How accurate are your network measurements?

Hiroshi TSUNODA
Tohoku Institute of Technology
35-1, Yagiyama Kasumi-cho, Taihaku-ku,
Sendai-shi, Miyagi, 982–8577 JAPAN
E-mail: tsuno@m.ieice.org

Glenn Mansfield KEENI
Cyber Solutions Inc.
ICR Bldg, 6-6-3, Minami Yoshinari, Aoba-ku,
Sendai-shi, Miyagi, 989–3204 JAPAN
E-mail:glenn@cysols.com

***Abstract*—Network measurement provides a useful insight into the dynamics of a network. Though network measurement is widely deployed and the results have significant implications in accounting, operations, security and quality of service management, the quality and nature of the statistics obtained have not been closely scrutinized. In this paper, we take a closer look at the measurement practices widely deployed and discuss the inaccuracies inherent in the measurement. We show that the inaccuracy essentially has its origins in the timestamp attribute of a measured value. Timestamp is a necessary attribute but its definition is imprecise. We discuss the issue of inconsistency that arises due to this imprecision.**

I. Introduction

Network measurement is an essential part of network monitoring and management. Traffic, status and various other parameters are routinely monitored and measured. The measurements constitute a snapshot of some aspect of the network. This snapshot provides the basis for management and control.

In this paper, we examine the accuracy and consistency issues in network measurement in real world situations, *i.e.,* in large distributed network environments illustrated in Figure 1. End users' networks are connected to a provider's access router via customer premises equipment (CPEs) (e.g., routers, switches, modems, set-top boxes, etc). For network management purposes, a manager is located on a network management station in the provider network, and agents are deployed on the CPEs.

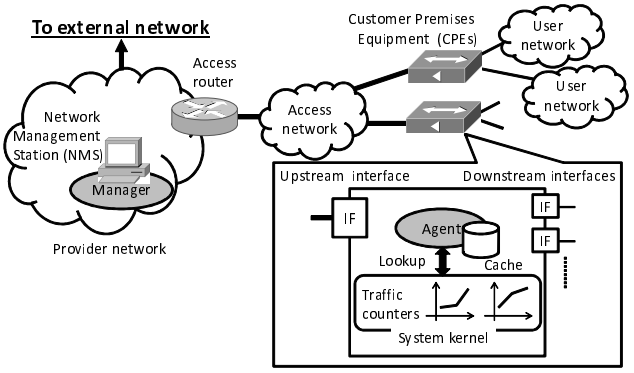

Fig. 1. Assumed environment

A manager sends a request for some information to an agent using some management protocol, such as the Internet standard Simple Network Management Protocol (SNMP) [1]. The agent samples the related components and sends back its response.

In the request-response mode of management, there are latencies in the manager, intermediate networks, and the agent. Latencies in the manager and the agent include the delay for processing the contents of request and response packets. Latency in intermediate networks includes transmission and propagation delays. In this work we focus on the cases where the latencies and its variation are significant. First we examine the inaccuracy in traffic measurement and then take a closer look at the issue of inconsistency of measured information.

The remainder of the paper is organized as follows. Related works are surveyed in Sec. II. The issue of inaccuracy in traffic measurement is explained in Sec.III. In Sec. IV, we examine the issue of consistency of the measured information. Section V discusses the new challenges, traffic measurement in mobile and cloud environments, followed by conclusions in Sec. VI.

II. Related work

Offline measurements use various information, such as traffic traces and management logs. An arbitrary level of detail, precision and accuracy may be achieved in offline measurements, by analyzing the stored information. In [2], the packet-wise statistics of IPTV traffic, such as packet size distribution and the frequency of retransmission and reordering, are computed and analyzed offline. As another example, deep packet inspection, a technology for analyzing packet payloads in real-time, has been used for the purpose of detecting and restricting the communication of P2P applications [3]. In these cases the main source of inaccuracy will be the failure in capturing, processing, and storing of information. For example, packets dropped by the traffic trace collector will cause a corresponding inaccuracy in traffic measurement.

Per-flow traffic measurement is an important way for getting accurate user-wise or application-wise statistics. It is useful for identifying *elephant flows* (flows that include a huge number of packets) and for usage-based billing. However, a problem with per-flow measurement is its lack of scalability [4]. Cisco NetFlow [5] uses a packet sampling technique for improving the scalability, but it introduces a significant measurement error [6]. The current trend is to implement measurement functions in high-speed but expensive SRAM [6].

Use of the Internet Protocol Detail Records (IPDR) [7] mechanism has been spreading in recent years, especially in

the area of cable internet. IPDR defines a data format and a protocol for exporting network measurement and management information [8]. Due to its flexibility, efficiency and scalability, IPDR is a promising technology for future network management and measurement.

The use of explicit time-tags is proposed in [9]. It requires that a managed object has a value and an explicit timestamp which indicates the time at which the value was observed. This technique is applied to high resolution traffic measurement [10] and network management in a mobile environment [11]. These solutions require additional instrumentation in network management agents and do not apply to most of the deployed CPEs which have limited storage and processing power. In this paper, we limit our scope to the typical situation where an agent, in response to a request, samples the corresponding managed objects and sends the results to the manager.

III. Inaccuracy in network traffic measurement

Two of the primary aspects of traffic measurement, namely, volume measurement and peak measurement, have significant implications in accounting, operations, security and quality of service management. For accounting management, the operator is required to have user-wise network bandwidth utilization figures as accurately as possible. Billing will be based on these figures. In cases where the tariff is not linear but a step function of the traffic volume, the inaccuracy, if any, gets amplified in the billed amount. In the operations and/or security management context, if some user is consuming an unfair share of the network bandwidth, the network administrator would want to be notified of it and would want to be able to identify the user. Moreover, for sophisticated quality of service management, an operator needs to know the accurate bandwidth usage in the *service level agreement* (SLA) context, and would want to detect and track SLA violations as quickly as possible [12].

In this section, we discuss the inaccuracy issues in traffic measurement that stem from the measurement procedure.

A. Basic procedure for network traffic measurement

An agent on a CPE maintains counters of the various facets of network traffic such as number of packets and/or number of octets, errors etc. These counters are cumulative in general.

A manager samples the counters periodically, computes the delta of the two samples and thereby computes the bandwidth utilization for the interval between two consecutive samples. Figure 2 illustrates how the bandwidth utilization is computed. v_t denotes the value of the cumulative traffic counter at the agent at time t. From two samples $v_{t_{i-1}}$ and v_{t_i}, the bandwidth utilization (Bw_i) between t_{i-1} and t_i is calculated as

$$Bw_i = \frac{\Delta v_i}{\Delta t_i} = \frac{v_{t_i} - v_{t_{i-1}}}{t_i - t_{i-1}}. \quad (1)$$

In periodic polling, $\Delta t_i \approx \Delta t$, is the polling interval.

In the ideal case, v_{t_i} is the value of the counter at time t_i, the timestamp of v_{t_i}. We will term this the data timestamp.

However, in the absence of explicit time-tags, a manager cannot know the exact value of t_i and uses some $\widehat{t_i}$ as an approximation of t_i.

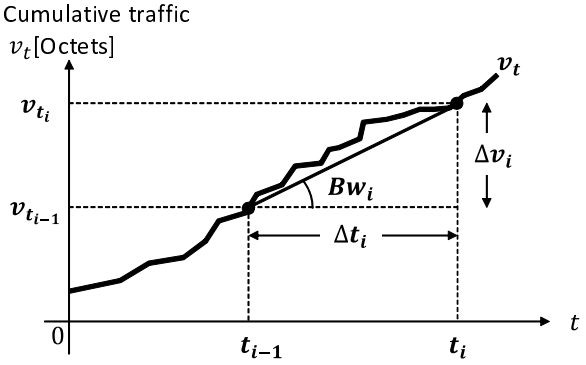

Fig. 2. Basic concept for calculating bandwidth utilization

Furthermore, depending on agent implementations, the counter value, $\widehat{v_{t_i}}$, returned by the agent is only an approximation of the traffic counter value v_{t_i} at time t_i.

Effectively, the bandwidth utilization is computed based on $\widehat{v}_{t_i}$ and $\widehat{t}_i$ as shown in Eq. 2.

$$\widehat{Bw_i} = \frac{\widehat{\Delta v_i}}{\widehat{\Delta t_i}} = \frac{\widehat{v}_{t_i} - \widehat{v}_{t_{i-1}}}{\widehat{t}_i - \widehat{t}_{i-1}} \quad (2)$$

The rest of this section discusses several factors contributing to errors in $\widehat{\Delta v_i}$ and $\widehat{\Delta t_i}$.

B. Inaccuracy of data timestamps

Figure 3 illustrates two successive polls - ($(i$-1)-th and i-th). According to this figure, v_{t_i} is the value at time t_i^A, the time at which the value is actually sampled. Thus, t_i^A is the timestamp of v_{t_i}.

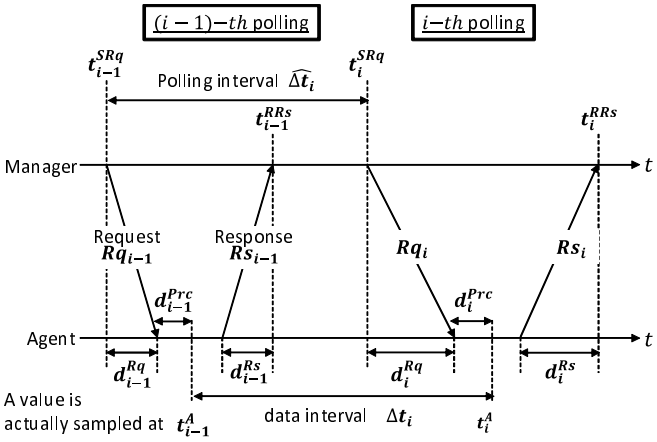

Fig. 3. Sequence diagram of a polling process

A manager, without any means of knowing the exact value of t_i^A may use t_i^{SRq} (t_i^{RRs}), the time when the manager sent the request to (received the response from) the agent, as an approximation. However, the request/response latency, (d_i^{Rq} and d_i^{Rs}), between the manager and the agent, and, the request processing time at the agent (d_i^{Prc}) are all variables depending

on the network conditions and processing load at the agent. Hence, even if the manager adjusts t_{i-1}^{SRq} and t_i^{SRq} so that the polling interval $\widehat{\Delta t_i} = t_i^{SRq} - t_{i-1}^{SRq}$ becomes constant, data interval $\Delta t_i = t_i^A - t_{i-1}^A$ will vary.

We conducted an experiment in a real network. A manager polled an agent on a Linux device D via the Internet. The average round trip time between manager and agent was 29.7 milliseconds. D was subjected to a constant bit rate (1Mbps) UDP stream. There was little background traffic.

We computed the bandwidth utilization using the measurement results of polling at five second intervals ($\Delta t = 5$ seconds). The measurement by a manager on D itself shows the actual bandwidth utilization is a straight line as shown in Fig. 4.

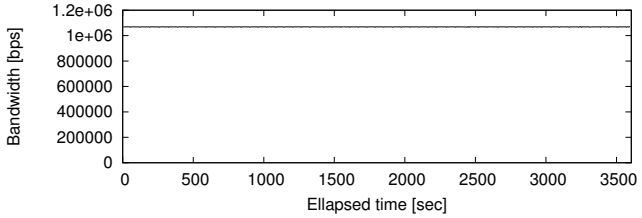

Fig. 4. Actual bandwidth utilization

We used the Managed Object *sysUpTime* [13] for estimating t_i^A. *sysUpTime* gives the time (in hundredths of a second) since the agent was last re-initialized. In the absence of explicit time-tags, the *sysUpTime* object fetched from the agent along with the traffic counter values is an estimate of t_i^A.

Figures 5 and 6 show the variations of polling interval and estimated data interval (from *sysUpTime*), respectively.

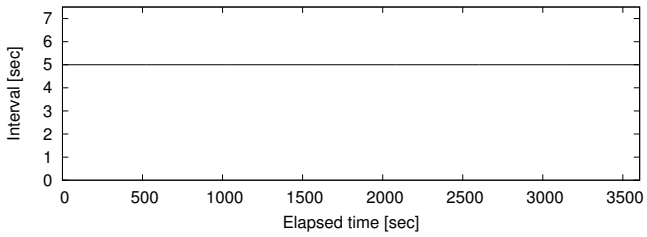

Fig. 5. Polling interval $\widehat{\Delta t_i}$

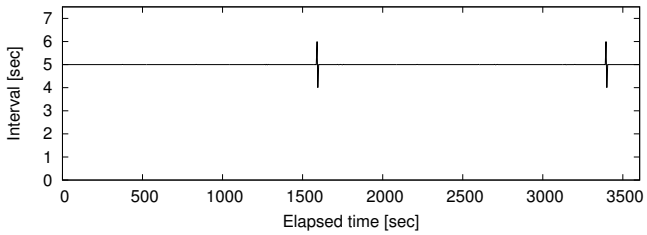

Fig. 6. Estimated data interval Δt_i

As shown in Fig. 5, $\widehat{\Delta t_i}$ is almost constant. But, Fig. 6 shows Δt_i has two spikes. These spikes are due to retransmission of requests. When the manager does not receive a response within a pre-defined time T_r, it resends the request. In this experiment, T_r is one second. As shown in Fig. 7, when a request R_i is retransmitted, actual data interval Δt_i increases by T_r followed by a decrease of same magnitude. In general retransmissions are carried out by the communications API, and cannot be detected by the management application.

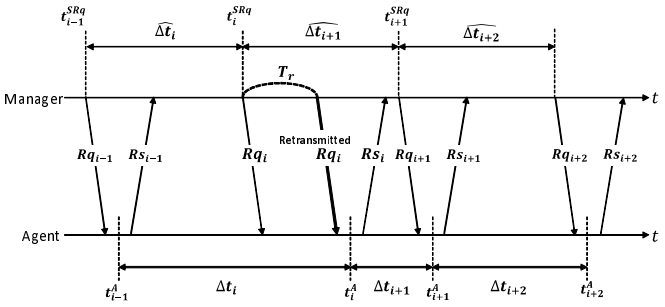

Fig. 7. Retransmission of a request

Figures 8 and 9 show bandwidth utilization computed using polling interval and estimated data interval, respectively. There are two spikes in the bandwidth utilization computed using polling interval though a steady value is expected (Fig. 4). The spikes are caused by the difference between polling interval and actual data interval. On the other hand, the estimated data interval using *sysUpTime* gives more accurate results (Fig. 9) which match well with the expected values shown in Fig. 4.

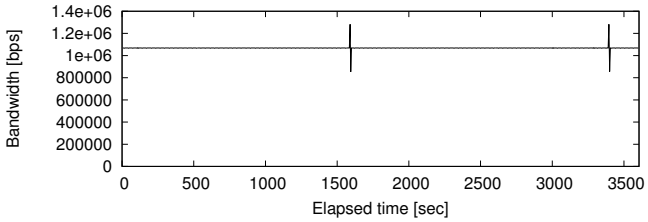

Fig. 8. Computed bandwidth utilization $\widehat{Bw_i}$: polling interval $\widehat{\Delta t_i}$ is used as data interval Δt_i

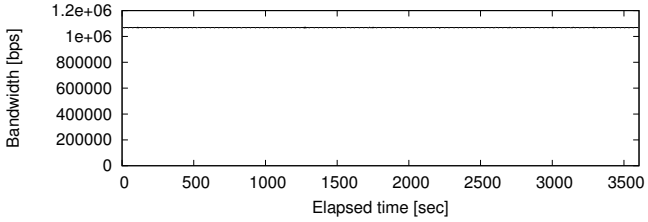

Fig. 9. Computed bandwidth utilization Bw_i: data interval Δt_i estimated using *sysUpTime*

In the next experiment, we will show the fluctuations of actual data interval that occur due to variations in inter-request arrival interval at the agent.

Figures 10 and 11 illustrate variations of polling and estimated data interval, respectively. In order to show the difference clearly, the y-axis range between 4.9 seconds to 5.1 seconds is magnified. It is clear that the estimated data interval has fluctuations while the polling interval is steady.

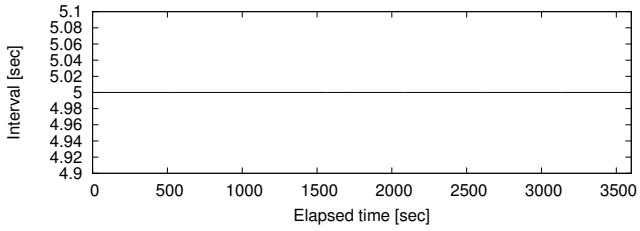

Fig. 10. Polling interval $\widehat{\Delta t_i}$

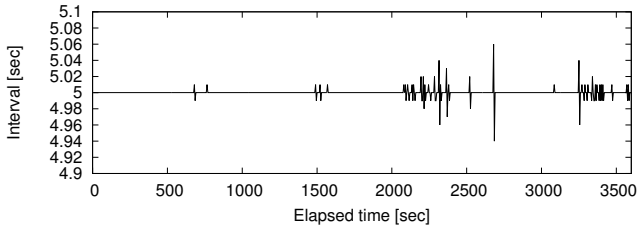

Fig. 11. Estimated data interval Δt_i

Figures 12 and 13 show computed bandwidth utilization $\widehat{Bw_i}$ and Bw_i, based on polling interval and estimated data interval, respectively. Although computed bandwidth utilization is a straight line in both figures, we find some differences when the y-axis range between 1.05Mbps to 1.1Mbps is magnified as shown in Figs. 14 and 15. The computed bandwidth utilization using polling interval in Fig. 14, shows several peaks that are not found in Fig. 15.

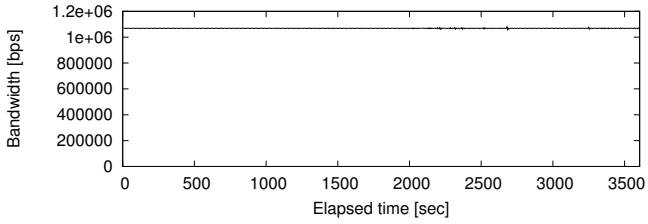

Fig. 12. Computed bandwidth utilization $\widehat{Bw_i}$: polling interval $\widehat{\Delta t_i}$ is used as data interval Δt_i

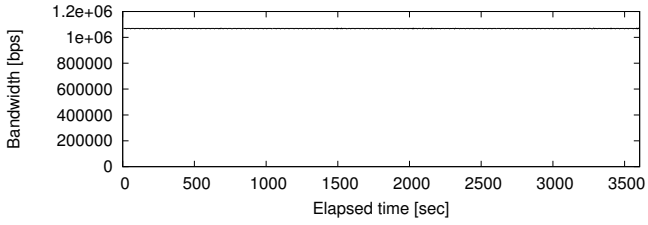

Fig. 13. Computed bandwidth utilization Bw_i: data interval Δt_i estimated using *sysUpTime*

Fig. 16 depicts the intervals at which request packets are seen at the manager. It is fairly steady. However, the inter-arrival times of those request packets at the agent show a significant variation in Fig. 17. This implies that variation in estimated data interval is probably due to the varying latency between manager and agent.

From the above results it is clear that the actual data interval

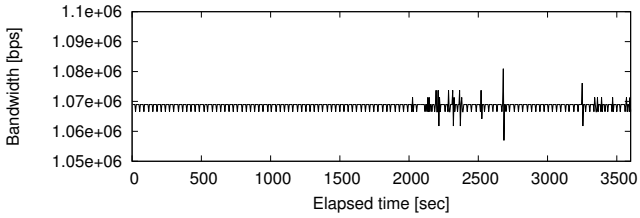

Fig. 14. Computed bandwidth utilization $\widehat{Bw_i}$: polling interval $\widehat{\Delta t_i}$ is used as data interval Δt_i

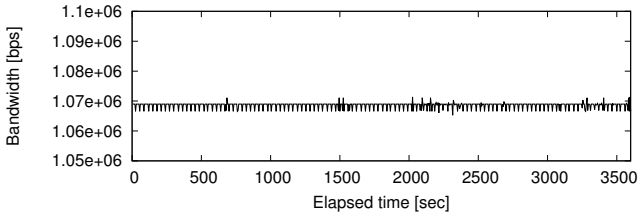

Fig. 15. Computed bandwidth utilization Bw_i: data interval Δt_i estimated using *sysUpTime*

varies due to several factors in the operational environment. The effect of the variation depends on the value of Δt. Table I summarizes the maximum error rate in $\widehat{\Delta t_i}$, Δt_i, $\widehat{Bw_i}$ and Bw_i, for various settings of Δt. It is clear that the computed bandwidth is less accurate when polling interval is used, and the maximum error rate increases as Δt decreases.

To compute bandwidth utilization accurately, the data interval must be accurately estimated. For accurately estimating each data interval it is necessary that every data value be accompanied by a timestamp.

Experimental results show that *sysUpTime* is a good candidate for a timestamp t_i^A.

Although *sysUpTime* is a close estimation of data timestamp, it only gives the time since the network management agent was last re-initialized. In the absence of some form of time synchronization between manager and agent, it is

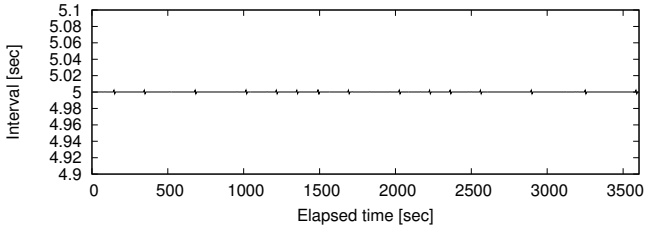

Fig. 16. Request sending interval at the manager

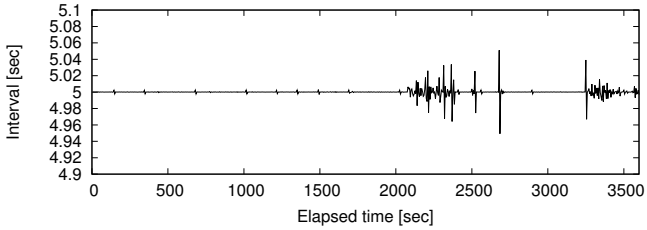

Fig. 17. Request arrival interval at the agent

TABLE I
MAXIMUM ERROR RATE

Δt	Variations in $\widehat{\Delta t_i}$	Variations in Δt_i	Error rates of $\widehat{Bw_i}$	Error rates of Bw_i
5sec.	0.01%	1.20%	1.15%	0.31%
10sec.	0.00%	0.20%	0.26%	0.17%
30sec.	0.00%	0.03%	0.04%	0.04%
60sec.	0.00%	0.02%	0.02%	0.02%
300sec.	0.00%	0.00%	0.01%	0.00%

not possible for the manager to know the exact time on the managers clock, at which data is actually sampled. In short, the manager can note the exact value of t_i^{SRq} on its own clock in Fig. 3 but cannot know the exact value of a counter on the agent at that time. On the other hand a manager can obtain the exact value of the counter at some time t_i^A but will not know the exact time on its own clock for that observation.

C. Non-realtimeness of the sampled value

An agent, when queried for the value of some object, fetches the value and returns it to the manager. The act of fetching may be a simple local memory lookup or a complex set of chained lookups some of which may not be local. In some implementations, to reduce the load due to such lookups, the fetched value is cached and reused for a small cache-lifetime. This causes an additional latency in the value of the object as seen by the manager. In the experiments detailed in the previous section, we have disabled the cache function on the agent in order to focus on the measurement errors due to the inaccuracy of data timestamps. When the cache function is enabled the counter value sampled and returned by the agent is not updated in real-time, but is updated in discrete time intervals. This means that v_{t_i} at t_i^A in Fig. 3 may be inaccurate.

Figure 18 illustrates the problem. For a constant traffic rate, the traffic v_t increases linearly, but the value returned by the agent, $\widehat{v}_t$, increases in steps. Let r be the constant traffic rate and $\tau_j (j = 1, 2, 3, 4, \cdots)$ be the times at which the counter is updated; $\widehat{v}_t$ can be

$$\widehat{v}_t = \begin{cases} v_{\tau_j} & (t = \tau_j) \\ v_{\tau_{j-1}} & (\tau_{j-1} \le t < \tau_j) \end{cases} \quad (3)$$

The update interval $\Delta\tau$ is cache-lifetime which is implementation dependent.

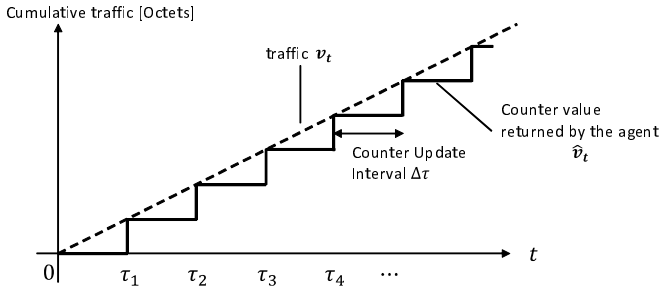

Fig. 18. Discrete nature of the counter information

Fig. 19 shows the value of a *ifInOctets* counter provided by a net-snmp [14] agent implementation. The device which the agent was running on was subjected to constant bit rate traffic, the cache-lifetime $\Delta\tau$ is set to 15 seconds. Implementations using cache-lifetimes are wide spread especially in devices where computing resources are scarce.

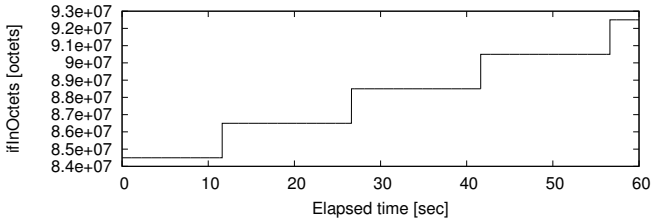

Fig. 19. Discrete nature of *ifInOctets* counter on a net-snmp agent

In the next experiment, we compute the bandwidth utilization using results of polling at five second intervals ($\Delta t = 5$ seconds). The target device was subjected to a constant bit rate (10Mbps) UDP stream. There was little background traffic.

Figure 20 shows the computed bandwidth utilization obtained from the net-snmp agent. The computed bandwidth utilization is not constant. This is because Δt is smaller than $\Delta\tau$. In this case, every third poll brings the updated value of traffic counter because $\Delta\tau$ is three times larger than Δt.

It is clear that Δt should be larger than $\Delta\tau$. However $\Delta\tau$ is implementation dependent and is usually not explicitly specified. There are instances where provisions are made for specifying $\Delta\tau$. For example, IP-MIB [15] provides a managed object *ipSystemStatsRefreshRate* which gives the intervals at which the counters in the corresponding table are updated. However, if such a managed object is not provided, network operators have no choice but to set Δt to a large value (e.g., five minutes in MRTG). But this setting impacts the granularity of the measurement.

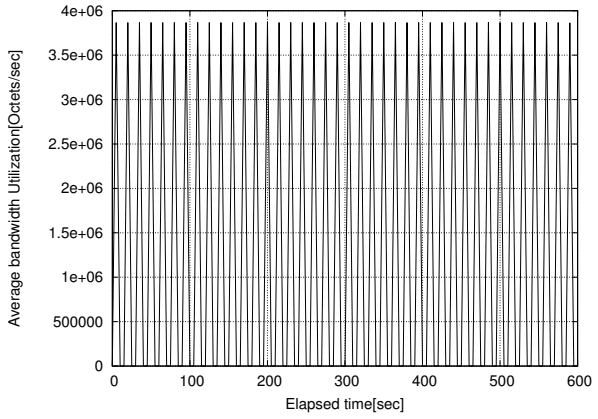

Fig. 20. Computed bandwidth utilization (Net-snmp) Δt = 5 sec

We conducted the same experiment with a Cisco Catalyst 3550 where $\Delta\tau$ is one second; smaller than Δt of five seconds.

Fig. 21 shows the experimental results. We can see several spikes in the computed bandwidth utilization.

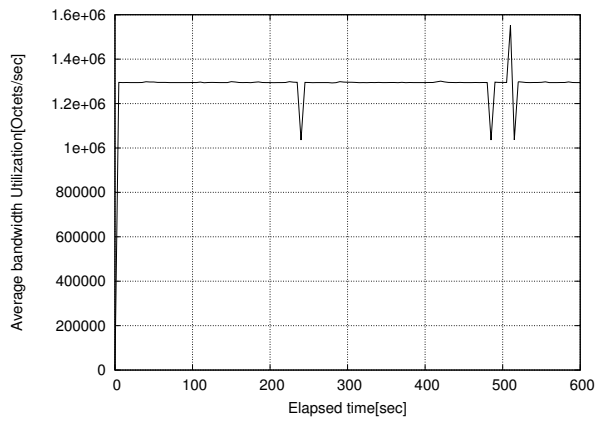

Fig. 21. Computed bandwidth utilization (Cisco Catalyst 3550) Δt = 5 sec

These spikes occur due to variation in the inter-request arrival interval. If the variation overlaps a cache-lifetime boundary there will be an upward spike followed by a downward spike. Fig.22 explains the reason. We assume that the request from the manager arrives during the target period (from τ_j to τ_{j+1}), and the lookup of the counter information is performed during this period. However, in the actual case, due to the variation of the one-way delay and the request processing time, the measurement may be performed before τ_j or after τ_{j+1}.

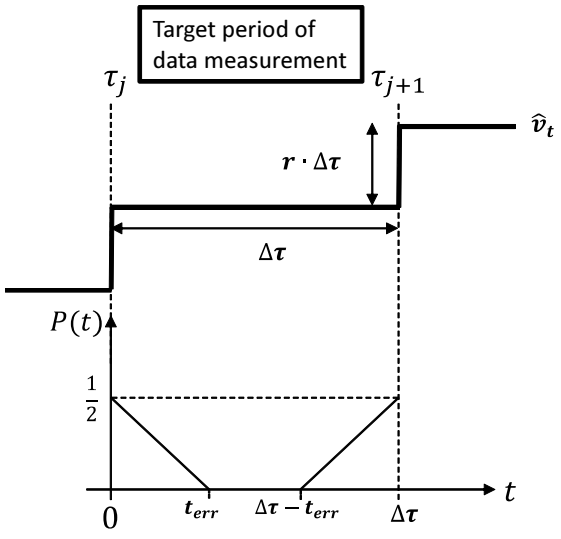

Fig. 22. Estimation of the error occurrence probability

In the lower half of Fig.22, P(t) gives the probability that the data measurement is performed before τ_j or after τ_{j+1} where t is the planned time of the data measurement. In this figure, we assume that the actual data measurement time is given by the uniform distribution from $t - t_{err}$ to $t + t_{err}$, for simplicity. In this case, $P(t)$ is given as follows.

$$P(t) = \begin{cases} -\frac{1}{2t_{err}}t + \frac{1}{2} & (0 \le t \le t_{err}) \\ 0 & (t_{err} < t < \Delta\tau - t_{err}) \\ \frac{1}{2t_{err}}t + \frac{1}{2} & (\Delta\tau - t_{err} \le t \le \Delta\tau) \end{cases} \quad (4)$$

If the planned measurement time t is randomly selected from 0 to $\Delta\tau$, the measurement error occurs with the probability given by the following equation.

$$\frac{2t_{err}}{\Delta\tau} \cdot \frac{t_{err}}{4} = \frac{{t_{err}}^2}{2\Delta\tau}. \quad (5)$$

When a measurement error occurs, the value sampled by the manager becomes $r \cdot \Delta\tau$ octets smaller or larger than the value that should truly be sampled by the manager. Then the estimated bandwidth utilization becomes $\frac{r \cdot \Delta\tau}{\Delta t}$ octets/second smaller or larger than the actual.

In this experiment (r = 10Mbps = 1.25MBps, $\Delta\tau$ = 1 second and Δt = 5 second), the theoretical value of measurement error is $250,000$ octets/second. This value is approximately equal to the amounts of increase or decrease that we can see in Fig.21.

According to the discussion in this subsection, a larger Δt leads to smaller error in the estimation of bandwidth utilization. However, the large value of Δt introduces another problem as discussed in the next subsection.

D. Counter discontinuity

The counters maintained by an agent have a finite capacity. These are generally 32 bit or 64 bit counters. A counter wraps when it exceeds the maximum value ($CounterMaxValue$). This causes a counter discontinuity. Counter discontinuities may occur due to other reasons too e.g. system re-initialization. Bandwidth utilization is calculated from the delta of two samples. For accurate traffic measurement, a manager must confirm that the counter is continuous between the two samples. There are two approaches to handling traffic counter discontinuities.

a. Ignore counter values between discontinuities.
b. If discontinuity is not due to reboot, assume that it is due to a counter overflow. e.g.

if $sysUpTime\,(t_i) \ge sysUpTime\,(t_{i-1})$
  if $\widehat{v}_{t_i} \le \widehat{v}_{t_{i-1}}$
    $\widehat{v}_{t_i} += CounterMaxValue$
  $\widehat{Bw_i} = \dfrac{\widehat{v}_{t_i} - \widehat{v}_{t_{i-1}}}{\widehat{t}_i - \widehat{t}_{i-1}}$
else
  $\widehat{Bw_i} = unknown.$

Here $\widehat{v}_{t_i} \le \widehat{v}_{t_{i-1}}$ is taken as a sign of counter continuity. This case leads to data loss. Since the agent is rebooted everytime the network device is restarted, this data loss can be controlled/caused by a user of a CPE. If the polling interval is reasonably large the user may fool the traffic

measurement system and associated accounting system to enjoy a free ride!

IV. Inconsistency among a set of measured information

All the factors described in the previous section are related to the imprecise definition of the timestamp attribute of measured values. In network management, it is important to monitor a set of objects. These objects together provide a snapshot image of some aspect of the managed domain. The lack of a precise definition of the timestamp attribute also affects the consistency of the data set that is expected to provide the desired snapshot image.

Say, there are N information components that need to be retrieved from an agent to generate some snapshot. In SNMP terms this would mean that N Managed Objects ($MO_k, k = 1 \cdots N$) need to be retrieved. Assume that the timestamp of MO_k is t_k. This is the time at which the source of MO_k was examined. This is the last time the value of MO_k was updated. For some time $t_{snapshot}$ a snapshot will be consistent if the following condition is satisfied.

$$t_1 = t_2 = \cdots = t_k = \cdots = t_N = t_{snapshot} \tag{6}$$

$t_{snapshot}$ is the timestamp of the snapshot. In other words, if for some MO_k, t_k is not equal to $t_{snapshot}$, MO_k may have been updated during the interval $t_k \sim t_{snapshot}$ and the snapshot may have changed. At the manager, it is difficult to actually retrieve and examine all t_ks to ensure that the condition in Eq. 6 is met. If an explicit $t_{snapshot}$ is maintained and made available at the agent, it is not necessary to retrieve and examine all the t_ks. To ensure consistency, it is sufficient if it can be confirmed that $t_{snapshot}$ has not changed during the retrieval of the constituent MOs.

The set of constituent MOs in a snapshot may be large. This introduces an additional source of inconsistency as the retrieval process spans some time period T_p. If a maximum of P MOs can be fetched in a packet, $T_p \approx (N/P) * RTT$, where RTT is the average round trip time between the manager and the agent. On the one hand in the absence of a snapshot timestamp $t_{snapshot}$ it is impossible to ensure the consistency of the obtained snapshot. On the other hand, as T_p becomes larger, the possibility that some MO_k has been updated, $t_{snapshot}$ has changed and consequently the retrieved snapshot is inconsistent, increases.

The routing table *ipRouteTable* in Management Information Base for Network Management of TCP/IP-based internets [16] is one example. By examining the routing table it may be possible to understand the network configuration and to debug some routing problems. Another example is where network maps are generated [17], [18] from the OSPF Link State Database *ospfLsdbTable* in the OSPFv2 MIB [19] and from the BGP4 Received Path Attribute Table *bgp4PathAttrTable* in the BGP4-MIB [20]. These tables are potentially large. It is not unusual for each table to contain thousands of entries. A finite time will elapse before the relevant objects are retrieved from the agent. If a table has been modified during this time, the observed values and corresponding snapshot is likely to be inconsistent.

An added complication arises when the set of MOs that need to be retrieved are on different agents. Information in MIB tables e.g. *dot1dTpFdbTable* in Bridge MIB [21] is used to detect the network devices and their point of connection: the switch and corresponding port, in a network [22]. This effectively provides an OSI Layer-2 map of the target network along with the connected devices and has significant implications in intranet security management and audit. The relevant MIB tables are of finite size and there may be a large number of switches in a network. Obtaining a consistent Layer-2 snapshot is a challenging task. Let R_x be a row in a conceptual table of a switch. R_x gives the list of MAC addresses seen on a physical port x. Assume that a terminal T_m is connected to the port x when the manager retrieves MOs for R_x. The MAC address of T_m appears in R_x. If T_m changes its connected port to port y after the manager has retrieved R_x and before R_y is retrieved, the manager will find the MAC address of T_m in R_y, too. In the map generated from the data terminal, T_m will appear in multiple ports (ports x and y). That will be an example of an inconsistent network map.

In order to confirm that a snapshot is consistent a $t_{snapshot}$ must be maintained and made available at the agent. An object like a *lastUpdatedTimeStamp* which indicates when a component was last updated, could serve the purpose. A cursory survey of the existing IETF standards track MIBs shows that more than 1200 tables are defined in the MIBs but less than 200 have provisions to indicate the last time the table was changed. This implies that it is not possible to confirm the consistency of the values of objects in most MIB tables and thus the snapshots are of limited utility.

V. Discussion

Network and device mobility introduce a new dimension to network measurements. Mobile entities may have poor/intermittent connectivity with the manager. One immediate consequence of this is wide fluctuations in the latency, which lead to added inaccuracy in the estimation of data timestamp. Further, when the (wireless) link between manager and agent is re-established, since the mobile node and/or network may have been operational through the period of disconnection, some sensitive applications may require the manager to re-start the data retrieval from the point of (temporary) disconnection. In such cases *sysUpTime* cannot be used as the implicit data timestamp. An explicit time-tag is necessary.

In cloud environments where dynamic provisioning is the keyword, the measurement parameters themselves will change with time. In such cases it is imperative to have appropriate meters set up within the cloud which will react quickly enough and provide a simple interface to the measurement manager.

VI. Conclusion

In this work, we have examined the issues of inaccuracy and inconsistency in network measurement.

For accurate traffic measurement a manager must have three information components: the sampled value, an accurate timestamp of the value, and a flag indicating the time at which the counter of the value experienced a discontinuity. Further, to ensure that a snapshot is consistent, a manager must have access to some object that will indicate the last time any of the objects constituting the snapshot was updated.

In the SNMP context, *sysUpTime* may be used as an implicit timestamp in some cases. Auxiliary objects which indicate the last time when one of the corresponding counters experienced a discontinuity e.g. *ifCounterDiscontinuityTime* in IF-MIB[23], and when a component of a snapshot was last updated e.g. *hrSWInstalledLastUpdateTime* in HOST-RESOURCES-MIB[24], do exist. The provision of the *non repeaters* parameter in the SNMP getbulk request can be put to good use to retrieve these auxiliary objects, efficiently. Some thought on the issue of accuracy and consistency has gone into the design SNMP and the corresponding MIBs.

Yet a quick survey shows that several of the widely used network traffic monitoring tools, e.g. MRTG [25], RRDTool [26], use the error-prone polling interval instead of *sysUpTime* or any other form of time-tag for calculating the data interval. Not many SNMP MIB tables provide means for confirming continuity of counters and consistency of tables. There is considerable scope for improvement of accuracy and consistency in present measurement practices.

REFERENCES

[1] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet-Standard Management Framework. RFC3410, December 2002.

[2] Y.J. Won, Mi-Jung Choi, Byung-Chul Park, J.W. Hong, Hee-Won Lee, Chan-Kyu Hwang, and Jae-Hyoung Yoo. End-User IPTV Traffic Measurement of Residential Broadband Access Networks. In *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, pages 95 –100, April 2008.

[3] Milton Mueller and Hadi Asgharia. Deep Packet Inspection and Bandwidth Management: Battles Over Bittorrent in Canada and the United States. *Telecommunications Policy*, 36, 2012.

[4] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, August 2003.

[5] Cisco Systems, Inc. Introduction to Cisco IOS NetFlow - A Technical Overview, May 2012. http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html.

[6] Tao Li, Shigang Chen, and Yibei Ling. Fast and compact per-flow traffic measurement through randomized counter sharing. In *INFOCOM, 2011 Proceedings IEEE*, pages 1799 –1807, 2011.

[7] TM Forum. Standardized Interfaces IPDR, 2013. http://www.tmforum.org/IPDR/4501/home.html.

[8] Wenxuan Guo, A. Dunstan, and J. Finkelstein. Using IPDR to transfer network management and measurement information. In *Communications Quality and Reliability (CQR), 2011 IEEE International Workshop Technical Committee on*, pages 1 –6, may 2011.

[9] G. Keeni. The Managed Object Aggregation MIB. RFC4498, May 2006. http://www.ietf.org/rfc/rfc4498.txt.

[10] Glenn Mansfield, Sandeep Karakala, Takeo Saitoh, and Norio Shiratori. High Resolution Traffic Measurement. In *Proc. of A workshop on Passive and Active Measurements on the Internet(PAM2001)*, pages 67–73, 2001.

[11] Glenn Mansfield Keeni, Kazuhide Koide, Takeo Saitoh, and Norio Shiratori. A Bulk-retrieval Technique for Effective Remote Monitoring in a Mobile Environment. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 01*, AINA '06, pages 889–894, 2006.

[12] Takahiro Murooka, Masashi Hashimoto, and Toshiaki Miyazaki. A High Time-Resolution Traffic Monitoring System. *IEICE transactions on information and systems*, 87(12):2618–2626, December 2004.

[13] R. Presuhn. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). RFC3418, December 2002.

[14] Net-SNMP. Net-SNMP, 2013. http://www.net-snmp.org/.

[15] S. Routhier. Management Information Base for the Internet Protocol (IP). RFC4293, 2006.

[16] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. RFC1213, 1991.

[17] G. Mansfield, K. Jayanthi, T. Ika, K. Ohta, and Y. Nemoto. Network cartographer. In *Proceedings of the fourth ACM international conference on Multimedia*, pages 439–440, 1996.

[18] G. Mansfield, M. Ouchi, K. Jayanthi, Y. Kimura, K. Ohta, and Y. Nemoto. Techniques for automated network map generation using SNMP. In *In the proceeidings of INFOCOM '96*, volume 2, pages 473–480, March 1996.

[19] D. Joyal, P. Galecki, S. Giacalone, R. Coltun, and F. Baker. OSPF Version 2 Management Information Base. RFC4750, December 2006. http://www.ietf.org/rfc/rfc4750.txt.

[20] J. Haas and S. Hares. Definitions of Managed Objects for BGP-4. RFC4273, January 2006. http://www.ietf.org/rfc/rfc4273.txt.

[21] K. Norseth and E. Bell. Definitions of Managed Objects for Bridges. RFC4188, 2005.

[22] Glenn Mansfield, Cyber Solutions. Information Security Business in Japan. In *Security Solutions in Asia and Germany, ICT-Conference*, September 2007. http://www.cysols.com/research/papers/SecurityBusinessInJAPAN.pdf.

[23] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC2863, 2000.

[24] S. Waldbusser and P. Grillo. Host Resources MIB. RFC2790, 2000.

[25] Tobi Oetiker. Tobi Oetiker's MRTG - The Multi Router Traffic Grapher, February 2011. http://oss.oetiker.ch/mrtg/.

[26] RRDtool - About RRDtool. http://oss.oetiker.ch/rrdtool/.