Experiences in Traceroute and Available Bandwidth Change Analysis*

Connie Logg
SLAC
2575 Sand Hill Road
Menlo Park, CA 94025
+1 (650)926-2523
cal@slac.stanford.edu

Les Cottrell
SLAC
2575 Sand Hill Road
Menlo Park, CA 94025
+1 (650)926-2879
cottrell@slac.stanford.edu

Jiri Navratil
SLAC
2575 Sand Hill Road
Menlo Park CA 94025
+1 (650)926-3332
jiri@slac.stanford.edu

ABSTRACT

SLAC has been studying end-to-end WAN bandwidth availability and achievability for 2.5 years via IEPM-BW [1]. IEPM-BW performs network intensive tests every 90 minutes. Based on that experience we have also developed a light weight available bandwidth (ABwE [2]) measurement tool that can make a measurement within a second. We are now extending this to a WAN measurement and detection system (IEPM-LITE) aimed at more quickly detecting and troubleshooting network performance problems and also to be more friendly on lower performance paths. IEPM-LITE uses ping, forward traceroutes, and ABwE sensors to monitor, in close to real-time, Round Trip Times (RTT), changes in available bandwidth and routes to and from target hosts. This paper discusses the experiences, techniques and algorithms used to detect and report on significant traceroute and bandwidth changes. The ultimate aim is to develop a lightweight WAN network performance monitoring system that can detect, in near real time, significant changes and generate alerts.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Network Performance – Measurement techniques, Performance analysis, Trouble shooting

General Terms

Algorithms, Measurement, Performance, Reliability.

Keywords

Bandwidth, availability, capacity, problem detection, anomalous events, traceroutes, real-time alerts, WAN, network monitoring, plateau algorithm.

* This work is supported by U.S. DOE Contract No. DE-AC03-76SF00515.

Copyright 2004 Association for Computing Machinery.

ACM acknowledges that this contribution was authored by a contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

SIGCOMM'04 Workshops, Aug. 30 & Sept. 3, 2004, Portland, OR, USA.

1. INTRODUCTION

Modern High Energy Nuclear Physics (HENP) requires large volumes of data to be effectively distributed between collaborators world-wide. In response to this demand, in September 2001, SLAC embarked on an Internet end-to-end active performance testing and analysis project known as IEPM-BW [1], to evaluate and monitor the performance of the paths to its collaborators. The IEPM-BW system uses network intensive tools such as iperf [3] and bbftp [4] to make achievable throughput estimates for TCP and bulk data applications. Concurrently, a lightweight, non-intensive bandwidth measurement tool (ABwE [2]) has been developed. The intent is to use ABwE to perform similar testing and analysis to that performed in IEPM-BW while impacting the network less and with reduced security requirements. The reduced network traffic is more friendly, enables testing of non-high performance paths (e.g. for paths to developing regions), reduces the need for scheduling, and/or enables measurements to be made on a more frequent basis (e.g. with a frequency of minutes rather than hours) so we can present closer to real-time feedback on critical network paths.

The measurement system we are now developing is referred to as IEPM-LITE. It performs pings, traceroutes, and frequent light-weight available bandwidth measurements using ABwE. Ping and traceroute information are gathered every 10 minutes. ABwE measurements are made at intervals of one to three minutes. The measurements are currently made only from SLAC. The intent, as for IEPM-BW, is to deploy the toolkit to make IEPM-LITE measurements from other major HENP computing centers. The 40 remote hosts monitored from SLAC are located in US DoE Labs (8 hosts) accessed via ESnet, US universities (12) and NASA (1) accessed by CENIC-Abilene, Canada (1) accessed by ESnet-CAnet, a .org host (1), European Research and Education (R&E) facilities (.ch (2), .cz (1), .de (2), .fr (1), .it (2), .nl (1), .uk (3)) accessed by ESnet-GEANT-national R&E networks, Japan (3) accessed by ESnet-SINET, Pakistan (1) accessed by CENIC-SingTel-PakTel, and Novosibirsk(1) accessed by ESnet-SINET-Rostelcom. The bottleneck capacities of the paths from SLAC cover a wide range from 512Kbits/s (Novosibirsk) to 1Gbits/s.

A drawback of this and other monitoring performed by SLAC is that there are hundreds of reports and graphs generated daily and no one has time to look at them all. Critical performance changes often go unnoticed. For example, in August 2003 [5], we did not notice until about a month after the event that iperf throughput

from SLAC to Caltech had dropped dramatically due to incorrect route advertisements and consequent poor route selection. Once this drop was noticed and investigated, it was fixed in about 4 hours. If we had been alerted to this change, it could have been reported earlier, and fixed in a more timely fashion.

As a network end-user one of our goals is to automatically detect major decreases in bandwidth to our collaborator sites. Currently our main interest is in changes that persist for sufficient time to be able to review the change event and if appropriate take action (e.g. report it to our upstream provider's Network Operations Center). Further the bandwidth change events that interest us are typically step down changes, such as those often caused by route changes, rather than incremental changes caused by traffic or host related congestion. As shown in [6], very few route changes result in a bandwidth change. Further many significant step changes in bandwidth are not associated with route changes, and are presumably caused by non layer 3 devices such as switches or sudden changes in utilization.

This paper discusses the techniques for bandwidth change event detection and traceroute analysis that we are currently developing to generate alerts, and when possible, correlate available bandwidth changes to route changes.

2. BANDWIDTH CHANGE DETECTION

For each measurement ABwE provides three bandwidth estimates for a path: the dynamic bandwidth capacity (*Cap*) by analyzing the minimum inter-packet delay; the cross-traffic (*Xtr*) by analyzing the inter-packet delay variability; and the available bandwidth (*Abw* = *Cap* – *Xtr*). Of these probably *Abw* is of most interest to a user, however it is more sensitive to cross-traffic over which we have little control. Changes in *Cap* are more likely to be caused by route changes or operator errors etc. *Cap* estimates are thus generally preferred for our work. However in some cases the minimum packet separation is observed to be relatively fixed at the equivalent of a 1 Gbits/s link[1] and thus insensitive to events. So we have analyzed both *Abw* and Cap data. The ABwE bandwidth estimates are also very rough since each is made by analyzing the separation of only 20 packet-pairs. Estimates can thus vary dramatically from minute to minute and have large outliers. Therefore, ABwE also provides smoothed data using an Exponential Weighted Moving Average (EWMA) [7]. EWMA predicts the next value in a time series given the current value z_t and the current prediction as $\hat{z}_{t-1}$ where $\hat{z}_t = z_t(1-\varepsilon) + \varepsilon \cdot \hat{z}_{t-1}$ is the smoothing factor, and ABwE sets $\varepsilon = 0.75$.

The event detection algorithm evolved from work by NLANR [8] to develop a "plateau" algorithm for ping RTTs. It involves buffering the time sequence bandwidth data into two buffers: a history buffer (**h**) for base-lining, or (exclusive) when a datum meets specific requirements, into a trigger buffer (**t**). Each buffer has a maximum number of entries or duration parameter λ and τ respectively. τ determines how long the bandwidth change must exist before an analysis of its data is performed to see if we have encountered a significant change or event. Note that if the ABwE data is taken once a minute, τ is roughly the number of minutes (assuming no data is lost) that a drop must exist before an event may be deemed to have occurred. The history buffer is initialized at start up with λ data points. Once **h** is initialized, we enter the data processing loop. The mean ($\mathbf{m_h}$) and standard deviation ($\mathbf{o_h}$) of **h** are then calculated.

Two other user set parameters besides the buffer lengths are used in the analysis and event detection:

- The sensitivity (β) is the number of standard deviations ($\mathbf{o_h}$) beyond $\mathbf{m_h}$ that a datum must lie to be considered a trigger value. The default value for β of 2 appears to work well in most cases.
- Threshold (δ) is the relative difference between the buffer means $\Delta = (\mathbf{m_h} - \mathbf{m_t},) / \mathbf{m_h}$, that must be exceeded for an event to be detected. Once we are in an event detected state, this threshold must again be met before another event is detected. We are in an event detected state when an event has been detected and we have only seen trigger data.

Setting the buffer lengths typically depends on user requirements. For example, how long a change must be sustained (τ), before it is considered significant, depends on how long the user wants a drop in bandwidth to be sustained before she / he is notified. The setting of δ also depends on the user's requirements.

2.1 Pseudo Perl [9] code[2] for modified "plateau" algorithm

External parameters:
β = sensitivity (default = 2);
δ = threshold (default 40%)
λ = history buffer length (default = 1800 minutes[3])
τ = trigger buffer length (default = 180 minutes)
Code variables:
@y, y = list of & current bandwidth estimates
$\mathbf{m_h}$, $\mathbf{o_h}$ = history buffer mean & standard deviation
$\mathbf{m_t}$, $\mathbf{o_t}$ = trigger buffer mean & standard deviation
$\mathbf{m_e}$, $\mathbf{o_e}$ = event buffer mean & standard deviation
@**h** history buffer, current length = scalar(@**h**)
@**t** trigger buffer, current length = scalar(@**t**)

```
mt = me = ot = oe = 0;
foreach y (@y) {
 if (y > (mh - β * oh)) {#then NOT a trigger
   a=0;
   if(scalar(@h)>λ){a=shift(@h);}#remove oldest
   me = 0;
   if (y > (mh + 2 * β * oh)) {#outlier?[4]
```

[1] This may be caused for example by the scheduling algorithms of network devices such as routers or Network Interface Cards (NICs).

[2] In perl: a variable name with an @ prefix is an array; the scalar function applied to an array gives the length of the array; the shift function shifts the oldest value of the array off and returns it; push treats the array as a stack and pushes the second argument onto the end of the array given in the first argument.

[3] The user specifies λ and τ as times, however, the algorithm uses the time stamps of the first and last data points to calculate the equivalent buffers sizes.

[4] This is to guard against large outliers often seen in the ABwE data.

```
    if(scalar(@t) >0){a = shift(@t); next;}
  }
  if (a < 0 && abs(y - mh) / mh < 0.2) {y = -y}[5]
  push(@h, y); #push y into history buffer
  if (y > 0) {
    (mh, oh) = calcstats(@h);
    #calcstats returns mean & stdev for
    #positive non-zero values in array
    if (scalar(@t)> 0) {a = shift(@t);}
  }
 }
 else {#trigger data
  push (@t, y); #add value to trigger buffer
  if (scalar(@t) < τ) {next;}#enough triggers?
  (mt, ot) = calcstats(@t);#yes, so see if event
  if ( (mh - mt) / mh > δ) {#event?
    unless (me == 0) {#already in event state?
      if((me - mt) / me) >= δ) {
        me = mt; oe = ot;
        foreach t (@t) {push (@h,t);}
        while (scalar(@h) > λ) {a=shift(@h);}
        (mh, oh) = calcstats(@h);
        @t=(); #empty trigger buffer
      }
      else {a = shift(@t);}
    }
    else {#not in event state
      me = mt; oe = ot;
      foreach t (@t) {push (@h,t);}
      while (scalar(@h) > λ) {a=shift(@h);}
      (mh,  oh) = calcstats(@h);
      @t = ();#empty trigger buffer
    }
  }
  else {a=shift(@t);} #no event
 }
}
```

Figure 1 shows the time series of the EWMA smoothed (as a continuous line) and raw (before EWMA smoothing) (as points) bandwidth estimates of *Cap* with an event marked by a circle. Also marked are the current size of the trigger buffer (τ), m_h and $m_h - \beta * o_h$. The analysis was performed on the raw *Cap* data since the long term effects of smoothing may cause the algorithm to miss short changes in bandwidth. For example for the data in Figure 1, δ is 20% for the EWMA *Cap* but 48% for the raw *Cap*, so with a δ of 40% we would miss the event when analyzing the EWMA *Cap* data

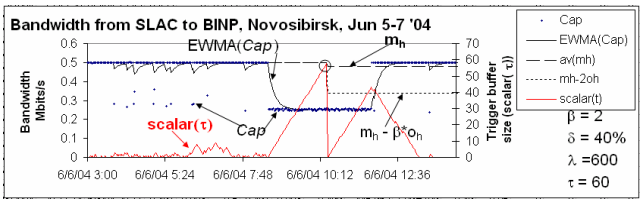

Figure 1. Bandwidth estimates from SLAC to BINP Novosibirsk, June 2, 2004

[5] Following the ideas of [8], only data that differs from the mean by > 20% is used in the statistics calculations. This is to prevent long periods of low bandwidth variation (such as may be observed for the smoothed data) from preventing the means and standard deviations tracking the most recent trends.

2.2 Commentary

If there are breaks in the data, they are ignored, and the analysis continues.

Besides adjusting the algorithm of [8] to detect step downs, we added the threshold check when the event buffer fills. We replaced the use of variances by standard deviations when separating the trigger data from the history data. We also observed that the bandwidth can drop gradually over a long period of time. Data flushed from the trigger buffer when a full trigger buffer does not indicate an event can gradually lower the mean of the history buffer. Therefore, data dropped from the trigger buffer, as an examination shows that no event is warranted, is discarded (not moved to the history buffer).

2.3 Parameters

Given reasonable values for the sensitivity and threshold, τ has the most dramatic effect on the frequency of event detection. It determines the amount of time the bandwidth must be depressed in order to check for an event. Figure 2, illustrates the effect of τ. The open boxes on the top axis indicate that an event occurred at that point in time.

In Figure 2, τ = 20, i.e. since the data points for these EWMA *Abw* measurements are ~1 minute apart, the bandwidth must be depressed for ~20 minutes before an event is detected. With a δ of 60% we detect 9 events[6]. For τ = 60 no events are detected, and for τ = 40 minutes, 3 events are detected (those on 4/9/04 at ~13:17 and ~21:20, and on 4/10/04 at ~13:30; the third, fifth and eighth squares in the figure).

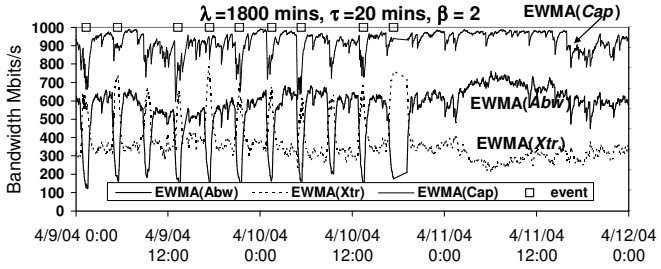

Figure 2. τ = 20 mins – nine events detected in EWMA(*Abw*) from SLAC to Caltech

β controls how far off the mean a data value must be to be entered into the trigger buffer. Reasonable values of β are in the interval 2 to 3. Lower values will detect more failures but may result in more false positives. As the sensitivity increases, the number of events observed often increases, as only values representing the more dramatic drops are put in the trigger buffer, and hence the threshold of change is reached more often.

The threshold δ is the normalized change between m_h and m_t (or between m_e and m_t, if m_e > 0). The optimum value for δ is determined by how large a change the user is interested in, and the

[6] These dips were identified to be caused by an application, running at regular intervals on the remote host, and not related to network conditions.

size of the normal periodic changes for the path. We typically use δ in the range 30-40%.

We typically set λ to about a day. The analysis time is proportional to λ, so one has to be careful with longer values.

2.4 Validation

We analyzed time series of ABwE bandwidth from SLAC to all 40 remote hosts for 100 days from March through June 2004. This was done using λ = 1800 minutes, τ = 180 minutes and δ = 0 (i.e. we detect all events that fill the trigger buffer). About 25% of the hosts manifested one or more events. We carefully looked at the events and time series, and categorized the events into three types:

1. Step down changes in bandwidth ("step");
2. Diurnal changes ("diurnal");
3. Changes caused by known events causing congestion (e.g. a regularly scheduled cron job, or network bandwidth tests) ("host").

To first order, a given host sees one type of event (for our data there was one exception, SDSC). Events for a given host typically have a small range of values of Δ (standard deviation(Δ) / mean (Δ) ~ 0.11± 0.1) typically indicating that backup routes are consistent or the diurnal behavior is consistent. This manifests itself in a multi-modal Distribution Function for Δ. By observing the Cumulative Distribution Function (CDF) we identify the values of Δ for various CDF percentiles. Table 1 shows, for each host with one or more events, the event type, the number of events below a given threshold δ, the CDF percentile and corresponding value of Δ together with the percentage of events detected or missed.

Table 1. Event probabilities

Host	Event Type	CDF Nb. Events	5%	10%	25%	50%	75%	90%	95%	Percentile Δ
BINP Novosibirsk	Step	2	2	2	2	2	0	0	0	
CESnet Prague	Step	1	1	1	1	1	0	0	0	
DL Liverpool	Step	2	2	1	0	0	0	0	0	
INFN Milan	Step	1	1	0	0	0	0	0	0	
Internet2 Atlanta	Step	2	2	2	2	2	2	2	2	
NIKHEF Amsterdam	Step	2	2	2	2	2	2	2	1	
ORNL Knoxville	Step	2	2	2	2	2	0	0	0	
SDSC San Diego	Step	2	2	2	0	0	0	0	0	
TRIUMF Vancouver	Step	1	1	1	1	1	1	1	1	
U Mich Ann Arbor	Step	1	1	1	1	1	1	1	1	
Total	Step	16	16	14	11	11	6	6	5	
Miss	Step		0%	13%	31%	31%	63%	63%	69%	
Caltech Pasadena	Diurnal	2	2	0	0	0	0	0	0	
Indiana Bloomington	Diurnal	2	2	2	0	0	0	0	0	
SDSC San Diego	Diurnal	4	4	4	0	0	0	0	0	
U Florida Gainesville	Diurnal	10	10	10	10	10	7	0	0	
Total	Diurnal	18	18	16	10	10	7	0	0	
Miss	Diurnal		0%	0%	0%	0%	0%	0%	0%	
ANL	Host	30	30	30	28	0	0	0	0	
Total	Host	30	30	30	28	0	0	0	0	
Miss	Host		0%	0%	7%	100%	100%	100%	100%	

Table 1 shows that though the modified "plateau" algorithm detects step changes with a range of values of Δ, it is unable to reject diurnal changes with a simple change in δ, without eliminating many valid events. Reference [8] suggests using a long λ of at least 1.5 days for diurnal changes and 3 days if the weekend/weekday behavior is different. Such large values of λ however, will take a long time to adjust to a change in the base bandwidth. Also the time to calculate the statistics increases with λ. For our data, extending λ to 3 days did not noticeably improve the elimination of false positives or reduce the number of missed events.

3. TRACEROUTE ANALYSIS

We currently utilize the standard Linux traceroute with a 2 second timeout, 1 probe per hop, a maximum hop count of 30 hops, and start the traceroute after leaving the SLAC border. For each destination host we study the performance of traceroute with UDP and ICMP probes and choose the most appropriate probe type (i.e. the one that resolves the route before the 30 hop maximum and/or minimizes the number of non-responding routers). By default we use UDP probes.

The goal of the traceroute analysis is to categorize the traceroute information and detect "significant route changes" between the current traceroute and one taken previously. The algorithm for categorizing the traceroutes is conceptually as follows. For each hop of a traceroute we compare the router IP address against the router IP address for the same hop for the previous traceroute measurement for a given path. If the router for this hop did "not respond" (i.e. the traceroute reported an asterisk (*)) for either this or the previous traceroute, then the Hop Change Information (HCI) for this hop is noted as "unknown". If the router responded (i.e. provided an IP address) for this hop, for both this and the previous traceroute, then the IP addresses reported for this hop are compared:

- If they are identical then the HCI is marked as "no change".
- If the addresses are not identical then:
  - if they only differ in the last octet then the HCI is marked as "minor change same first 3 octets".
  - if the addresses are in the same Autonomous System (AS) then the HCI is marked as "minor change same AS".
- If neither "minor change same first 3 octets" nor "minor change same AS" are identified then the HCI is marked as a "significant route change". We also sub-classify the "significant route change" into whether or not the change involves one ("minor significant route change") or more hops ("major significant route change").

When all the hops have been compared between the current and previous traceroutes, then precedence is given to any "significant route change", followed by "minor change same AS", "minor change same subnet", "unknown", and "no change" in that order.

In addition, unless the HCI is set to "no change" we also note whether the current traceroute did not terminate until the "30 hop" limit was reached and/or whether the destination is pingable. Since the destination hosts are

chosen to be normally pingable, a non-pingable destination usually means the destination host or site is not reachable, whereas a "30 hop" pingable destination is probably hidden behind a firewall that blocks traceroute probes or responses. In all cases except "significant route changes" we also note whether an ICMP checksum error was reported in a current traceroute.

Other cases that are noted include: the traceroute reporting "host unknown" which probably means the host name is currently not resolvable; whether the destination has multiple addresses. For routes with "no change" we also note whether any exclamation mark (!) annotations [10] occur in the traceroute; or whether a router interface "stuttered", i.e. a router interface reported for several TTL (Time To Live) settings. Annotations take precedence over stutters in the reporting. In some cases annotations and stutters are a persistent feature of particular routes, e.g. a router near IN2P3 in Lyon France that blocks access to the site network for UDP probes. The blocking causes a stutter since the router reports an ICMP destination (port) unreachable message on the first probe since the TTL has expired, and when trying to forward the next probe with a larger TTL, finds it is blocked with an ACL, cannot forward and reports ICMP destination (prohibited) unreachable message. In other cases (e.g. network unreachable) the annotations occur occasionally and typically mean the required network link is down and there is no alternative.

3.1 Traceroute Visualization

. The information is displayed in a table representing the routes for a single selected day (Figure 3). The default is today. The table columns represent the hour of the day and each row represents a remote destination host. The rows are labeled with anonymized host names and URL links are provided to: an HTML table of the day's routes, a text table of the routes, route number information (i.e. the route number, the associated route and the time last seen), the raw traceroute data, plots of the available bandwidth event analysis information and the ABwE dynamic bandwidth capacity, cross traffic and available bandwidth. The columns are labeled with the hour of the day.

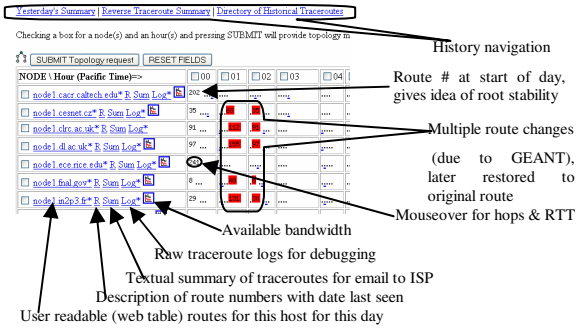

Figure 3. Section of a Traceroute Summary Table

For each non "significant route change" the cells of the table contain a single colored character for each traceroute measured in that hour. The single character represents the HCI or "30 hop" route categorization (i.e. period = "no change", asterisk = "not respond", colon = "minor change same first 3 octets", a = "minor change same AS", vertical bar = "30 hop", question mark = "host unknown", quotation make = "stutter") and the characters are colored orange if an ICMP checksum is noted, red if the destination host is not pingable, and black otherwise. The use of a single character to display the route categories allows a very dense table to be displayed that in turn facilitates visually scanning for correlated route changes occurring at particular times for multiple hosts and/or hosts that are experiencing multiple route changes in a day.

For each "significant route change" the route number is displayed colored red if changes were noted in more than one hop and orange otherwise.

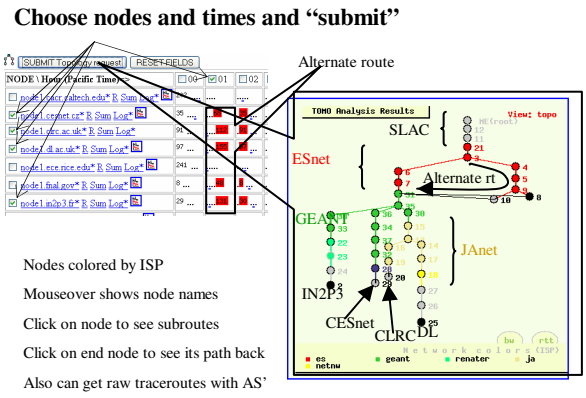

Figure 4. Traceroute display selected by check boxes on Traceroute Summary Table

If a bandwidth event was noted for a destination in an hour then that cell's background is colored to indicate the existence of the event.

Each row and each column also contains a check box (see Figure 4) that can be selected to submit requests for either a topology map for the selected hosts and times, or the routes together with their router AS information.

The web page also includes documentation and historical data.

4. BANDWIDTH AND ROUTE VISUALIZATION

To assist in developing and debugging the algorithm, and to provide a visual understanding to users, we display overlapping time series of *Cap*, the EWMA of *Abw* and *Xtr*, $\mathbf{m_h}$ and ($\mathbf{m_h}$ - β * $\mathbf{o_h}$), RTT, forward and reverse significant route changes (identified by a vertical bar) and events (identified by a large asterisk). Mouseover an event asterisk shows the time, Δ, ($\mathbf{m_h}$ - $\mathbf{m_t}$) / (sqrt($\mathbf{o_h}^2$ + $\mathbf{o_t}^2$)/2), $\mathbf{m_h}$, $\mathbf{o_h}$, $\mathbf{m_t}$, and $\mathbf{o_t}$. Clicking on the asterisk zooms the display in around the event.

Figure 5 shows the time series of the smoothed *Abw*, *Xtr* and *Cap* (as continuous lines), plus the raw *Cap* (as dots) and two

significant route changes correlating with a bandwidth change event (marked with a circle).

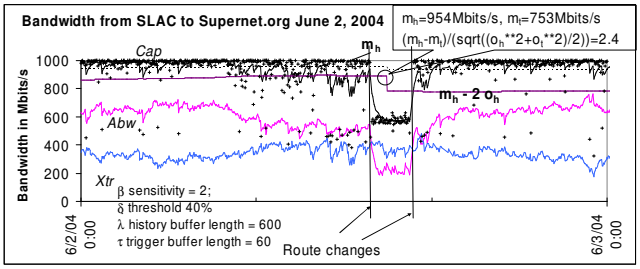

Figure 5: Bandwidth from SLAC to supernet.org June 2, 2004

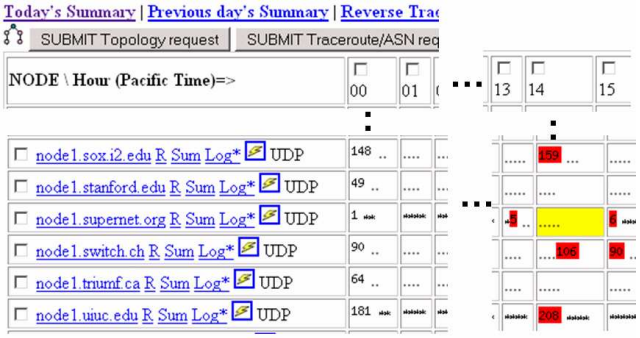

Figure 6: Part of the traceroute table for June 2, 2004.

Figure 6 shows part of the traceroute table associated with the bandwidth change event of Figure 5 for supernet.org for June 2, 2004. The route numbers at the significant route changes are given in red (black rectangles in black and white), and the box at 14:00 hours is colored yellow (shaded in black and white) to indicate that an event was detected. It can also be seen that other routes also experienced route changes at similar times.

5. FUTURE WORK

The major cause of false positives currently is due to diurnal changes, especially following a weekend when there is little congestion and hence stable bandwidth. We are currently working on incorporating Holt-Winters forecasting [7] to try and account for diurnal and other periodic effects. Areas for future exploration include further study and development of the bandwidth change algorithm (including using multivariate techniques [11] for example to look at *Cap* and *Abw* simultaneously, to add in RTT estimates and to look at multiple paths), and providing a better understanding of how to set the algorithm's parameters to meet required needs. As part of this we are gathering a "library" of interesting events both with true and false positives to benchmark the algorithms against. We also intend to apply the bandwidth change algorithm to other data including RTT estimates, and extend the tools to analyze AMP [12] data. Our intention is to use the events to initiate the gathering of further critical relevant data, and include that in the email alerts. We will also explore ways of filtering the alerts to reduce the number of emails.

Other work includes implementing the bandwidth change detection algorithm in a symmetric fashion to detect rises in throughput and thus quantify the time span of varying throughput levels.

6. ACKNOWLEDGMENTS

We gratefully acknowledge Jerrod Williams for work on the traceroute visualization and Ruchi Gupta for work on the plotting code.

7. REFERENCES

[1] *Experiences and Results from a New High Performance Network and Application Monitoring Toolkit*, R. L. Cottrell, C. Logg and I-Heng Mei, SLAC-PUB-9641, presented at PAM 2003.

[2] *ABwE: A practical Approach to Available Bandwidth Estimation*, J. Navratil and R. L. Cottrell, SLAC-PUB-9622, presented at PAM 2003.

[3] *Iperf – the TCP/UDP Bandwidth Measurement Tool*, NLANR. Available http://dast.nlanr.net/Projects/Iperf/

[4] *Bbftp – Large files transfer protocol,* IN2P3. Available http://doc.in2p3.fr/bbftp/

[5] *Case study of August 2003 bandwidth drop between SLAC and CALTECH*. Available http://www.slac.stanford.edu/grp/scs/net/case/caltech

[6] *Correlating Internet Performance Changes and Route Changes to Assist in Trouble-shooting from an End-user Perspective*, C. Logg, J. Navratil, R. L. Cottrell, SLAC-PUB-10341, PAM 2004

[7] *Introduction to Time Series and Forecasting*, P. Brockwell and R. Davis, Springer New York, 1996.

[8] *Automated Event Detection for Active Measurement Systems*, A. J. McGregor and H-W. Braun, Passive and Active Measurements 2001. http://byerley.cs.waikato.ac.nz/~tonym/papers/event.pdf

[9] *Programming Perl*, L. Wall, T. Christiansen, J. Orwant, O'Reilly & Associates Inc., 2002

[10] Unix man pages for traceroute, available for example at http://www.zytek.com/traceroute.man.htm

[11] *Diagnosing Network-Wide Traffic Anomalies*, A. Lakhina, M. Crovella, C.Diot, SIGCOMM 2004.

[12] *The NLANR NAI Network Analysis Infrastructure*, A. McGregor , H-W. Braun, J Brown. IEEE Communication Magazine special issue on network measurement, pp 122-128, May 2000.