Constantinos Dovrolis, Mark Allman, k claffy
CAIDA
Oct. 2001
Request for Comments: CAIDA-E2E-001
Obsoletes: none
Category: DRAFT

End-to-end Throughput-related Performance Metrics

1 Goals of this draft

1. Define several throughput-related end-to-end performance metrics
2. Identify uses/scope for each of these metrics
3. Attempt to establish a consistent terminology

Summarize what is known about the measurement of these metrics (existing methodologies and tools), and comment on what is still an open issue in this research area.

1.1 Background

We start with two hosts: S (source) and R (sink). S sends packets to R through a packet network infrastructure P(S,R). We are interested in measuring several throughput-related performance metrics concerning P(S,R).

Throughput-related metrics: in one way or another, these metrics measure a **rate** of information transfer. They are all measured in bytes/bits per second. We are interested in IP-layer throughput. Any framing/header overhead at sub-IP layers takes away from data throughput.[1]

Major assumption: all throughput-related metrics are fundamentally dependent on the underlying network path that the measurement flow follows. If this network path is not well-defined, in the sense that routing changes or multipath-forwarding take place while the measurements are in progress, then throughput mesurements may not be stable.

In the rest of this draft, we consider the case that the network infrastructure P(S,R) is a **constant sequence of store-and-forward network links** for the duration of the measurement process. Specifically, we denote te Hhop network path from S to R as

```
P(S,R)={L_0, L_1, L_2, .... L_H}.
```

However, we will also consider known ”technological deviations” from this model, such as multichannel links, or internal router ”links” that introduce variable delays.

[1] Obviously, this type of metric can only be defined in terms of a ‘measurement flow’ (sequence of packets), rather than on a per-packet basis. This is different than other metrics (delay, loss event) that provide a measurement for each packet sent.

2 Metric-1: Capacity C

Intuitively, the **capacity** of the path P(S,R) is the maximum possible throughput that the path can provide to a flow, given its "structural" technological constraints. By structural constraints, we refer to the characteristics of the underlying network links and forwarding elements. These characteristics are normally constant (wireless links are one of several exceptions though). An important point is that the capacity of a path does not depend on how heavily loaded the path is with cross traffic.

More formally, we can define the capacity of P(S,R) as follows. Each link L_i of the path can forward packets with a maximum possible throughput C_i (some bits per second). C_i is referred to as the "transmission rate" or "link capacity" of L_i. C_i is determined by the physical or technological limitations of the underlying transmission link, or by the processing limitations of the associated forwarding element. The capacity C of the path P(S,R) can be then defined as:

```
C = min_{i=0 to H} C_i
```

The link that determines the capacity of the path is referred to as the **narrow link** of the path. If the path includes several links with the same capacity C, we can arbitrarily define the narrow link to be the last of these links in the path.

There are some technological issues that can complicate the definition of the path capacity:

1. Forwarding elements sometimes have different capacity values for different packet sizes. Usually this is due to processing power limitations resulting in smaller packets leading to lower throughput. Since the capacity of a path is associated with its maximum possible throughput, we must define it for path-MTU packets.

2. Some forwarding elements include shapers (e.g., leaky buckets), allowing a larger transmission rate for only a certain burst size. We should probably define the capacity of such elements based on their peak rate (rather than their sustained rate). How common are shapers today?

3. Some links have variable tranmission rates that depend on the physical characteristics of the underlying physical link. The most common example of this occurs with certain wireless links *(more specifically?)*.

4. Some links are based on "medium sharing" (e.g., CSMA Ethernet, cable modems). For these links, the capacity must be defined for the case that there is only one source in the link.

5. Some routers use a form of hashing to map flows to different sub-links (for load balancing?). Suppose that such a link consists of K sub-links with capacity c. Even though the capacity of the link C=K*c, an individual flow will never become more than c.

2.1 Applications/uses of the capacity metric:

1. A diagnostic/debugging tool for network managers: Operators may want to verify that the path provides the maximum throughput that they expect. For instance, network operators

may want to experimentally check whether a path can provide the throughput of its slowest link, or whether it is limited by the intermediate routers, or whether an Ethernet interface is configured as 10 or 100Mbps.

2. An SLA verification tool for network users/providers: A user may want to periodically check whether his provider truly offers the agreed-upon capacity, either only at the access link between the user and the provider, or on certain paths which are part of the SLA.

3. An upper bound on throughput: Applications and transport protocols may wish to specify the maximum throughput for a particular path, if it is not used by other flows. Applications should not regulate their transmission rate based on the measured capacity though; instead they should adopt standard congestion control practices (see related RFCs).

Ideally, applications should not attempt to get higher throughput than the capacity of the path, assuming that they reliably know C, because then they only create persistent queueing or losses within the network. This is not consistent with TCP's behavior, though, as TCP does not assume that C is known. Even if it was known, it would probably make sense to add a constraint on the maximum congestion window, of the form:

cwnd ¡= C / RTT

where RTT is the minimum RTT in the path (without queueing delays).

2.2 Measurement approaches:

Blasting the path with a sequence of back-to-back UDP packets: This approach is not guaranteed to work in the presence of cross traffic (besides the problems of 'intrusiveness' that it causes), because the measurement flow will get 'dispersed' as it is multiplexed in the network queues with packets from cross traffic. Consequently, the measured throughput will underestimate the path capacity.

Packet-Pair Dispersion (PPD) methods: This measurement technique is based on the dispersion of two back-to-back packets. *(see literature.. should we review the literature here??)* It is not easy to get a correct estimate with this approach, especially when the path is heavily loaded. This is an interesting research area, and there are already some tools available that use this approach (bprobe, nttimer, pathrate).

Variable Packet Size (VPS) methods: This technique attempts to measure the variations of the RTTs in the path as the size of the probing packets varies. From those variations, estimates can be made for the capacity of **each link in the path**, and thus of the end-to-end path. Tools based on VPS are: pathchar, pipechar, pchar, clink, and nettimer. Unfortunately, their results are often wrong, probably because of the presence of "hidden" layer-2 forwarding elements introducing additional transmission delays into the path. These additional delays cannot be accounted for by these tools, and thus the final capacity output is underestimated.

others? tbd

3 Metric-2: Available bandwidth A

The available bandwidth of a link L_i in a time interval T is the part of the link's capacity that is idle (unused) during T. If u_i(T) is the utilization of L_i in T, we say that the available bandwidth A_i(T) in the same time interval T is:

```
A_i(T) = C_i [1-u_i(T)]
```

Similarly, the available bandwidth of a path P(S,R) is defined based on the link with the minimum available bandwidth along P(S,R):

```
A(T) = min_{i=1 to H} { C_i [1-u_i(T)] }
```

The link with the minimum available bandwidth is referred to as the **tight link** in the path. Several important points about this definition must be clarified:

- The available bandwidth is a dynamically varied measure. Therefore, it is crucial to specify the time interval T in which it is measured. Most people are familiar with MRTG/RRD graphs of used/available bandwidth, generated from SNMP data. In those graphs, the length of each measurement interval T is 5mins (default value). We refer to the length of T as the **measurement timescale** of A.

- As the measurement timescale decreases, the variance of the available bandwidth increases. The exact shape of this relation (measurement variance vs timescale) depends on the statistical characteristics of the network traffic. In general, the more bursty the traffic is, the higher the variance of the available bandwidth will be.

- Suppose that we measure A in a time interval T1=(t, t+tau) to be A1. This measurement will be useful if it can be used as a predictor for the available bandwidth in a future time interval, say T2=(+tau, t+2*tau). Such a prediction will not be possible if the underlying network traffic is not stationary in (t, t+2*tau). Consequently, the measurement timescale tau should also be chosen based on the stationarity characteristics of the traffic. For insance, if we know that the average utilization in a path shows significant variations in time intervals that are longer than 5 minutes, it would not make sense to use a measuring timescale that is longer than 1-2 minutes.

- The available bandwidth **should not** be interpreted as the throughput that a new flow can get in the path. We know from basic queueing theory results that a saturated link with a fixed number of buffers will introduce losses and significant queueing delays. Even if a flow can accurately measure A, it may not be desirable for the flow to set its transmission rate to A, because that would introduce losses and increased delays in the tight link.

- Clearly, a congestion unresponsive flow can get a higher throughput than A, if it does not react to losses or queueing delays. Similarly, a TCP connection may get a throughput that is more or less than A (see next section).

3.1 Application/uses of A:

The available bandwidth metric is probably of more importance to network operators and managers for monitoring the absolute load in network paths. This is particularly important for paths that are not in the network manager's administrative control (e.g., those paths that are provided by another ISP). In those cases, the network manager may not have access to the SNMP utilization data of the path routers.

If there is a simple and reliable way to estimate A, it may also be of some use to applications and transport protocols. Specifically, a transport protocol such as TCP can use A to set the initial `ssthresh` variable. Or, a streaming application can use a certain fraction of A as the initial streaming rate, provided that the application adopts standard congestion control techniques for reacting to packet losses or ECN notifications.

The only known way to measure the available bandwidth of a link or path is by using SNMP data directly from the corresponding routers. This approach has proven to be very useful in practice, as many network managers monitor the load of their links with SNMP-based tools such as MRTG/RRD. These tools cannot be used, however, if we don't have SNMP access to the path routers. Additionally, many network managers today do not publicize such load graphs because they consider them confidential information regarding the status of their network. Finally, such tools are only used on a per-link basis, and so they cannot provide information for an end-to-end path.

Currently, there are no end-to-end measurement tools that can estimate the available bandwidth of a path. Tools such as `cprobe` or `nettest` attempt to measure A from the end-to-end dispersion of long packet trains. It has recently been shown however [1] that this methodology measures a load-dependent characteristic of the path (called Asymptotic Dispersion Rate), that is **not** in any way related to the available bandwidth. It remains an interesting open research problem to estimate the available path bandwidth through end-to-end probing.[2]

4 Metric-3: Bulk-Transfer Capacity (BTC)

The BTC metric, currently standardized by the IPPM WG, focuses on the throughput of a peristent (long) TCP connection, when the connection's throughput is only limited by the network infrastructure P(S,R). More explicitly, a TCP file transfer measures the BTC of P(S,R) when the following conditions are met:

- The transfer's throughput is not limited by the receiver's advertised window (i.e., the value of that window is always larger than the sender's congestion window)
- The number of segments transferred is much larger than the connection's bandwidth-delay product. This constraint stems from our need to measure the connection's throughput after the initial slow-start phase, while the connection is in a congestion avoidance phase.
- The TCP implementation at both the sender and he receiver follows the descriptions of *RFCs XXXX*.

[2] *** Mark: we are developing a new technique/tool here that may provide some good results for measuring A. I can talk to you more about it if you want.

- Question: What if the connection experiences retransmission timeouts? Do we consider such a connection's throughput as a valid BTC measurement?

The BTC metric is obviously dynamically varied, depending on the amount of cross traffic in the path. Consequently, we have to deal with the same hard questions of "measuring timescales", discussed in the previous section. The "measuring timescale" T for BTC is determined from the size of the transferred file S and the achieved average TCP throughput R (R=S/T). [3]

4.1 Applications/Uses of BTC:

- The BTC metric provides useful information about the path P(S,R) to a user wanting to transfer large files from S to R using TCP. Given that TCP is the major transport protocol in use, the BTC metric is of primary importance to users.

- The BTC metric can be used by non-TCP applications (say streaming apps) wanting to measure the throughput that they would receive in the path if they were using TCP's congestion control algorithms.

- The BTC metric can also be used for network monitoring and debugging, or SLA verification, if it is known that there is no cross traffic in the path. If there is cross-traffic, then the measured BTC will underestimate the path's capacity.

An important difference is that the BTC is not limited by only one link in the path, as is the case for the capacity and available bandwidth. Instead, the BTC is affected by all links in the path. The reason for this is that the throughput of a TCP connection in the congestion avoidance phase is determined by the round-trip time (RTT) that the connection experiences (including queueing delays in any link along the forward or reverse path), and the end-to-end loss rate (mainly in the forward path). Consequently, any link that introdces queueing or losses can affect the BTC measurement. In some cases though, there is a certain link in the path causing most of the queueing delays and almost all losses. In those cases, we can talk about the **BTC-bottleneck** of the connection. The BTC-bottleneck, when it exists, may or may not be the same link as the tight link. For example, a severely under-buffered link that causes frequent losses may be the BTC-bottleneck, even though it may have a higher available bandwidth than other links in he path. In the general case however, the BTC is determined by several links in the path.

A second important consideration is whether the BTC metric is equivalent to the available bandwidth A. The conventional wisdom is that a persistent TCP connection capture the "available bandwidth" in the path, and so one would expect that BTC=A. If we adopt the definition of available bandwidth given earlier in this draft, this is clearly not the case. Some of the reasons are given next:

- Even if A=0 (a totally saturated tight link, and thus path), a TCP connection will get some throughput (and thus BTC¿0), because **TCP shares the capacity of a path with other TCP connections**. In other words, the BTC measurement-flow will interact with the rest of the

[3] *** Mark, I think that it is probably a better idea to fix T, rather than S, while performing BTC measurements. In that way, we can compare the stability of the measurements when they capture time winows of equal length ***

TCP connections in the path, sharing the path's capacity with them in a certain way. To measure the available bandwidth, however, it is important that our measurement traffic does not affect the cross traffic at all, keeping the utilization level in the tight link at the same point that it was before the measurements.

- The BTC may be less than A, especially if some links in the path are under-buffered. In that case, the BTC measurement-flow will be experiencing losses while repeating its sawtooth-pattern, before it accumulates a sufficient number of backlogged packets in the router queues. For a single connection, it is straightforward to see that the tight link needs to have at least a bandwidth-delay product of buffering space, in order for a TCP connection to saturate it. For more general conditions (many connections with different characteristics) it is not entirely clear how much buffering space we need so that a BTC flow can always saturate the path.

- The BTC may be more than A, in the sense that the BTC measurement flow can get more than the available bandwidth in the path, reducing the throughput of the cross traffic in the path. This happens especially in over-buffered paths. The reason for this is that the BTC flow will increase the average queue size in the tight link, and thus it will increase the average RTT and loss rate for all other connections that pass through that link. This effect will cause the throughput of those connections to drop, causing th BTC to be higher than what was available before the BTC measurement started.

From the previous three points, it is clear that the BTC is not only a function of the capacity and utilization in each link of the path, but also a function of the amount of buffering space, and of the nature of the cross traffic in the path (UDP or mice flows would interact with the BTC flow in different ways than elephant flows).

4.2 Measurement methodologies and tools:

The BTC can be measured with tools that transfer a large fle between two hosts. Such tools are ttcp, iperf, treno, cap (others?). Some of these tools actually use TCP to transfer a large file, while others (cap, treno) **emulate** some elements of TCP's congestion control using UDP. Also, some of these tools require access at both ends of the path (cap, ttcp), while others (treno) require access only at the receiver R.

Review tools here and describe differences.

An important point about the BTC metric, is that its measurement is significantly simpler and more robust than the measurement of the capacity and available bandwidth metrics. In other words, every TCP transfer (given the constraints at the start of this section) is a valid estimate of the BTC metric *** *is this true Mark? Do you reject measurements if timeouts occurred in the transfer?* ***. On the contrary, measurements of the the capacity or the available bandwidth may simply be **wrong**, when they are compared to the actual capacity or available bandwidth in the path.

References

[1] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Infocom*, (Alaska), 2001.