom." These replicas represent both a curse and a blessing. The curse is that for many forms of basic analysis, such as overall traffic mix, we need to accurately identify their presence lest they unduly skew our view of the prevalence of particular types of traffic. The blessing, however, is that—as we develop in this paper—they provide a means by which to calibrate the switch measurements.

The final trace property we calibrate concerns "layout," by which we mean identifying key topological components of the measurements: (i) which traffic remains in the subnet versus involves communication with external hosts; (ii) accurately determining the IP subnet associated with the Ethernet broadcast domain; (iii) finding which end systems in our traces we *directly* monitored (i.e., we captured all of the packets the system generated because its immediate network link was one of those we tapped); and the difficult problem of (iv) detecting instances of "hidden" switches, meaning cases where one of the ports monitored in the trace does not in fact lead directly to an end system but instead to a switch (or hub) that services multiple end systems.

We proceed as follows. In § 2 we discuss the switch traces used for the study. We then turn in § 3 to robust identification of phantoms and a corresponding removal process. In § 4 we leverage the fact that we collected two simultaneous traces to assess the agreement in the timestamps across each pair of traces. Once we can soundly spot phantoms and pair traces, we then in § 5 formulate and analyze different procedures for assessing measurement loss. In § 6 we develop approaches for calibrating elements of network "layout," and in § 7 then analyze the fully calibrated traces to assess the significance of intra-subnet monitoring (i.e., the degree to which traffic from a switch-based vantage point provides insight beyond that available to monitoring of only inter-subnet traffic). We conclude with a summary in § 8.

2. ANALYZED TRACES

Working in conjunction with LBL's networking staff, we captured the enterprise traces used for this study between October 2005 and March 2006. The intent behind the general approach was to record full packet payloads from a set of 10 switch ports for roughly a day, after which the monitoring would move to another set of 10 ports, either off of the same switch or (if exhausted) a new switch. In particular, the setup entailed two sets of Finisar Shadow 10/100 taps, each capturing both directions of 5 FastEthernet (copper) links, as illustrated in Figure 1.

[1] Note, LBL operators informed us that the switches are not meant to be running the Spanning Tree Protocol, although we found evidence that in some cases they do.

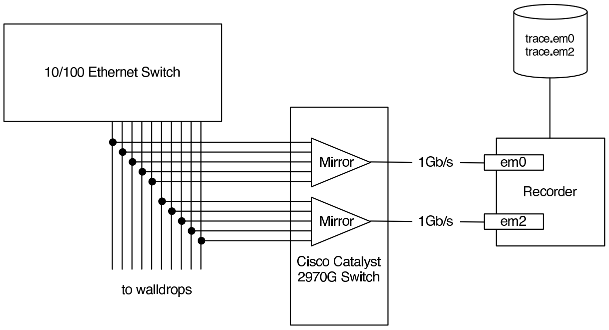


Figure 1: Measurement apparatus (courtesy Tom Kho).

All 10 taps were plugged into a second *aggregation* switch, consuming a total of 20 ports, since each tap fed two ports due to the full-duplex nature of the monitored links. We then aggregated 10 of those ports on the aggregation switch into a Gbps SPAN port, and the other 10 into a second Gbps SPAN port. These two Gbps ports then connected to two NICs on a workstation running `tcpdump` which recorded the aggregated traffic (including payloads) to disk. One minor variation to this approach occurred for the data captured in October 2005, where, due to cabling problems, the monitoring employed only 8 taps (2 sets of 4) rather than 10.

As mentioned above, the plan of operation was for a networking staff member to every day rotate the tapping arrangement to a different set of 8–10 ports, with this sometimes entailing moving the entire monitor apparatus to a new switch at a new location. (The locations included several different buildings at the enterprise.) The intent was for the monitored ports to always be directly connected to end systems; however, the network operators cautioned us that this could not necessarily be achieved because they do not always know when a port plugged into the monitored switch comes from a privately managed switch or hub rather than an end system. We revisit this question in § 6.

We collected 51 pairs of traces, i.e., 102 total traces, each half of a pair reflecting the Gbps SPAN feed from the aggregation switch that covered both directions of 4–5 FastEthernet ports on the monitored switch. Thus, each trace pair captured 8–10 FastEthernet ports off of a single switch. These traces in aggregate comprise 2,228 hours of traffic (individual traces usually running about 23 hours), totaling 869M packets[2] and about 400 GB of payload.

Note that certainly a juicy use of these traces would be to characterize the traffic in traditional ways for this little explored network type (e.g., traffic mix, peak-versus-average load, etc.). However, *we cannot soundly do so until we calibrate the traces* using the techniques we develop in this present work. As an exemplar of why the calibration effort is required, we present a high-level characterization in § 7 that illustrates how a switch-level view of the network can illuminate dramatic new insights about enterprise networks.

3. IDENTIFYING PHANTOMS

The predominance of phantoms in switch traces becomes clear upon casual inspection: we immediately see many identical packets very closely separated in time. We cannot however simply strip out packets that exactly repeat a previously seen packet, because we do not want to presume that sources never transmit multiple times separate instances of identical packets. For instance, we see multiple identical ARP packets throughout the traces, and even retransmitted on fairly short timescales (e.g., 1 second). Strictly speaking, it is impossible from our traces to know whether a given set of replicated packets reflect phantoms or true, separate end-host transmissions. However, given our knowledge of the nature of the network's operation—in particular, switches should replicate broadcast packets, should not necessarily replicate unicast packets, and when replicating should do so quickly—allow us to proceed with identifying phantoms with high confidence, as follows.

First, we examine the distribution of the time intervals between instances of identical packets appearing in a given trace, where identical means yielding the same MD5 hash over the entire packet. Figure 2 shows three examples of this distribution. We find that the particulars of the distribution vary significantly across our traces, but the overall form always exhibits a strong *mode* of intervals ≤ 10 μsec (lefthand side of the figure), another broad mode for values of roughly 1 sec or higher, and sometimes (as in the first two subfigures, but not in the third one) a third mode in the range of 100 μsec to 1 msec.

An initial, erroneous rule. Upon inspecting such distributions, it is easy at this point to then presume that the first mode reflects switch replication, since the very small time intervals correspond with back-to-back linespeed packets, which is what we would expect as the result of a switch's immediate replication of packets out multiple ports. One can then remove phantoms by eliding any packets whose contents match those of another packet seen no more than say 15 μsec in the past. One then interprets the other modes as representing truly distinct (separately originated) packets.[3]

Using a 15 μsec rule, however, in fact turns out to be a mistake. We initially used this definition, and only when further analyzing the implications of this approach did we discover the problem is more complex. When we applied the 15 μsec rule for identifying phantoms, we found that some traces exhibited frequent patterns of a set of identical packets being split into two parts. For example, if the replication size was 4 total copies (3 phantoms), then we would find regions in a trace where each packet was split into a group of 3 identical packets (1 end-host transmission and 2 phantoms) followed closely by 1 identical packet that is presumed to be a second end-host transmission. However, inspecting these incidents then revealed that in fact together the shortfalls arose from a single flight of 4 copies that had more time between them than 15 μsec.

This indicates that the natural 15 μsec rule is in fact too aggressive. As discussed in § 5 we want to build on the identification of phantoms for calibrating estimates of measurement loss by comparing the number of phantoms we expect to see with the number we actually observe. It is, therefore, important that we cull all phantoms from the traces before we assess measurement loss, or the phantoms will suggest more loss than actually occurred. For instance, if we expect to see four replicas for each broadcast packet and can correctly gather the phantoms together we may find no loss, whereas erroneously forming two groups with two packets each will suggest a measurement loss rate of 50%. On the other hand, the phenomenon of senders retransmitting identical packets is an interesting one in terms of understanding network dynamics

[2] This is the number of packets written to our trace files. As developed in subsequent sections, some of these packets are phantoms that need to be removed before drawing conclusions about the data.

[3] It is illuminating to note that on a 100 Mbps Ethernet the closest possible separation of minimum-sized packets is about 5 μsec, and for full-sized packets a bit under 125 μsec. However, we have confirmed that the timestamp differences for such full-sized packets are generally well under 10 μsec (below even the minimal full-sized packet spacing for the Gigabit Ethernet aggregation link). This discrepancy indicates that the timing reflects the monitoring apparatus's kernel timestamping packets that it retrieves (at a rate much higher than 100 Mbps) in batches out of a buffer, rather than the fine-grained spacing on the wire.

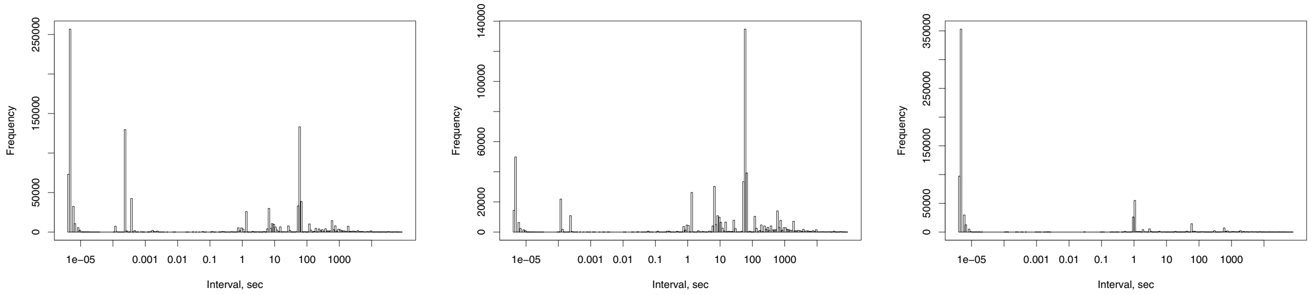


Figure 2: Distribution of interval between identical packets seen in three sample traces.

(in particular, load due to redundant traffic), so we do not want to use an overly conservative rule for identifying phantoms unless we know it incurs little in the way of "false positives" (misidentifying truly distinct packets as phantoms).

A more accurate rule. Given that identifying phantoms on the basis of a separation ≤ 15 μsec misidentifies some phantoms as reflecting separate source transmissions, we now turn to assessing whether a higher threshold might yield better results, or will too often conflate separate transmissions as phantoms. To do so, we rely on the notion of *sole-sourced* packets—those which we can infer with high confidence were transmitted only a single time within a given trace. We proceed as follows.

First, note that we confine our analysis to Ethernet broadcast packets, since those *should* be replicated to create *expected phantoms*, whereas for unicast packets or even for the general group of Ethernet multicast[4] packets that may or may not hold.

Next, we observe that any packet whose contents appears in a given trace 6 or more times was *not* sole-sourced, since the replication process can create at most 4 phantoms for a given end-host transmission.

If we knew that all 5 ports in a given trace were active, then we could assume that any packet whose contents appear ≤ 5 times was sole-sourced. However, applying this approach we find nominally sole-sourced packets separated by large amounts of time between the phantom copies (in some cases hours). Surely no switch replication process introduces such delays. In fact, it is plausible that no switch process introduces delays exceeding 100 msec. This is confirmed in Figure 2, which shows identical packets separated by either significantly less than 100 msec or significantly more than 100 msec. The same holds for the distribution computed for the other traces. Thus, we deem as sole-sourced broadcast packets for which (i) we see 5 or fewer total copies, and (ii) the intervals between the copies all lie below 100 msec. This rule is not ironclad—it's possible that with a small enough replication size and rapid transmissions by a source host that we will misclassify some non-sole-sourced packets—but it should suffice to find *all* of the truly sole-sourced packets, and we presume these will dominate.

Given this definition, we then examined the intervals seen between the copies of sole-sourced packets. Across all of the traces, we found a total of 20.4M such intervals. (Note, this number is much lower than the total number of packets in the traces because our analysis is limited to broadcast packets.) In 60% of the traces, the interval never exceeded 1 msec, and across all traces the 99th percentile never exceeded 2 msec (other than for a pathological trace with only one nominally sole-sourced packet in it). 99.998% of the intervals lie below 5 msec. All intervals lie below 58 msec.

Thus, a threshold of a few msec will work for correctly identifying the phantoms associated with virtually all of the sole-sourced broadcast packets. We can further estimate the corresponding *false positive* rate associated with a given threshold by determining how often we observe the same payload appearing > 5 times within a given threshold across all broadcast traffic (i.e., not just the sole-sourced packets). We know that structurally such occurrences *must* reflect multiply-sourced packets (or peculiar measurement problems). We find that a threshold of 60 msec yields 450 such false positives out of 7.8M unique packet payloads; with 5 msec it drops to 150; and with 100 μsec it further drops to 46.

In conclusion, we find that a value of 5 msec rather than 15 μsec gives us significantly more complete coverage of sole-sourced packets, and with at most a quite modest degree of misidentification of multiply-sourced packets. Accordingly, we define phantoms as identical copies of previous packets that we observed less than 5 msec in the past.

Analysis of the middle mode. In light of this new definition, we revisit Figure 2. Now we no longer interpret the middle mode from 100 μsec to 1 msec as representing truly distinct (separately originated) packets, and it behooves us to investigate why we observe a *separate* mode here rather than a single mode extending from a few μsec up to 1 msec.

The evidence indicates that switches *exhibit two different replication mechanisms*. Indeed, we find that this is the case. The middle mode is heavily dominated by a particular type of traffic, the Cisco Group Management Protocol (CGMP) [1]. These packets represent control traffic governing how switches forward IP multicast. We find that CGMP phantoms come with sharp intervals of time between them, exhibiting narrow spikes at 125 μsec and multiples thereof. This suggests that the switch uses a timer-driven mechanism to generate the replicas, and occasionally misses one or two beats of the timer when doing so.[5] We can rule out that these packets are instead generated by their source as multiple copies, since we consistently observe the switches replicating the packets to the same degree that they replicate broadcast packets (i.e., in reflection of the number of monitored ports that are currently active). If the replication occurred at the source, then sometimes these val-

[4] An Ethernet MAC address is defined as multicast (a *group* address) if the least significast bit of the first octet is 1; otherwise, the packet is unicast [4]. The Ethernet broadcast address is a particular multicast address with all bits set (ff:ff:ff:ff:ff:ff).

[5] We note that these packets are quite small, so it is presumably coincidence that the 125 μsec spacing happens to match that of maximum-sized Ethernet packets. We also note that intervals seen at multiples of 125 μsec, such as 250 μsec or 375 μsec, occur much more frequently than can be due to measurement loss leading to a failure to record intervening packets that all came 125 μsec apart.

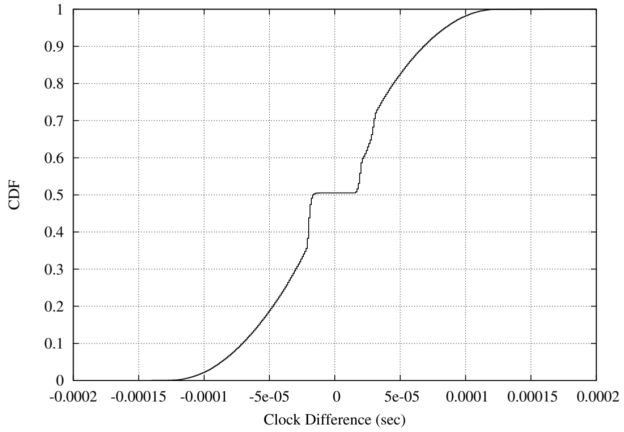


Figure 3: Distribution of the difference between a packet's timestamp on the first monitoring interface and the second monitoring interface across a sample merged trace.

ues would differ, since the source would have no way of knowing how many of the ports we were passively monitoring happened to currently be active.

4. TIMING

For the consideration of *timing fidelity*, we note that previous work has established that packet trace timestamps can exhibit a wide range of errors—many of which require pairs of traces recorded using different clocks to detect [7]. As sketched in § 2, our traces were all recorded using a single clock, and thus we cannot employ many clock-calibration methods. However, we do two quick calibration tests to ensure that we can in fact use the component trace of each pair to examine temporal aspects of the traffic with confidence that the events recorded in the component traces do in fact happen contemporaneously. After verifying that time progressed in monotone-increasing timestamps in all of our traces, we turned to the problem of comparing the timestamps recorded in each trace in a pair for the same event. While our setup used a single tracing host, it executed two separate tracing processes (`tcpdump`), leaving the calibration question of how closely the timestamps recorded by one match those recorded by the other. Our methodology proceeds as follows.

We first remove phantoms from each trace in our dataset using the $\Delta t < 5$ msec rule developed in § 3. Next, we merge each trace pair into a single trace—other than one pair of traces we removed, as discussed below—and collect identical broadcast packets together based on the MD5 checksum (with an indication of the origin trace for each packet). We then identify sole-sourced packets—which appear exactly once in each of two traces after removing phantoms—and examine the difference in their timestamps. In principle there are additional opportunities to compare the timestamping processes using non-sole-sourced packets. However, considering only sole-sourced packets side-steps the tricky issue of teasing apart pairs of packets that correspond to the same packet transmission, and we note that using only sole-sourced packets provides ample timestamp samples to solidly compare the two traces. In particular we find that the number of samples per trace ranges from 6,000–165,000 across the merged traces (i.e., per ≈23 hours of time).

For these samples, we examine the absolute value of the difference in the pairs of timestamps from sole-sourced packets. We find the median of this value to be 39 μsec, with 99.8% coming less than 152 μsec apart (maximum just over 5 msec). The largest median across any of the pairs of traces was 49 μsec. Finally, we note that the direction of the differences is roughly balanced so neither timestamping process runs consistently ahead of the other. Figure 3 shows the distribution of differences between packets observed on the first and second interfaces of our monitoring host for one of the trace pairs. The plot shows that (i) the differences are small, (ii) the differences go in both directions with each process ahead and behind almost exactly half of time, and (iii) there is a no-man's-land between ± 15 μsec that likely corresponds to the monitor's minimum time for switching from servicing (and thus timestamping) one interface over to the other. Note that this no-man's-land also points up an inherent shortcoming of the initial $\Delta t < 15\mu$sec rule for detecting replicas: a single interrupt switching the timestamping process from one interface to the other, arriving in the middle of a replication burst, can lead to an interval for the burst that exceeds 30 μsec (the shortest time until the timestamping can return to the original interface).

Examining the largest differences between the timestamping processes led to the discovery that one set of "paired" traces in fact did not constitute a pair, meaning that while the traces were recorded concurrently, they tapped different switches. (We confirmed this finding using the "layout" techniques discussed in § 6.) We omit these traces from any analysis we conducted using merged pairs of traces.

Based on sole-sourced packets we conclude that the timestamps of concurrently recorded traces are very closely aligned (as expected, since the same workstation recorded both of them), and thus we can use a fairly narrow window of time to analyze the merged pair for further details (either in terms of calibration—as we leverage the merged traces in § 5—or more generally in terms of characterizing the network).

5. DETECTING MEASUREMENT LOSS

We now turn to gauging the degree to which our monitoring setup incurred measurement losses, i.e., failed to record packets that traversed the tapped links. Such losses can arise from a variety of processes: a faulty tap that sometimes fails to physically copy bits transmitted over the link; buffer exhaustion in the aggregation switch as it multiplexes 10 streams of 100 Mbps each onto a 1 Gbps SPAN port; buffer exhaustion in the monitoring workstation, either in its NICs, its kernel, or the user-level `tcpdump` process; NIC PCI bus contention, since one machine was capturing two gigabit ethernet streams; or bit errors during transmission between these stages (presumed very rare). We need as best as we can to distinguish these effects, all of which only affect copies of the actual traffic, from true loss of actual traffic, since the latter represents an interesting networking event, while the former is a mundane measurement artifact.

We pursue four separate strategies for estimating measurement loss, two based on broadcast traffic (with one being generally conservative, the other less so), and two using unicast traffic. First, we can examine pairs of traces for the presence of "orphans"—broadcast packets that should appear in both traces in a pair but only show up in one. Second, as suggested previously we can assess variations of the *replication size* occurring in a trace, interpreting short-lived reductions in the number of phantoms observed that quickly return to the previous baseline level as likely reflecting measurement loss rather than tapped hosts going inactive and then becoming active again. This second approach is equivalent to the first where the reduction in replication size goes all the way to zero—but that event is not discernible when inspecting a single trace. Third, we exploit the structure and reliable nature of TCP traffic to infer measurement loss, by looking for instances where a receiver acknowledges data, but for which we do not see the data itself previously appearing in the trace. This technique lends itself to two variants, one based on the rate at which we observe such acknowledgments and the other based on the volume of data we can tell is missing.

We consider these four approaches in the following subsections. (Note that we defer a comparison between them until Figure 4, which presents per-trace estimates based on all four.) However, before we tackle these assessment strategies we need to address a peculiar measurement artifact. In the process of analyzing broadcast traffic present in one trace but not in its companion, we discovered that for many of the traces at either the beginning ("head") or the end ("tail") a fall-off in replication size occurs, coupled with a preponderance of orphans (packets missing from the companion trace). Sometimes, the replication size systematically falls off (5, 4, 3, 2, 1) and then likewise rises again. Based on changes in predominant MAC and IP addresses at these points, we identified the cause as reflecting the network operator physically moving the network monitoring taps from one set of ports to another—which in retrospect we would indeed expect to occur right at the beginning or end of a trace collection period, as the operator prepares to set up a new set of monitoring points.

Thus, we discovered that the traces are in fact polluted in the sense that they do *not* in their raw form reflect a single set of 5 monitored ports, but instead might each represent up to 10 such ports. We term this the *head/tail effect*. We manually determined the extent of this distortion and found that across all of the traces, the phenomenon did not manifest beyond the first or last 12 minutes of a trace.[6] Consequently, we *trimmed* each trace to delete its first and last 15 minutes (to have a conservative margin), and in addition *aligned* the trimming so that the two traces we would then merge started and ended at the same time. The analysis we discuss in this paper for pairs of traces uses these trimmed traces rather than the original raw traces.

5.1 Detecting Orphans

Given two traces comprising a trace pair from the same switch, we expect each broadcast event to manifest in both traces. Our first, generally quite conservative method for identifying measurement loss is therefore to verify this assumption and assess the loss rate when it does not hold. We used the merged traces described in § 4 to look for orphans. We expect that each broadcast packet transmitted by an end-host should appear in the merged trace exactly twice (i.e., one copy from each of the component traces).

We first identify all packets in the merged trace with the same MD5 hash value. We then flag instances where the number of such packets captured on the first interface differs from the number captured on the second interface. We record the imbalance as the number of orphans observed in the trace with fewer instances of the packet. This scheme is simplistic, and can fail in some cases. For example, when a roughly equal number of losses for the same MD5 hash value occurs in each of the component traces, the detection algorithm will underestimate the degree of measurement loss. However, it is very likely that the orphans that the process identifies do reflect measurement issues, and should provide a lower bound on the trace's measurement loss rate.[7]

We note that, along with the head/tail effect and the need to align trace pairs as discussed above, one other effect apart from measurement loss can also lead to orphans. Above we discussed how our employment of a 5 msec threshold to eliminate phantoms covers 99.998% of the intervals between known phantoms in the broadcast traffic (i.e., sole-sourced packets). Therefore, in about 0.002% of the cases we miss removing some phantoms. This leaves more traffic in the "phantom-less" traces than we actually should have, and we can then in turn misconstrue these extras as orphans because of the lack of a corresponding packet in the other trace.

Analyzing our dataset with the above described orphan detection method yields 797 orphans across all traces. In the context of the entire dataset this reflects a 0.007% measurement loss rate. We find that nearly 50% of the traces have no orphans. The maximum measurement loss rate estimated using this (conservative) method is 0.2% for one particular trace, roughly four times larger than the next largest. (As discussed above, we defer our look at the individual per-trace estimates to Figure 4 below.)

5.2 Leveraging Phantoms

Next we examine a second strategy to assess measurement loss that leverages the fact that we *expect* multiple copies of each broadcast packet—phantoms—to appear in our traces. For instance, when the full complement of switch ports is active we would expect to find 5 exact replicas of each broadcast packet in the trace file. This expected redundancy allows us to consider decreases in the replication level as possibly reflecting measurement loss.

An immediate issue for this approach is determining the expected replication level. We know this should be no more than 5 due to the number of ports we tap at once. However, the number of active switch ports varies across time in our traces (e.g., as end hosts are powered on and off). We therefore analyze the traces to delimit intervals across which the replication level remains constant, and then use the size and duration of variations in adjacent intervals to estimate loss rates.

We might reasonably expect to find in each trace a relatively steady baseline replication level that holds for a long time (minutes), pockmarked with brief depressions in the level caused by measurement loss. However, across all of the traces we find that the median interval length was just 3.4 seconds, the 75^{th} percentile 32 seconds, and the 95^{th} percentile just over 10 minutes. This indicates that there is not necessarily a solid baseline on which to base our analysis and measurement loss detection. Further, when the replication level changes it does not then necessarily simply change back to the previous value when some anomalous interval was over. We are largely still attempting to find a way to make sense of the progression of replication levels.

Given these puzzling dynamics, we instead employed a fairly simple approach to establish a plausible lower bound on the measurement loss rate. We observe that for 20% of the replication intervals the following properties hold: (i) the interval includes only a single packet (and its phantoms), (ii) the number of replicas in the interval is less than in the adjacent intervals, and (iii) the replication levels in the adjacent intervals are equal. In other words, these intervals represent a slight depression in an otherwise steady replication level, which likely reflects measurement loss. We use the magnitude of the depression as the number of drops.

We find that one-third of the traces exhibit no evidence of measurement loss using this technique; in the remainder, we find modest levels of measurement loss, topping out at 0.08%. After taking into account the discrepancy in the number of traces with no mea-

[6] Clearly, it could in principle occur anywhere in a trace, but we would not expect that, given how the operators told us they had acted. Furthermore, our "layout" analysis in § 6.2 provides strong evidence that this phenomenon did not occur.

[7] Note that when only considering broadcast traffic, finding an orphan in the manner we sketch means that we missed *all* instances of a particular packet.

surement loss, the distribution of loss rates determined using the orphan analysis and the replication analysis match fairly closely. The discrepancy in the number of traces showing no measurement loss likely arises due to the redundancy provided by broadcast traffic. Orphans indicate the absence of *all* copies of a packet from a given trace. Since we know that in general the monitor will capture for each trace multiple copies of each broadcast packet, orphans thus indicate a fairly significant loss event. In contrast, losses detected via a slight depression in the replication level do not require the same high bar for detection, and therefore we observe such loss in a larger number of traces.

5.3 TCP Sequence Gap Analysis

We can assess measurement loss in subnet traces in a quite different fashion by leveraging the structure of TCP transfers. Since TCP provides a high degree of reliability, in the absence of measurement loss we should only observe a receiver acknowledging data that a trace shows was previously transmitted by the sender. Thus, if we observe an acknowledgment (ACK) for a range of sequence numbers for which the trace lacks a copy of the corresponding data transmission, we can infer with high probability that a measurement loss occurred.[8]

We note that from a trace we can directly compute the volume of missing data in terms of TCP payload bytes, but not the number of lost packets, other than by making assumptions about the size of the missing packets. We also note that making similar inferences regarding missing TCP ACKs is significantly more difficult, as these measurement losses manifest by the TCP sender seemingly transmitting too aggressively for correct congestion control. Due to variations in how senders implement congestion control, accurately identifying these situations requires developing a model of the sender's particular algorithms, a problematic undertaking [6]. For our purposes, the simpler estimate based on ACKs that cover data sequence gaps suffices, since our aim is to develop a cross-check on the measurement loss estimates formulated earlier in this section.

We proceed as follows. For each trace, we processed it using the Bro network intrusion detection system [8], which performs TCP stream reassembly in its analysis. Bro's reassembly process already includes instrumentation for detecting ACKs that span sequence gaps. We extended this bookkeeping to count not only how often such ACKs occur, but also the volume of missing data in the gap(s), as well as how often the system processed a "candidate" ACK that *could* have exhibited a sequence gap (i.e., an ACK that includes a new sequence range not previously processed) and the total volume of new data covered by such ACKs. We only apply this analysis to fully established TCP connections, to eliminate ambiguities that arise for traces that miss the beginning of connections (and thus it's not clear just which data sequence numbers might have already been transmitted prior to the beginning of the trace), and also for connections for which one side has already closed the connection (these can lead to sequence number inconsistencies in the presence of RST packets).

We then compute two estimates of measurement loss: L_P, the number of ACK packets with sequence gaps divided by the total number of candidate ACK packets; and L_B, the volume of data (in bytes) missing in the sequence-gap ACKs over the total volume of new data acknowledged by candidate ACKs.

Neither of these values directly measures the loss rate in terms of fraction of missing packets. L_P can be an underestimate if gaps found for single ACKs cover multiple absent packets. Likewise, L_P can yield an overestimate if ACKs cover multiple packets of which only one was missing (for example, ack-every-other could yield for L_P a value of 1.0 if every other packet is missing, rather than 0.5). L_B will emphasize the presence or absence of larger packets over shorter ones, which again could constitute either an underestimate or an overestimate.

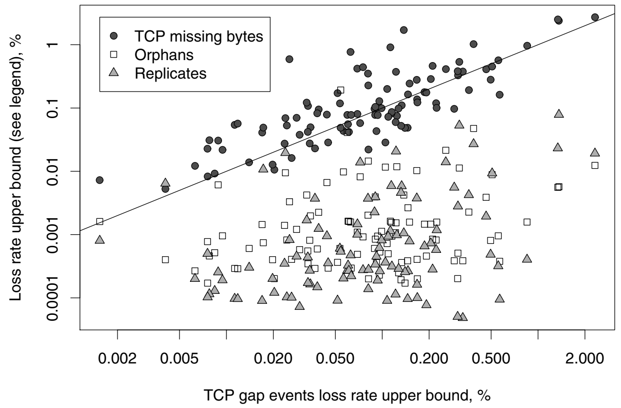


Figure 4: Estimated measurement loss rates computed using different approaches. The X-axis gives rates based on the frequency of observing gaps in the data acknowledged by TCP receivers. The Y-axis shows rates estimated from the number of missing bytes in such acknowledgments, as well as for the analyses developed in § 5.1 and § 5.2.

That said, we would expect in aggregate that both measures should often get us within say a factor of 2 of the true measurement loss rate, if we assume that the measurement loss process is independent from the particulars of the TCP stream's packet structure. If on the other hand we do not have such independence, then most likely the dependence tends towards correlation of high-rate TCP streams with measurement loss events. In that case, if the sender employs full-sized packets then L_B should come close to estimating the true measurement loss rate, while L_P will overestimate due to the common use of ack-every-other for TCP connections with large receiver windows (and hence few delayed ACKs that cover only a single packet).

Figure 4 compares L_P (X-axis) with L_B (Y-axis, circles) for each trace, with both rates in terms of percentages. The diagonal line marks equality between the two, so we see that while L_B estimates tend to run a bit higher than L_P, overall the two track one another fairly closely and do not exhibit a clear-cut skew. We also plot the estimates derived from mismatches in the number of broadcast "orphans" (per § 5.1, plotted with squares) and reductions in the replication level (per § 5.2, plotted with triangles). We see that the orphan- and replication-level-based approaches consistently give considerably lower measurement loss estimates, which fits with their more conservative constructions.

Note, for all four estimates, we plot not the direct estimate but upper bounds computed in terms of observing one more measurement loss than what actually appeared in the trace. For example, for L_P we show the rate we would have observed if simply one more candidate ACK had spanned a content gap, and for L_B if we had seen 1460 more missing data bytes. We aim with using these bounds to control for the granularity of each loss-rate estimate; for traces with few candidate ACKs / orphans / depressions in replica-

[8] Malfunctioning TCP receivers can in fact send acknowledgments for data never sent, but this situation occurs only quite rarely [8].

tion level, we would like to err on the side of overestimating measurement loss rather than underestimating it.

The fundamental conclusion we take from these measurement loss-rate estimates is that (i) often measurement loss is quite low (with the upper bound below 0.1% for the majority of traces), and (ii) it very rarely exceeds 1%.

6. MAPPING THE MONITORED SUBNET

The final class of calibration issues we address concern those we framed previously as relating to *layout*: identifying key topological aspects of each trace. These include determining which traffic reflects intra-subnet vs. inter-subnet communication, identifying which end systems the tapping directly monitored, and detecting instances of hidden switches/hubs, i.e., instances where in fact a tapped link does not lead directly to a single end system, but rather to another network element that provides connectivity to multiple additional systems. All of these characteristics have potentially significant import for measurement analysis, particularly for studies that emphasize traffic locality or that require comprehensive end-system tracing (i.e., capturing all of an end system's network activity).

We note that for traditional measurement vantage points, layout issues are often very simple to resolve. For example, when monitoring a site's access link one immediately can distinguish internal from external end systems, and when recording traffic directly on an endpoint there is no question of confusion regarding hidden network elements. However, enterprise switch measurements introduce significant complications for understanding layout due to the nature of the vantage point they reflect.

One might think that the way to deal with questions of layout is proactively: ensure that operators accurately record such information as they capture the traces. However, this solution is deficient in two regards. First, due to human error such records may in fact not match the reality of what the traces captured. Second, the operators might simply not *know* all of the layout particulars—especially a possibility with regard to hidden switches, which users can potentially deploy without ever informing the operator of their presence (unless the operators enforce that switch ports only accept traffic from registered MAC addresses).

Thus, while meta-information from operators is a highly useful resource, as is generally the case when pursuing sound measurement-based analysis, it behooves us to consider alternative or additional ways of calibrating traces with respect to layout issues. It turns out that elements of such calibration are quite subtle and take considerable care to develop and apply. Furthermore, some elements of determining layout characteristics rely upon others, so we need to proceed in a deliberative fashion. We do so by first considering the problem of distinguishing between intra-subnet and inter-subnet traffic § 6.1. We then employ information gained from that analysis to determine with high confidence which MAC addresses correspond to systems on the other end of monitored switch links § 6.2. Finally, we combine both forms of information to assess whether the links monitored in the traces include multiple hosts behind hidden switches § 6.3.

6.1 Intra- vs. Inter-Subnet Traffic

Our first task is to reliably determine which traffic flows reflect traffic that stays inside the Ethernet subnet (broadcast domain) or involves a remote endpoint outside the subnet. We will primarily analyze IP traffic (which dominates our traces), with brief comments about non-IP traffic.

For clarity of discussion, we adopt the following notation. Let A and B reflect two hosts involved in communication. Unless otherwise stated, we will assume that we are analyzing a unicast, unidirectional flow of packets sent from A to B. For these packets, let M_A and M_B stand for the corresponding MAC addresses as seen in the traces, and I_A and I_B the corresponding IP addresses.

We bootstrap our understanding of whether A or B lies external to the subnet as follows. We assume we can readily identify IP addresses corresponding to hosts external to the entire enterprise. If I_A is such an address,[9] then M_A must correspond to a router's MAC address. For each trace, we gather up all such M_A's. Inspecting these, we observe across our entire dataset just three distinct MAC addresses. Thus, with high confidence we conclude that those three addresses correspond to IP routers, and traffic involving them either is communication directly with the router (presumed rare) or leaves the subnet. In the latter case, it is either inter-subnet traffic if sent to an IP address internal to the enterprise, or WAN traffic if not.

We are not yet done, however, because the enterprise's topology might include both WAN routers and internal routers used only for inter-subnet communication. The above approach will not have found those, since we seed it with M_A's that reflect WAN addresses. So we next remove any traffic involving these identified routers. The remaining traffic is either entirely intra-subnet (if there are no other routers) or potentially involves other subnets at the site. For this traffic subset we compute the range of IP addresses seen in the flows (again taking care to remove nonsensical values that arise from configuration failures). We then widen this range to the nearest accommodating CIDR prefix. In our case, we know that the enterprise employs subnet blocks in the range of /24 to /22. If the addresses in the possibly-entirely-intra-subnet traffic fall into such a prefix, then we have good confidence that it indeed reflects only intra-subnet traffic.

For our traces, that is in fact what we found—for each trace, the possibly-internal traffic always fell within a CIDR prefix at most /22 in width. We then asked the operators whether the prefixes correctly described the enterprise's subnets. In all cases they did, except sometimes the inferred prefix was narrower than the actual prefix, due to our traces not happening to include the full range of intra-subnet IP addresses.

We could in principle apply a similar process for non-IP traffic, too. However, doing so is complicated by the need to understand the specific inter-subnet forwarding/routing employed by what are sometimes fairly obscure link-layer protocols. Instead, we simply confirmed with the operators that indeed the enterprise does not route any non-IP traffic between Ethernet subnets.

6.2 Determining Monitored Hosts

The next part of developing a network map is to figure out which nodes were directly monitored in our switch-level traces. The issue arises due to a basic ambiguity: if we see communication from M_A to M_B, then it could be that the directly monitored hosts were A, B, both, or neither. Our efforts at such identification were significantly complicated by the fact that a number of simple approaches we tried yielded quite unlikely results, finding more often than not that

[9] Note, a complication here arises when an internal host uses an erroneous IP address. Simple such instances we observed in our traces are private or self-assigned addresses, arising from dynamic configuration failures. More subtle cases use *legitimate* external addresses, even though the host employing them is operating inside the enterprise. These appear to come about due to mobile hosts that dynamically configured an address when at a location external to the enterprise, and now attempt to continue to use it rather than dynamically configuring another address. We can identify such instances by their quite low prevalence, and also by their failure to engage in productive two-way communication.

the traffic patterns appeared explainable only in terms of H directly monitored hosts for $H > 5$. Such a situation should not arise unless one of the monitored ports in fact leads to a hidden switch, and we believed that such switches would not be particularly common.

The approach we eventually developed—a generalization of the initial simple approaches—works as follows. First, we locate all flows of the form A communicating with B for which (i) the large majority of packets (90%) were *not* replicated, (ii) we similarly saw mostly non-replicated traffic from B to A, and (iii) at least 5% of the traffic (constituting at least 5 packets) flowed in each direction.

The goal behind these constraints is to find flows for which we can state with confidence that *either* A or B must be a monitored host, but *not both*. The reasoning proceeds as follows. If A and B are both non-monitored, then the switch(es) that mediate their traffic should learn forwarding paths between the two, and those paths by definition do not include our monitored ports. We should only see packets for their flows in those instances where the switches lack a forwarding entry for the destination; thus, any instances of their packets that appear in our traces should be replicated, violating (i).

In addition, we know that it is not the case that *both* A and B are monitored hosts (assuming no hidden hubs, a point we return to below) because if they were then the strong majority of the packets between the two would appear *twice* in the trace, once for each monitored port. Indeed, we locate such "doubly-monitored" flows by finding those for which a replication level of 2 copies predominates ($\geq$ 75% of packets). These are present in 34% of our traces.

Finally, we employ a simple form of graph coloring employing two colors, *red* and *green*, as follows. We construct a graph G where for each bidirectional flow between an A and B that satisfies (i), (ii) and (iii) above, we place an edge between nodes that represent the two systems. We first color *red* the nodes corresponding to the routers we identified in § 6.1, and *green* any nodes to which they have edges. (All traces had at least one router present in them.) We then recursively continue coloring in this fashion, alternating between *red* and *green* every time we traverse an edge.

If we arrive via an edge at a node that we need to color *red*, but the node is already *green*, or vice versa, then we have detected an inference inconsistency, and we abort. When done, we also check to ensure that any nodes corresponding to doubly-monitored flows were both colored *green*.

At the end of this process, we have a collection of nodes colored *red*, which reflect non-monitored nodes (since they reside in an equivalence class with a router, and we have high confidence that the monitoring didn't include a link to the router); *green*, reflecting monitored nodes; and uncolored, reflecting nodes that were unreachable from the original set seeded by the router's flow activity (i.e., G has disconnected components). We could in principle color these disconnected components too, though we would not know in that case which color corresponds to monitored hosts vs. non-monitored ones. However, these occur in only 6 traces, in each case for very small components. We discuss treatment of these uncolored nodes below.

In none of the traces did this approach produce a coloring inconsistency. This gives us additional confidence that the approach does indeed uncover a basic facet of the monitoring layout; i.e., the *green* hosts very likely do correspond to monitored hosts.

A final confirmation in this regard, albeit an unexpected one, was our discovery that the deduction process often flagged the same MAC addresses in consecutive traces recorded by the same aggregation interface on the monitor device. This behavior manifested primarily on traces captured over a weekend, and has a natural explanation that for the second trace, the operators did not in fact move the taps to a new set of ports.

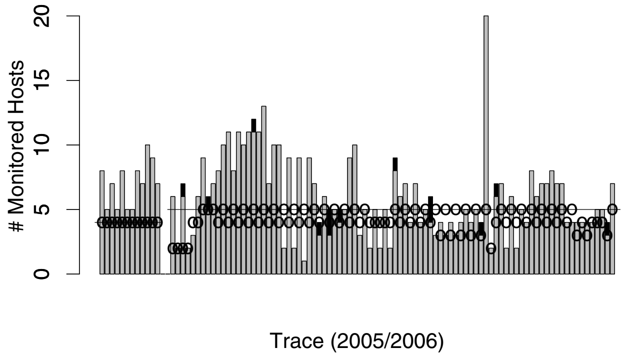


Figure 5: Deduced monitored hosts per trace: number seen concurrently (grey), total (black), observed replication level (circles).

Having done this work to deduce the number of monitored hosts per trace, we now face a conundrum, as illustrated in Figure 5. Here we plot for each trace the total number of monitored (*green*) nodes. For those traces with uncolored nodes, we add half (rounded down) of their total number, since the disconnected component represents at least that many monitored nodes. (For example, if the component shows hosts A and B both communicating with C, then either A and B are monitored, or C is.) Doing so only adds one deduced node to each of 3 traces.

The bars on the left of the plot reflect traces from 2005, which each tapped 4 ports. The righthand group, from 2006, tapped 5 ports per trace. The plot includes horizontal lines showing these cutoffs. Finally the circles in the plot reflect the largest replication level seen for each trace.

The conundrum arises from the fact that for the majority (55%) of the traces, we deduce *more monitored hosts than tapped ports*. This number further rises to 67% if we consider instances where we deduced more monitored hosts than the largest observed replication level.

This mismatch perturbed us. It implies either (a) we somehow are seeing unreplicated packets for ongoing communication between two non-monitored hosts, indicating that our understanding of switch operation is incorrect; (b) end systems employ multiple MAC addresses when using a single network link; (c) at different times, different end systems are plugged into the same network link; (d) the network operator performing the monitoring moved the taps to different switch ports during the middle of tracing; or (e) our links did not connect to end systems but instead to network segments shared among multiple hosts via hidden switches or hubs. We cannot directly assess the likelihood of (a). We find (b) quite unlikely to occur often enough to affect the majority of traces. Similarly, while (c) will likely sometimes occur, it struck us as implausible that it would do so frequently enough to cause the observed overrun of monitored hosts, particularly for the larger counts, and similarly for (d). While it would not be surprising to find (e) occurring, the impression we had formed was that for the LBL network it too would be an occasional, rather than dominant, effect.

We set out to test (c) by splitting each trace into 15-minute intervals and counting for each interval the number of deduced monitored hosts that appeared active in both that interval and the one immediately following it. We consider such hosts as simultaneously active, and therefore unable to reflect different systems plugged into the same link at different times. We then looked for the 15-minute interval in the trace that had the maximum number of such concurrent host activity.

Figure 5 shows these maximum values in grey, drawn over the total number of deduced monitored hosts, which we show in black. The fact that the plot appears virtually entirely as grey bars thus indicates that *for almost all traces,* ***all*** *of the deduced monitored hosts appeared together*. This finding conclusively rules out (c).

In the process of examining 15-minute epochs, we also computed over what proportion of all of a trace's 15 minute intervals did each monitored host appear active. We found that in all but 5 of the traces, there was at least one monitored host that appeared active during *every* 15-minute interval, and in only two traces (both very lightly loaded) did the most active monitored host appear in less than 75% of the intervals. Thus, we also can conclusively rule out (d).

This then leads us to the problem of considering whether (e) (widespread use of hidden switches/hubs) indeed explains the discrepancies in Figure 5, or if we must consider the seemingly very unlikely explanations of (a) or (b).

6.3 Detecting Hidden Switches and Hubs

We now turn to the problem of determining whether our monitored ports in fact frequently (more than half the time) included links that run not directly to individual end systems, but instead to "hidden" switches or hubs that provide connectivity to multiple end systems. If we cannot establish the frequent presence of such hidden elements, then we are forced to consider alternative, seemingly quite implausible, explanations for the excessive number of apparently monitored hosts in Figure 5.

This is a difficult problem, since we lack the ability to associate a given MAC address directly with a given switch port. However, we eventually devised the following methodology for detecting the presence of some (not all) instances of hidden elements. Our key observation is that *if* our traces often include hidden elements, then during the recording of at least some our traces communication probably occurred between hosts A and B, both of which communicate directly using the hidden element.

If the hidden element is a hub, then due to its broadcast nature we will see a copy of the traffic between A and B on the monitored link. This in turn will lead us to create an edge between the nodes corresponding to A and B in the graph G we devised in § 6.2, and thus we would discover a graph-coloring inconsistency (either A or B could be colored *green*, but not both). However, we did not discover any such inconsistencies, which gives us confidence that our traces do not include numerous hidden hubs.[10]

Detecting hidden switches, however, presents a more difficult problem. If A and B connected to the hidden switch communicate directly, then the switch might not replicate *any* of their traffic, or at best only their initial packets, so we will have little opportunity to observe the activity on the tapped port. However, we can infer such communication as follows. Whenever two hosts begin communicating via IP for the first time—or after a lengthy lull or a reboot—each will generate an ARP request for the other. These requests are normally broadcast, since the whole point of needing to use ARP is that neither A nor B know what MAC address to associate with the other. However, replies to ARP requests come back to the requester via unicast. In addition, normally the initial ARP request will have provided the switches between A and B with awareness of M_A (A's MAC address), and thus the switches will *not* replicate the ARP reply.

These dynamics then lead to the observation that if A and B connect to a hidden switch, then when they communicate with one another we will sometimes see A sending ARP requests for B, and likewise B sending ARP requests for A—*but we will not see ARP replies in either case*. We see the requests due to their broadcast nature; the hidden switch replicates the request onto the monitored link. But the replies will proceed directly back through the switch, without replication onto the link.

We can therefore sometimes detect hidden switches in the presence of such intra-switch communication by identifying instances where a host A sends an ARP request for a host B, for which we do not see a reply; and host B likewise sends an apparently unanswered request for A. If from graph coloring we have that both A and B are *green*, then we can say with high confidence that both reside behind a hidden switch: we know that both are located on a monitored port due to their (separate) communication with external hosts, and thus were they not behind a hidden switch we should have seen the corresponding ARP replies. If both A and B are *red* then we know they both reside externally (this will be a common case when we see only the broadcasted ARP requests, but not the unicasted ARP replies). Finally, if one is *green* and the other *red*, then we have discovered an inconsistency. (This last did not in fact occur.)

This procedure presents one difficulty, however. We do not directly observe "host A" sending an ARP request for "host B". Rather, we observe M_A sending an ARP request for I_B. Since we do not see the reply, we do not directly obtain the pairing of I_B with M_B, and therefore cannot immediately match the ARP request from M_B for I_A as the other half of the communication setup between A and B. We address this consideration by generating for each trace a mapping of all MAC/IP address pairs seen in any IP packets (whether as source or destination). We then look for any instance where M_A makes an unanswered ARP request for I_B, for which a MAC address M'_B seen associated with I_B also sent an unanswered ARP request for I'_A, an IP address seen associated with M_A.

Applying this approach, we find that 14 traces manifest such communication setup between a pair of *green* hosts, with half of those including more than one pair of intercommunicating hosts. We accordingly conclude that hidden switches are not that uncommon in our traces.

Furthermore, our finding of 14 instances is an *underestimate*, because it requires that a pair of the hosts connected to the hidden switch not only happened to communicate with one another during the tracing period, but also had to refresh their ARP caches for each other. Above in § 6.2 we found that 34% of traces included doubly-monitored flows between pairs of monitored hosts. (These hosts are *not* behind hidden switches, as we see two copies of the packets of the flow, indicating that the hosts reside on separate monitored ports.) We might then roughly estimate that among a group of hosts connected to a switch, during a day-long period only about a third of the time does one of the hosts attached to the switch communicate with another host attached to the switch. If that rate happens to hold for hosts connected to hidden switches too, then the 14 instances we observe might be only about a third of the total (less, in fact, due to our need for the hosts to ARP for one

[10] It is possible for a given trace that A and B both reside in an unconnected component of G, and hence were not colored at all. However, we know that very few traces had such components, and therefore there are at most just a few hidden hubs.

another). We would then extrapolate that all-in-all, at least ≈ 42 of our traces—and possibly significantly more—might include hidden switches. This figure is fairly consistent with the 55–67% of traces exhibiting aberrant levels of monitored hosts as seen in Figure 5.

Thus we argue that we have good evidence that hidden switches are widely prevalent in our dataset.

7. TRAFFIC LOCALITY

After the arduous process of calibrating our datasets, we can now briefly turn our attention to developing high-level characterizations. We view the results in this section as motivating the work presented above, because the analyses show that subnet-level traffic patterns manifest quite differently from both wide-area traffic and internal traffic that leaves the subnet. Therefore, while the above calibration efforts may seem like mundane logistical issues, they indeed form the key foundation for then obtaining sound insights into a rather unexplored area of networking.

The first step in characterizing enterprise traffic is to attribute each packet to a particular *locality*. In our analysis we distinguish three such forms: (i) *subnet* specifies traffic that stays strictly inside a given broadcast domain, (ii) *LBL* contains traffic that leaves the subnet, but remains confined inside the LBL enterprise, and (iii) *WAN* denotes traffic that involves communication with hosts external to LBL.

For mapping packets into locality bins, we leverage an observation made in § 6.1: for the *LBL* and *WAN* bins, one of the MAC addresses must belong to a router. Thus, for each packet we apply the following rules: (i) if none of the MAC addresses corresponds to a router, we put the packet in the *subnet* bin; (ii) if one of the MAC addresses is a router, and the corresponding IP address belongs to LBL, the packet falls into the *LBL* bin; and (iii) if a MAC address is a router, and the corresponding IP address is outside LBL, then the packet belongs to the *WAN* bin.

We find three types of packets that do not fall into any of these locality bins. For all three, we observe the router's MAC address as the source, but instead of IP, the packets are: DECnet, CGMP, or ARP. The operators informed us that LBL did not route non-IP traffic at the time we captured the traces, and therefore these packets must have originated from the router itself. Indeed, CGMP packets are sent by routers to populate Catalyst 5000 switches with multicast-aware entries, and ARP to resolve unknown MAC to IP mappings. The number of DECnet packets is very low (no more than 200 packets per trace),[11] while CGMP appears in fairly high numbers in less than half of our traces. Both of these protocols, unlike ARP, sent packets exclusively to Ethernet multicast addresses. Since all three protocols operate inside a subnet, we deem it plausible to designate all of them as *subnet*, i.e., local to the LAN.

Figure 6 plots the relative traffic locality mix across all traces. We first note the wide variety of bin size proportions in our dataset—illustrating that it is impossible to devise a single rule-of-thumb regarding locality patterns in an enterprise. Additionally the plot suggests that in half of the traces subnet traffic dominates, and thus previous studies of traffic captured from a router's vantage point have missed much of the activity taking place in an enterprise—again underlining the importance of obtaining enterprise switch measurements.

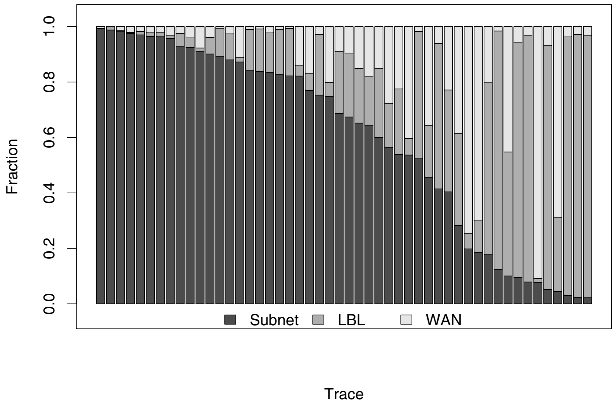


Figure 6: Traffic locality mix. The X-axis is sorted by the *subnet* bin share in descending order.

In addition to variability in locality, we see variety in network and transport protocol usage across locality as well. While by definition *LBL* and *WAN* traffic solely involves IP, the *WAN* traffic is furthermore mainly TCP—with a median share across traces of 97%—while in the *LBL* traffic, UDP dominates, with a median share of 58% across traces. In the *subnet* IP traffic, the proportions of TCP and UDP range from 1% to 99% (!), with medians of 36% and 60% respectively. IP traffic prevails over non-IP at subnet locality, with a median of 72% vs. 29%. The dominant non-IP traffic is ARP (54%), LLC (34%) and IPX (6%). Ethernet multicast traffic comprises 37% of the packets in the subnet bin and mainly consists of non-IP traffic (97%)

Finally, we turn to a high-level view of dominant network, transport and application protocols. We plot the most dominant protocols in each locality in Figure 7, with circle radii corresponding to the relative volume in number of packets. To ensure readability, we include only the 15 most frequent protocols in each locality.[12]

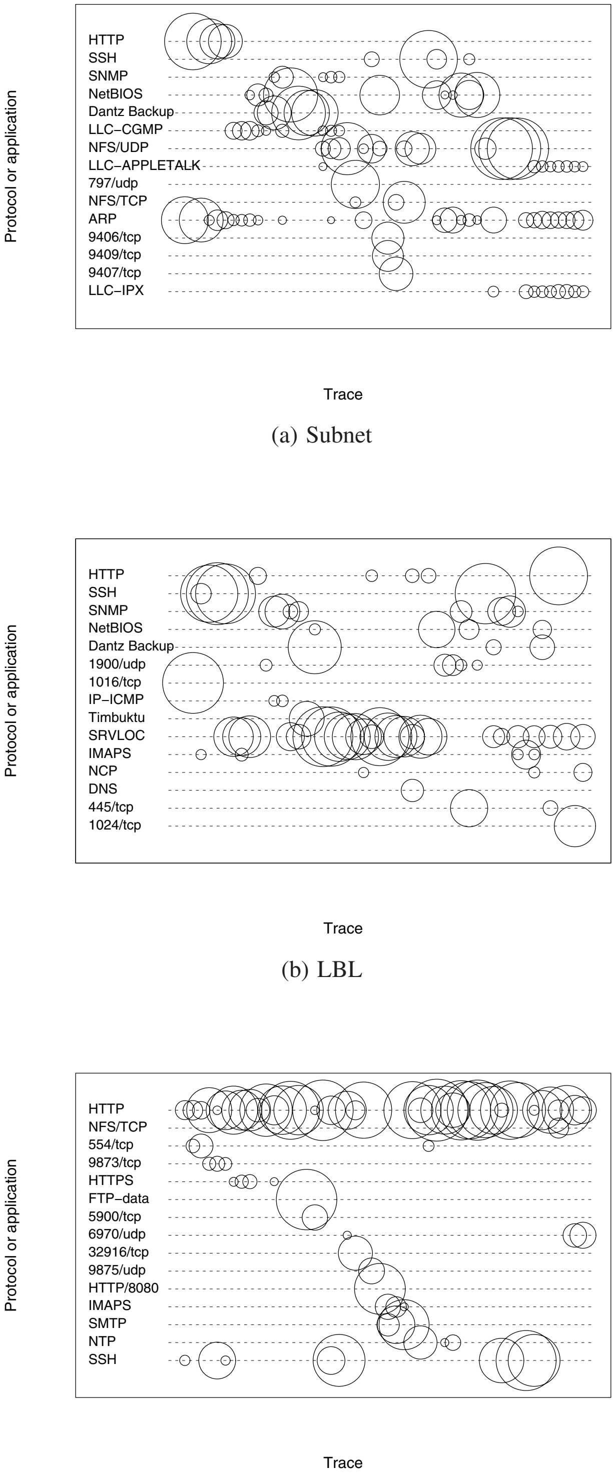
We found only two protocols that appear for each category of locality: HTTP (80/tcp), and SSH (22/tcp). Three more protocols (see the top of the plots) appear common both to *LBL* and *subnet*. Finally, NetBIOS (139/tcp), Dantz (497/tcp), NFS (2049/udp) and ARP prevail in intra-subnet traffic. In the *LBL* category we also find svrloc (427/udp) in large proportions in majority of traces. This set of plots clearly shows massive heterogeneity of traffic across both type of locality and set of monitored ports.

An illuminating way to underline the importance of calibration is to present a showcase that directly compares some properties of calibrated and non-calibrated traces. For this purpose, out of the four calibration aspects discussed in this paper we choose "gain," as it offers an intuitively expected and easily visualizable contrast. By analogy with Figure 7 we calculated relative protocol shares in non-calibrated traces.[13] We plot the difference in shares between original and final traces in Figure 8. We present subnet locality, which exhibits the most striking contrast. In interpreting the plot, the reader should bear in mind that it shows changes in relative prevalence among the different protocols, rather than absolute

[11] We have not yet formulated a plausible reason for DECnet packets to originate at the router, and there are some indications that the enterprise's routers may have actually routed DECnet traffic in some cases. However, in the amounts we observed the traffic will not skew our results, and therefore we did not further analyze these packets for this initial study.

[12] The plots are not meant to capture a comprehensive picture, but to illustrate the dominant protocols.

[13] We used the raw traces, for which none of the calibration techniques were applied.



(a) Subnet

(b) LBL

(c) WAN

Figure 7: Major protocols and applications.

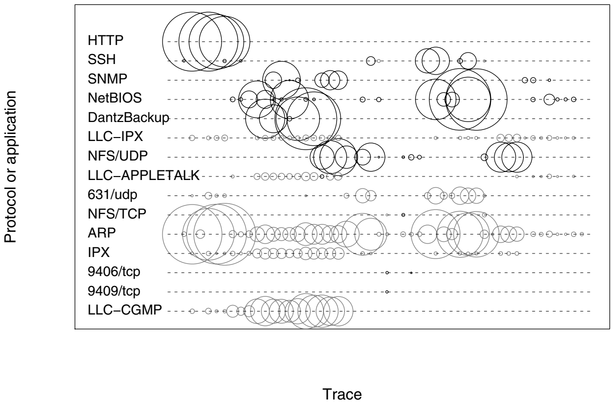


Figure 8: Difference in subnet protocol shares between calibrated and non-calibrated traces. Circle diameter scales with change in share, with black indicating a greater share in the calibrated trace and grey in the non-calibrated. The largest circles correspond to a change of about 40%.

changes. Thus, a protocol like HTTP, which itself does not change in attributes much between the uncalibrated and calibrated traces, can exhibit significant change in its overall share of the traffic. In general, the plot highlights how correct calibration can have a dramatic effect on the accuracy of determining traffic mix.

8. SUMMARY

In this paper we have presented a number of techniques for calibrating packet traces captured at switches connecting end hosts in terms of: *gain*, *loss*, *timing*, and *layout*. While we have developed concrete strategies that we successfully employed with our dataset, we view the main contribution of this paper to be the *reasoning* about how to go about calibrating such traces in a sound fashion. In particular, we identified the following key notions: (i) using *sole-sourced* packets as unambiguous “stakes in the ground” to hunt for thresholds and compare clocks, (ii) employing *expected replication* of broadcast packets to point to missing events from traces and aid in mapping networks, (iii) leveraging *TCP semantics* to identify measurement loss, particularly in terms of seemingly erroneous acknowledgments for date we never observed in transmission, and (iv) leveraging *multiple, simultaneous* data collections to further illuminate unrecorded events and bolster confidence in the timestamping process. These general principles apply to similar collections, and we encourage others working with enterprise traces to calibrate their analyses using strategies in this paper, though not necessarily with the same fine-grained details (e.g., the 5 msec threshold for removing phantoms). These fine-grained constants and thresholds may well not hold with different switches and monitoring gear. Further, when collecting new datasets we encourage researchers to do so in a way that they can leverage the general concepts we have outlined in this paper to calibrate their final dataset. Finally, we have illustrated the importance of collecting switch-level measurements, as observations from other vantage points will manifest clear differences at the various locality scopes present in our data.

Acknowledgments

Mike Bennett and Jason Lee of LBL captured the traces used in this work. Mike has also provided numerous key insights into the measurement process and the operation of LBL's Ethernet subnets. We thank Ran Atkinson and Tom Kho for fruitful discussions on topics related to this paper. We are also grateful to the anonymous reviewers, whose comments helped to improve this paper. This work was supported in part by TEKES as part of the Future Internet program of the ICT cluster of the Finnish Strategic Centres for Science, Technology and Innovation. This work was funded in part by US NSF grants FIND-0721933 and CNS-0831535, and by a grant from the US DHS.

9. REFERENCES

[1] Colasoft. CGMP (Cisco Group Management Protocol). `http://www.protocolbase.net/protocols/protocol_CGMP.php`, 2006.

[2] F. Giroire, J. Chandrashekar, G. Iannaccone, K. Papagiannaki, E. Schooler, and N. Taft. The Cubicle vs. The Coffee Shop: Behavioral Modes in Enterprise End-Users. In *Proc. PAM*, 2008.

[3] R. Gusella. A measurement study of diskless workstation traffic on an Ethernet. *IEEE Transactions on Communications*, 38(9), Sept. 1990.

[4] IEEE Standards Association. IEEE 802.3 LAN/MAN CSMA/CD Access Method. `http://standards.ieee.org/getieee802/802.3.html`, 2008.

[5] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A First Look at Modern Enterprise Traffic. In *Proc. ACM IMC*, Oct. 2005.

[6] V. Paxson. Automated packet trace analysis of TCP implementations. In *Proc. SIGCOMM*, 1997.

[7] V. Paxson. On calibrating measurements of packet transit times. In *Proc. SIGMETRICS*, June 1998.

[8] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Comp. Networks*, 31(23–24), 1999.