Estimating Available Capacity of a Network Connection *

Suman Banerjee
Department of Computer Science
University of Maryland at College Park
suman@cs.umd.edu

Abstract

Capacity estimations for an end-to-end connection across the is a measure that can be useful to many applications. Its applicability encompasses QoS guarantees, congestion control and other related areas. In this paper, the available capacity of a connection, is defined and estimated through measurements made only at the endpoints. The measurements detects and accounts for variability of cross traffic that pass through this connection. Modifications to current techniques to measure packet service time of the 'bottleneck' router is also described. Finally, estimation results on wide-area network connections from recent experiments to multiple sites is presented.

1 Introduction

In this paper techniques to estimate *available capacity* of an end-to-end connection, are presented. Available capacity at time, t, is defined to indicate the amount of data that could be inserted into a network path at a given time, so that the transit delay of these data packets would be bounded by a maximum permissible delay, Δ. This measure is closely related, but not exactly identical, to the bandwidth measure, studied by some previous researchers [Bo 93], [CaCr 96b] and [Ja 97]. To estimate the available capacity of a connection, measurements need to be made at the endpoints only.

The available capacity measure is interesting to various applications in different ways. First, the knowledge of available capacity can be beneficial to real-time applications that require estimates of network carrying capacities. Since, the capacity measures are with respect to the transit

*Scholarly paper submitted to the Faculty of the Graduate School of the University of Maryland at College Park in partial fulfillment of the requirements for the degree of Master of Science, 1999.

delay of packets, such applications might decide to forgo sending packets during periods of low available capacity, that might be below its minimum requirements from the network. Second, for uncontrolled non real-time data sources, this estimate would provide a feedback about the network congestion, which, might be used to control its traffic generation rates. The available capacity estimation technique can be refined to serve as either controlling or policing mechanisms for all uncontrolled data sources. Another interesting application of available capacity measure has been previously discussed in [CaCr 96b], where web clients choose an appropriate web proxy or server depending on the bandwidth.

In section 2 certain issues on Internet measurements are discussed. In section 3, the network model is analyzed and available capacity is quantified. In section 4, the measurement tool NetDyn [Sa 91] is described. Mechanisms to evaluate the service time of a 'bottleneck' router in a connection, is provided in section 5. In section 6, the results from some of the recent experiments are estimated and vaildated. Finally, in section 7, this work is summarized and some future directions of research are outlined.

2 Issues on Measurements

Taking measurements in the Internet is a difficult problem to approach, particularly, because of the large number of factors that are unknown and uncontrollable, from an end-user perspective. Without exact knowledge of router design, amounts of traffic from other sources and how they vary, when and how routes change, it becomes difficult predict network behavior. The presented technique for Internet measurements has been based on establishing a model that captures most of the interactions, and refining the model through experiments to explain various phenomena.

An important design decision in the measurement techniques was to ensure that no extra facilities are needed from the network itself. This lends to ease of deployment. Through the work reported in this paper, the quality of measurements that can be made of network orutes, by merely observing traffic timings from the end-points, has been explored.

Since it was desired to measure latencies from an user process perspective, regular UDP packets were used for these measurements. Using ICMP or other related techniques was avoided, because such packets are treated differently by some routers in the Internet. ICMP-based techniques have been used before, e.g., `pathchar` [Ja 97]. Hence, those measures provide a perspective different from that of a regular user.

Some of previous work on Internet measurements have used TCP sources [Pa 97]. In these experiments, it was decided to use UDP instead, for a number of reasons. The interactions of packets through the network is complex enough and adding to it the complexity of the congestion control mechanisms of TCP would make the analysis of network characteristics more difficult.

Also, UDP packets can be controlled better, since the protocol stack minimally delays the sending of the UDP packets. Finally, most of the measurements that are made, depend on the size of packets. For TCP, the data segments and acks would typically differ in size, so the measurement on the data segment and the corresponding acks cannot be combined.

Another important point of contention, is whether measurements should be made actively or passively. The work reported here, makes active measurements, by sending probe packets into the network. The ideal mode of measurement should be passive, and the presented technique can easily be extended to perform passive measurements. Since, at this time, the attempt is also to learn more about the measurement techniques and the quality of information through them, it has been felt that active measurements provides a better understanding.

3 Network Model

In this section a deterministic model for a network connection and the definitions related to available capacity are described. The network model and associated analysis that is presented here, extends previous work discussed in the paper [JaAgSa 93].

3.1 Deterministic Model

For the purpose of this paper, a network connection is assumed to be uniquely defined by the two endpoints, and, all packets with the same source and destination addresses are part of the same connection. It is assumed that all packets of this connection follow a fixed sequence of routers. Each router is has multiple servers, serving a single queue of incoming packets. The number of servers in a router vary between 1 and m. (Figure 1). All the servers in a router have the same deterministic service times.

The service discipline followed at each of the routers is order-preserving, which reduces to a pure first-come-first-served (FCFS) discipline under the given assumptions. All servers are assumed to be work-conserving. Currently, it is also assumed that no other traffic is present in this path. This assumption will later be relaxed to factor arbirtary cross traffic. Using this model, the aim is to determine the departure times of the packets from the last router based on their arrival times at the first router, the service delays at the routers and the transit delays between the routers.

3.1.1 Notation

The notation used for the rest of the paper is as follows -

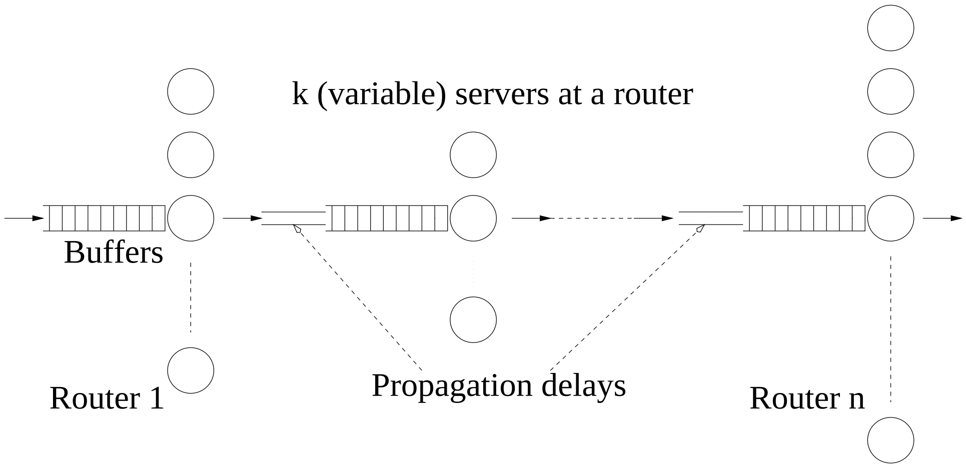

Figure 1: Sequence of multi-server routers with order-preserving discipline

- δ^i : Transit delay encountered by the packet in traversing the link between R_{i-1} and R_i. Also, $\delta^1 = 0$ is assumed, for convenience of derivation.
- s^i : Service time required to serve a packet at router R_i. It is assumed that all the packets require the same amount of processing time in any server of a given router.
- t^i : $\delta^i + s^i$, indicates the minimum time a packet takes between its departure from the router, R_{i-1} and its departure from the router, R_i.
- $\tau(a,b)$: $\sum_{i=a}^{b} t^i$, with $a \leq b$, which is the minimum transit time for any packet between routers, R_a and R_b.
- τ : $\tau(1,n)$ indicates the minimum transit delay for a packet between the arrival at the first router and departure from the last router. This is the time required due to the processing at each router and the transit delay at each hop.
- a_j^i : Arrival instant of packet j at router R_i. For convenience of derivation, $a_q^i = 0, \forall q \leq 0$.
- d_j^i : Departure instant of packet j from router R_i. For convenience of derivation, $d_j^0 = a_j^1$ and $d_q^i = 0, \forall q \leq 0$.
- $max_l(a,b)$: $\{\max(i) | R_i$ has l servers, $s^i \geq s^k, a \leq k \leq b$, where R_k has l servers$\}$ i.e., the highest indexed l-server router, which has the maximum service time among all routers between R_a and R_b (both inclusive). If there is no l-server routers then $max_l(a,b)$ is undefined and we define $s^{max_l(a,b)} = 0$.
- v_j^i : The virtual waiting time observed by packet j at router, R_i, is the amount of time packet j had to wait in the buffers of router R_i before it could be processed.

3.1.2 Without Cross Traffic

For any given packet, j, the following relations always hold -

$$\begin{aligned} a_j^i &= d_j^{i-1} + \delta^i, \forall i : 1 \leq i \leq n & (1) \\ d_j^i &= a_j^i + v_j^i + s^i, \forall i : 1 \leq i \leq n \\ \Rightarrow d_j^i &= d_j^{i-1} + v_j^i + \delta^i + s^i = d_j^{i-1} + v_j^i + t^i, \forall i : 1 \leq i \leq n & (2) \end{aligned}$$

Hence,

$$d_j^n = a_j^1 + \sum_i v_j^i + \sum_i t^i = a_j^1 + \sum_i v_j^i + \tau, \forall j : 1 \leq j \leq n \quad (3)$$

Note, in the above expression, d_j^n denotes the time when packet j departs from the last router, i.e., it is received by the user process at the destination. a_j^1 denotes the time the packet j arrives at the first router, ie., the time at which the packet is passed down by the user process to the network layer.

The attempt is to relate the terms, $d_j^n, d_j^0, s^{max_i(1,n)}$ and τ.

More specifically, the relationship is stated as a theorem -

Theorem 1 *For packets flowing through a network, modeled as above, the departure time of packet j from the network is given by*

$$d_j^n = \max(d_j^0 + \tau(1,n), \{d_{j-i}^n + s^{max_i(1,n)}\}_{i=1}^m)$$

If packet j was not buffered anywhere in the network, then $d_j^n = d_j^0 + \tau(1,n)$.
Otherwise, packet j was last buffered at some router R_i, with l servers, where $i = max_l(1,n)$. In this case, $d_j^n = d_{j-l}^n + s^{max_l(1,n)}$.

The proof of this theorem is given in the appendix.

Theorem 2 *A sequence of m-server routers $R_i, 1 \leq i \leq n$, (all the routers have m servers) is equivalent to a single composite m-server router, R, where service time of each of the servers of R is s^{max_m}, and propagation delay to the router R is τ.*

This is depicted in figure 2. Basically, for a sequence of m-server routers, theorem 1 states that -

$$d_j^n = \max(d_j^0 + \tau, d_{j-m}^n + s^{max_m})$$

From lemma 1 (appendix), for the composite router, the same relationship is got, by noting that the propagation delay to this router is τ and the service time of each server is s^{max_m}.

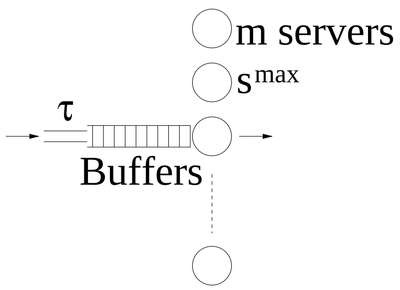

Figure 2: Composite router equivalent to a sequence of m-server routers

3.1.3 Factoring Cross Traffic

In a more realistic scenario, packets from other connections would also pass through the routers. Such a stream of packets is called, *cross traffic packets*, and the packets belonging to the connection under observation, *probe packets*. The cross traffic packets will also require processing from one or more of the routers used by the probe packets. As a consequence, processing time for the cross traffic packets need to be accounted for in estimating transit delays.

Consider the simplified scenario where are the routers are single servers. In this case, the probe packet departure instants from the network, in the absence of cross traffic, would be given by -

$$d_j^n = \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) \tag{4}$$

where, s^{max}, is defined as $s^{max_1(1,n)}$. The router corresponding to this service time is the *bottleneck router*.

To account for cross traffic, an extra variable is introduced, C_j, in the above equation for each packet j. Hence, the equation looks like -

$$d_j^n = \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) + C_j \tag{5}$$

C_j is defined as the *encountered delay* of the probe packet j for reasons other than processing of probe packets at the routers. Note, that C_j measures the delay, which is not already measured in $C_{j-i}, \forall i \geq 0$.

The delay is essentially due to either processing of cross traffic packets at the routers or routers taking breaks from packet processing, e.g., during routing updates. The term C_j does not exactly reflect the amount of cross traffic in the network during the period probe packet j is in the network. This is because there might be some cross traffic that arrived and got processed at a router without affecting any probe packets, if no probe packets arrived at that router over this duration. Hence, c_j only includes a lower-bound measure of the cross traffic in the network. Rewriting the equation 5 in terms of C_j, as -

$$C_j = d_j^n - \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) \tag{6}$$

gives a measure of the encountered delay for the packet j.

3.2 Virtual Waiting Time

Virtual waiting time at a router defines the amount of packet processing work that is pending at the router at any time, t. It, thus, gives a measure of the buffer occupancy at the router. The virtual waiting time at a router, R_i, at time t, given by $v^i(t)$, is formally defined as the amount of time a packet arriving at time, t, would have to wait before it can be serviced by the router.

A few simple observations about virtual waiting time at a router are -

1. $v^i(t) \geq 0$.
2. $v^i(T) = \max(v^i(t) - (T - t), 0)$,
   where $T \geq t$, there are no arrivals between time t and T. Essentially, the virtual waiting time at a router decreases at a linear unit rate and is lower-bounded by zero.
3. If a packet arrives at time t, then -

$$v^i(t^+) = v^i(t^-) + s^i \tag{7}$$

   where s^i denotes the service time of the router R_i.

The above assumes infinite buffer capacities in the router. If the buffer limit in a router is given by B, in units of packets, then it is re-defined in equation 7 as -

$$v^i(t^+) = \begin{cases} v^i(t^-) & \text{if } v^i(t^-) + s^i > Bs^i \\ & \text{and the packet is dropped} \\ v^i(t^-) + s^i_q & \text{otherwise} \end{cases} \tag{8}$$

The packet drop is according to the operations of a drop-tail router. The equation would be different if other techniques, like RED [FlJa 93], are implemented. Figure 3 illustrates these above observations.

The virtual waiting time at router, R_i, observed by probe packet j, is given by $v^i_j = v^i(a^{i\,+}_j)$, where a^i_j is the arrival instant of packet j at that router. This leads to a discrete model of virtual waiting times. A recurrence relation of virtual waiting times at router, R_i, of the probe packets can be defined as -

$$v^i_j = \begin{cases} 0 & \text{if } j \leq 1 \\ \max(v^i_{j-1} + s^i_j - (a^i_j - a^i_{j-1}), 0) & \text{otherwise} \end{cases} \tag{9}$$

in absence of any cross traffic packets.

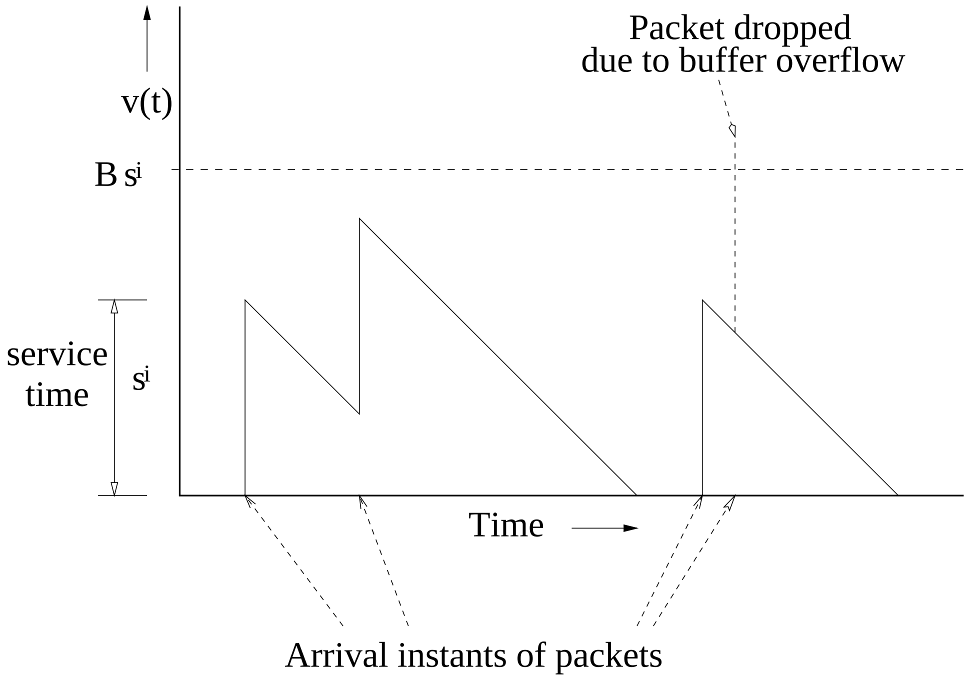

Figure 3: Virtual waiting time at a router

The above equation relates virtual waiting times of a single router. When the end-to-end connection comprising of multiple routers is considered, the term v_j^i is not observable from measurements made at the endpoints. So, Virtual Waiting Time for an end-to-end connection which can be measured from the endpoints needs to be defined. This is given by -

$$V_j = \sum_i v_j^i \tag{10}$$

The virtual waiting time of an end-to-end connection, thus, refers to the total time that a packet had to wait in the buffers of all the routers it encountered in the path.

From theorem 2, a sequence of single server routers can be replaced by a single composite router, with service time s^{max} and propagation delay to the router as τ. The virtual waiting time for this composite router would be given by V_j, the virtual waiting time of the end-to-end connection. Hence, analogous to equation 9, the recurrence relation of V_j terms can be stated as -

$$V_j = \begin{cases} 0 & \text{if } j \leq 1 \\ \max(V_{j-1} + s^{max} - (a_j^1 - a_{j-1}^1), 0) & \text{otherwise} \end{cases} \tag{11}$$

For completeness sake, a proof of the above equation is provided.

For $j = 1$, the packet is not buffered at any router, in absence of cross traffic. So $V_j = 0$.

For $j \geq 2, d_j^n = a_j^1 + \sum_i v_j^i + \sum_i t^i = a_j^1 + \sum_i v_j^i + \tau$, (Equation 3).

Also, $V_j = \sum_i v_j^i$.

So, $V_j = d_j^n - a_j^1 - \tau$
$= \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) - a_j^1 - \tau$ (Equation 4)
$= \max(d_j^0 - a_j^1, d_{j-1}^n + s^{max} - a_j^1 - \tau)$

But, $V_{j-1} = d_{j-1}^n - a_{j-1}^1 - \tau$ (similar to equation for V_j).
And, $d_j^0 - a_j^1 = 0$, by assumption.
So, $V_j = \max(0, V_{j-1} + s^{max} - a_j^1 + a_{j-1}^1)$, the desired result.

If the encountered delay is considered, as before, equation 9 would be modified as -

$$V_j = C_j + \begin{cases} 0 & \text{if } j \leq 1 \\ \max(V_{j-1} + s^{max} - (a_j^1 - a_{j-1}^1), 0) & \text{otherwise} \end{cases} \tag{12}$$

Note, this C_j term is the same as defined earlier in equation 6.

3.3 Available Capacity

In an end-to-end path of a connection, a packet moves from the source to the destination making use of the resources (e.g., buffer space and server processing time) at multiple routers along the path. These resources are shared by the packets of the given connection as well as cross traffic packets.

In this paper, the *instantaneous available capacity* of a path, $\alpha(t)$, is defined as a function of time to indicate the amount of router processing time available to packets of a connection. To determine the available capacity of a path, first the available capacity for a single router, R_i, is considered.

In absence of cross traffic packets, the router is capable of processing a probe packet in s^i time. Hence, this provides an upper-bound on the throughput that can be achieved on this connection. So, the end-user needs to keep track of the last time a packet was sent, and send packets exactly s^i apart, to achieve the maximum possible throughput.

Assume that the router has a buffer space for upto B^i packets. In presence of cross traffic, some of these B_i buffer slots would be occupied by probe packets as well as cross traffic packets. This is determined by the virtual waiting time at the router. If the end-user does not care about the transit delay, it is possible to insert a total of $\lfloor (B^i s^i - v^i(t))/s^i \rfloor$ at time t to arrive at the router at that instant. These packets will not be dropped by the router, since there is sufficient buffer space.

If the packets need to meet some QoS guarantee and the end-user wants the packets to have a transit delay $\leq \Delta$, then the number of packets that can be made to arrive at the router at time t is given by $\lfloor (\Delta - v^i(t))/s^i \rfloor$. Note, that $\Delta \leq B^i s^i$, the time needed by the router to process a completely filled buffer, or else this delay upper-bound would be violated by the router.

This leads to the first model of available capacity, which is called the *Greedy Available Capacity.* It is defined as -

$$\alpha(\Delta, t) = \max(\Delta - v^i(t), 0) \tag{13}$$

The above equation is for a single router. For a sequence of m -server routers, theorem 2 can be used to replace $v^i(t)$ by $V(t)$, where $V(t) = \sum_i v^i(t)$. Hence, equation 13 can be written for a sequence of multiple routers as -

$$\alpha(\Delta, t) = \max(\Delta - V(t), 0) \tag{14}$$

Thus, in this model, if the end-user introduces this amount of traffic into the network, it is possible that some of the cross traffic would be lost due to buffer overflow caused by the introduced traffic. This is not of consequence to the end-user and hence the model is called greedy.

The other extreme is the *Altruistic Available Capacity*, where introduction of traffic denoted by this available capacity would not affect the cross traffic packets at all, either in terms of delay or losses. This basically gives a measure of the duration of the next idle period of the router.

Another model is an intermediate scenario, where the traffic introduced by this metric would not cause any extra losses of cross traffic due to buffer overflow. However, the cross traffic packet may be delayed due to traffic introduced according to this metric. This model is called, the *Cooperative Available Capacity.*

It is easy to see that both Altruistic and Cooperative Available Capacities need knowledge of future cross traffic patterns, and hence cannot be experimentally determined. For the rest of this paper, only the Greedy Available Capacity model is considered.

In the discussion above, $\alpha(\Delta, t)$ has been treated as a continuous function of time. However, in practice, the system can be observed only at discrete instants in time. So, with each probe packet j of the connection, α_j, is associated, as the available capacity visible to the packet on its arrival.

In the greedy model, this leads to the relation -

$$\alpha_j(\Delta) = \Delta - V_j \tag{15}$$

All the available capacity measures described here, indicates router processing capabilities in terms of available processing times. To get the measure in terms of data size, the service time for the bottleneck router, i.e., the router with the slowest processing speed, needs to be determined. This is the router with service time, s^{max} defined before. Hence, available capacity is redefined in discrete time per probe packet j, as -

$$\rho_j(\Delta) = \frac{\alpha_j(\Delta).P}{s^{max}} \tag{16}$$

where, P is the packet size. Thus, ρ_j is the amount of data that could be injected into the network with packet j, without violating the transit delay limit, Δ.

To assess the practical applicability of these models presented, a series of experiments to multiple sites, were conducted.

4 Experimental Tool

The tool, NetDyn [Sa 91], was used to observe the detailed performance characteristics of an end-to-end connection. It has four main components that are shown in figure 4 each of which runs as a separate user process. The source process running at a local site, constructs a UDP packet with a sequence number (SSN) and a timestamp (STS) and sends it to the echo process running at a remote site. The echo process adds its own sequence number (ESN) and timestamp (ETS) to the UDP packet and sends it to the sink process which is usually run on the same local site as the source process. The sink process adds another timestamp (SiTS) and sends it via a reliable TCP connection to the logger process, which also usually runs on the same site as the source and the sink processes. The logger process just collects these packets and saves the information on permanent storage.

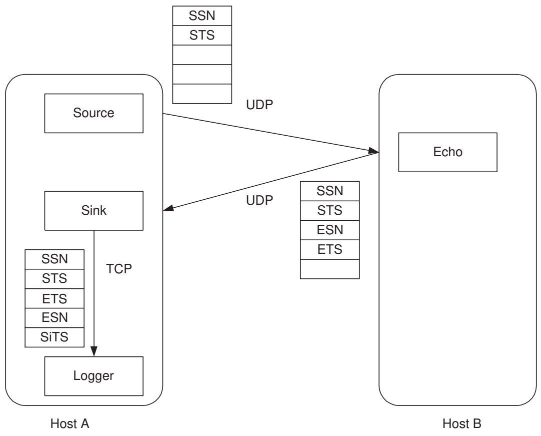

Figure 4: Organization of NetDyn

SSN	STS	ETS	SiTS	ESN
20	1000.863023	1000.992232	1001.062792	11
21	1000.873024	1001.002968	1001.073348	13
22	1000.873063	1001.002968	1001.071916	12
23	1000.883036	1001.011752	1001.08113	14
24	1000.883075	1001.012728	1001.083282	15
25	1000.892996	1001.022488	1001.089703	16
26	1000.89303	1001.022488	1001.089484	17
27	1000.902977	1001.026392	1001.090035	18
28	1000.903007	1001.027368	1001.09082	19
31	1000.922976	1001.046888	1001.108532	20

Figure 5: Logger Information

Each of the log records, thus, have two 4 byte sequence numbers and three 8 byte timestamps. An example of log records is shown in figure 5.

From the table it is easy to see that packets with SSN 21 and 22 were reordered in the return path, while packets with SSN 29 and 30 were lost in the forward path. Note, that the STS and SiTS timestamps are according to the clock in the local site machine, while the ETS timestamp is from the clock in the remote site machine. The clocks are, clearly, not synchronized.

Along with these processes, there is a TraceRoute process that runs at the local site, which runs the `traceroute` [Ja 96] utility, once every minute and records the route. Analysis of the traced routes provides information of route changes at the granularity of minutes.

The UDP packets are kept 32 bytes long minimally, but can be made of larger size. The sequence numbers help in detection of packet losses separately for the forward and reverse paths as well as packet reorders. The amount of traffic generated due to the measurements is about 25 Kbps in most cases.

The tool allows an user to specify the number of UDP packets to send, the size of each packet and the interval between the successive packets sent by the source.

NetDyn was the primary tool for the experiments.

5 Estimating s^{max}, the bottleneck service time

It has been shown earlier in literature [Wa 89] that in absence of cross traffic, no packets are ever queued after the bottleneck router, i.e., the router with service time, s^{max}. This can be seen from equation 4 and theorem 1, that if packet j is buffered somewhere in the network, then

$$d_j^n = d_{j-1}^n + s^{max}$$

i.e., the packets $j-1$ and j are exactly separated by s^{max} when they are discharged from the network. Hence, a pair of packets sent back to back would accurately estimate the bottleneck service time in absence of cross traffic. This packet pair technique has been used previously by researchers [Ke 91], [Bo 93], [CaCr 96a] and [Pa 97].

However, the scenario becomes different in presence of cross traffic packets. In particular, if some cross traffic arrives at a router between the arrival of the two packets $j-1$ and j of the pair, the inter-packet gap goes up. Also, if the two packets of the pair get queued at a router, R_i, after the bottleneck router, their difference reduces to the service time of that router $s^i < s^{max}$. Equation and derivations capturing these relationships can be found in [BaAg 98]. At many times it has been seen that the packet pair technique might actually capture the service time of the last router, instead of the bottleneck router.

Below two separate techniques to estimate s^{max}, is presented using data obtained through NetDyn.

5.1 Blocked Probe Packets

To understand the Blocked Probe Packets (BPP) technique, it is instructive to first observe the typical round trip time (RTT) plots that is obtained in the experiments in figure 6.

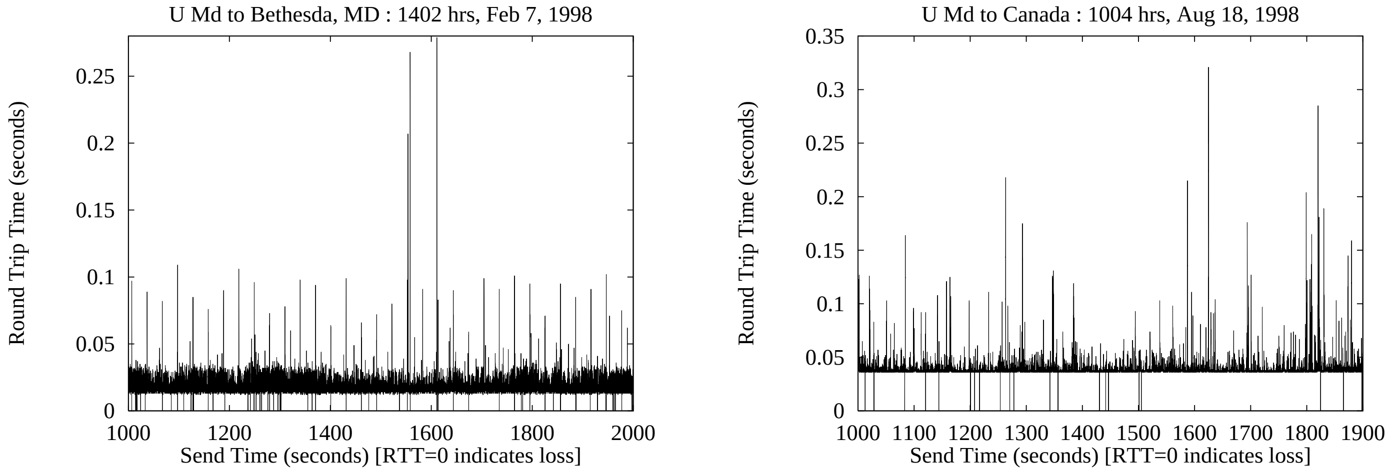

Figure 6: RTT plots showing frequent large peaks

It can be seen in the RTT plots, that there are frequent and somewhat periodic occurrence of large peaks in the RTTs of the packets. The packets before and after the duration of these large peaks are not significantly affected. These large duration of these peaks indicate that they cannot be due to cross traffic alone. They happen largely due to periodic breaks taken by the routers, mostly to perform routing updates, as reported in the paper [FlJa 94], and to do local statistics maintenance and updates. During these breaks, no packets are processed by the router. Thus, in these periods, a number of probe packets would accumulate in the buffers of the router. These packets would be later processed by the router in sequence, and they would be separated by some multiple of the service time of the router. Thus, if the Greatest Common Factor of the probe packet separations can be 'heuristically' esitmated, the service time of that router would be known. Observing a large number of these peaks, the service times of all the routers that are responsible for that peak can be estimated, and hence the service time of the bottleneck router.

However, for the experiments, this technique was not used. This and related techniques need to be studied closely before its applicability in measurements can be concluded.

5.2 Minimum RTT Packet Trains

In this technique of bottleneck service time estimation, a group of packets is sent back to back. Then the set of groups for which the one way transit time of the packets have been 'close' to the minimum observed in the experiment is identified. A low transit time indicates that the packets were queued for less time in the network. As a consequence, it is easy to see that these packets have a higher probability of passing through the routers in the network without being queued.

Finally, the distribution of these packet separations in a histogram, with ranges of 5 μs is observed and the modal value is picked as the estimate of the bottleneck service time. This gives

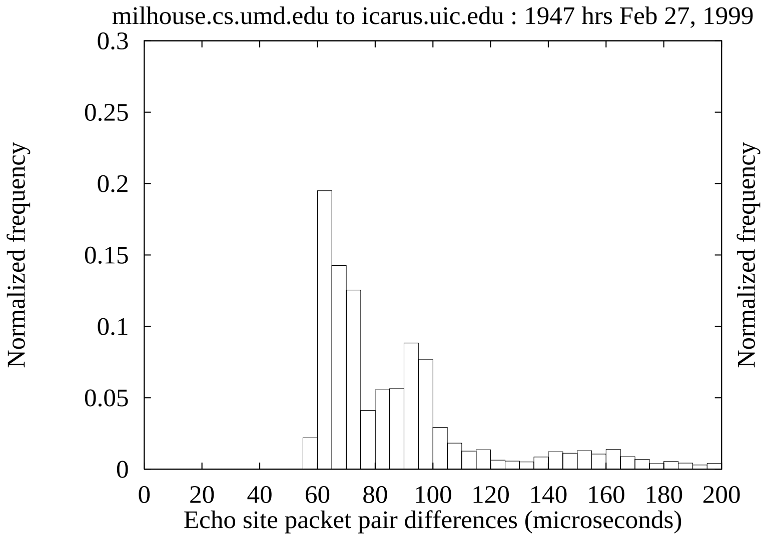

Figure 7: Using all packet pairs

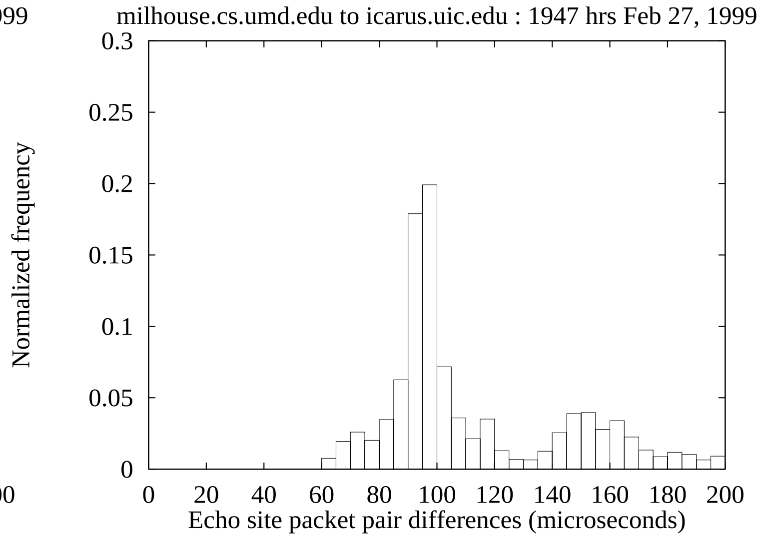

Figure 8: Using Min RTT packet trains

a good approximation of the bottleneck service time to within about 10 μs. In the observations of the histogram, it is seen that about 30-40 % of the packet separations lie within a range of 10 μs, while the remaining are evenly spread among other ranges. While some statistical analysis of this data might provide a more accurate estimate, the prevalent noise in the measurements make the significance of an accuracy below 10 μs, doubtful.

To see that this technique, indeed, provides better results than simple packet pairs, estimates made from an experiment conducted between *milhouse.cs.umd.edu* and *icarus.uic.edu* on February 27, 1999 are presented. In this experiment, a total of 120,000 packets were sent in groups of four. Packet groups were sent at intervals of 40 ms.

In the figures 7 and 8, histograms of the packet pair differences at the echo site, in ranges of 5 μs, between 0 and 200 μs, are plotted. The y-axis gives the normalized frequency of packet pair differences. When all packet pairs (Figure 7) are considered, the modal packet pair difference is about 65 μs, where nearly 20% of the packet pair differences lie. In figure 8, only those packets, whose RTTs were within 0.5 ms of the minimum of 20.05 ms, are considered. In this case, the modal packet pair difference is higher at about 100 μs, with again about 20% of the packet pair differences.

Packet pair differences higher than 200 μs are also seen, but the normalized frequency of such packet pair differences are negligible.

The bottleneck router service time in the minimum RTT packet train technique, thus, provides a better estimate of the bottleneck service time.

6 Results

Finally, the estimates derived using the given techniques are presented. Experiments using these techniques have been performed to multiple sites, ranging from geographically close regions (e.g., Bethesda, MD, USA which is within 20 miles of University of Maryland, College Park) to more distant places (e.g., Mexico, England and Taiwan). Here results from only two of our recent experiments to a couple of sites in the USA, have been described. One is to a site in Chicago and the other is in California.

The Chicago experiment was conducted on February 27, 1999 at 1947 hrs, by sending a total of 120,000 packets in groups of four, at intervals of 40 ms. The figures 9 and 12 plot the RTT and available capacity estimate for this experiment. The minimum RTT was 20.05 ms.

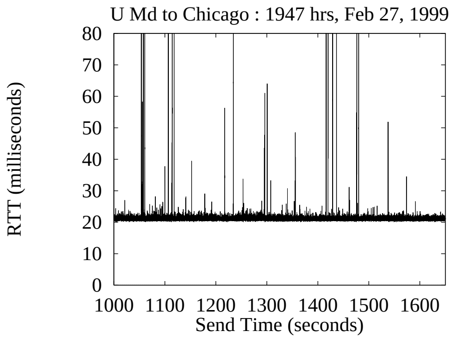

Figure 9: Round Trip Time

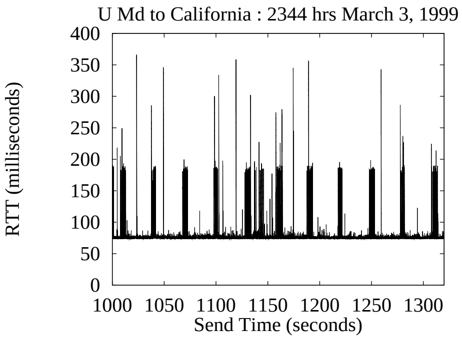

Figure 10: Round Trip Time

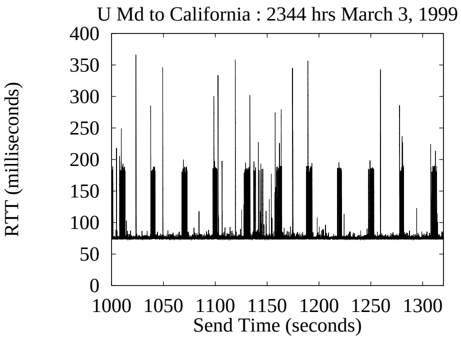

Figure 11: Round Trip Time

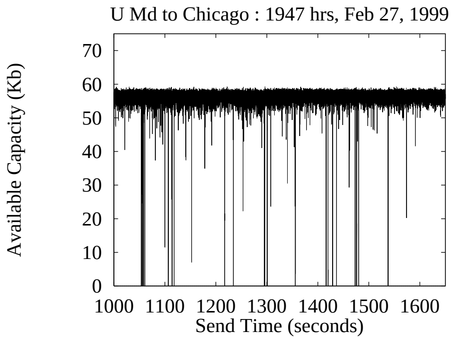

Figure 12: $\rho(22ms, t)$

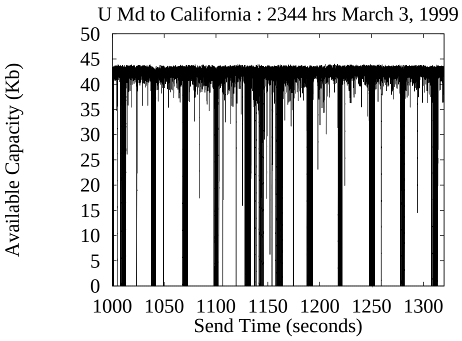

Figure 13: $\rho(75ms, t)$

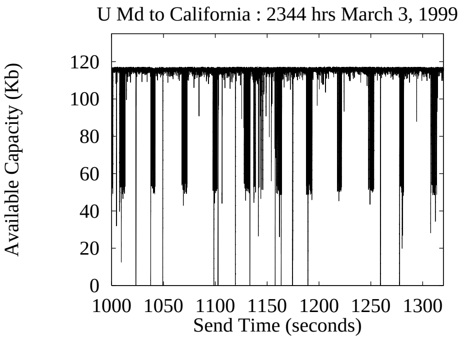

Figure 14: $\rho(200ms, t)$

The important observation to be made in these plots is the location of the RTT peaks in figure 9. These are periods of high encountered delay, and using the network model, the virtual waiting time of the end-to-end connection estimated, are correspondingly high at exactly the same instants. As a consequence, during these times, the available capacity (figure 12), goes to zero, as would be expected. This is an example of how the model accounts for changes in encountered delay in the network path.

A similar observation can be made in the California experiment, conducted on March 03, 1999 at 2344 hrs. A total of 76,000 packets were sent in groups of two, at intervals of 10 ms, with a minimum RTT of 72.7 ms. As an example, of how the model handles different transit delay requirements, available capacities are plotted for $\Delta = 75$ ms in figure 13 and for $\Delta = 200$ ms in figure 14. Figures 10 and 11 show the corresponding RTT plots for comparison to the available capacity plots. As would be expected, it is seen that for the periods when the RTT peaks ~ 185 ms, the estimate of available capacity, with $\Delta = 75$ ms, goes to zero in figure 13. For the same RTT peaks of ~ 185 ms, the estimate of available capacity is ~ 50 Kb, for $\Delta = 200$ ms in figure 14.

7 Conclusions and Future Work

In this paper, the definition of capacity that is available to a connection endpoint is defined, as visible to an user. The model used, has also been described and analytical expressions to calculate the available bandwidth has been provided. Also multiple mechanisms to calculate the service time of the bottleneck router in a connection is described and improved performance over packet pair techniques proposed previously is validated. Through the use of the NetDyn tool, estimates of the available capacity is made, and results from two of our recent experiments is presented.

A natural extension of the work described here, would be to enhance the mechanisms to permit passive measurements. Although, the amount of active measurement traffic that we generate is about 25 Kbps, if this technique is to be deployed widely, these small traffic amounts from large number of sites might swamp the Internet with such measurement traffic alone.

Another interesting extension would be to handle router buffering capacities and its implications in the model. Mechanisms to estimate the 'bottleneck' buffer capacity would be of significant relevance.

References

[Wa 89] J. Waclawsky. Window dynamics. PhD Thesis, University of Maryland, College Park, 1989.

[Ke 91] S. Keshav. A control-theoretic approach to flow control. Proceedings of SIGCOMM, 1991.

[Sa 91] D. Sanghi. `NetDyn`. http://www.cs.umd.edu/projects/netcalliper/NetDyn.html, 1991.

[JaAgSa 93] B.N. Jain, A. Agrawala and D. Sanghi. Model and analysis of a virtual circuit with cross traffic. Journal of Internetworking, V.4 N.2, June 1993.

[FlJa 93] S. Floyd and V. Jacobson. Random Early Detection for Congestion Avoidance. IEEE/ACM Transactions on Networking, V.1 N.4, August 1993.

[Bo 93] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. Proceedings of SIGCOMM, 1993.

[FlJa 94] S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. IEEE/ACM Transactions on Networking, V.2 N.2, April 1994.

[CaCr 96a] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet switched networks. BU-CS-96-006, Technical Report, Boston University, 1996.

[CaCr 96b] R. L. Carter and M. E. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. BU-CS-96-007, Technical Report, Boston University, 1996.

[Ja 96] V. Jacobson. `traceroute`. ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1996.

[Ja 97] V. Jacobson. `pathchar`. ftp://ftp.ee.lbl.gov/pathchar, 1997.

[Pa 97] V. Paxson. Measurement and analysis of end-to-end Internet dynamics. PhD Thesis, University of California, Berkeley, 1997.

[BZBHJ 97] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205, 1997.

[BaAg 98] S. Bahl and A. Agrawala. Analysis of a packet-pair scheme for estimating bottleneck bandwidth in a network. CS-TR-3924, Technical Report, University of Maryland, College Park, 1998.

A Appendix : Proof of Theorem 1

Observation 1 *A packet j is buffered at a l-server router, R_i, $\Leftrightarrow d^i_j = d^i_{j-l} + s^i$.*

Observation 2 *A packet j is not buffered at a router R_i, $\Leftrightarrow d^i_j = a^i_j + s^i = d^{i-1}_j + \delta^i + s^i = d^{i-1}_j + t^i$. Note, if $d^i_j = d^{i-1}_j + t^i = d^i_{j-l} + s^i$, we consider the packet j as buffered or not buffered at router R_i according to convenience of the proof.*

Observation 3 *In general, for a packet j at a l-server router, R_i,*

1. $d_j^i \geq a_j^i + s^i = d_j^{i-1} + t^i$
2. $d_j^i \geq d_{j-l}^i + s^i$

Observation 4 *For $r > i, d_j^r \geq d_j^i + \tau(i+1, r)$.*

Observation 5 *If packet j was last buffered in router, R_i with l-servers, and no where after that, then for $i < n, d_j^n = d_j^i + \tau(i+1, n) = d_{j-l}^i + s^i + \tau(i+1, n)$*

This follows from observation 1 and 2 and definition of $\tau(a, b)$.

Lemma 1 *For packet j at a l-server router, R_i, $d_j^i = \max(d_{j-l}^i + s^i, d_j^{i-1} + t^i)$.*

For packet j, there are two possibilities.

1. It is not buffered at router R_i.
   Then $d_j^i = d_j^{i-1} + t^i$ (Observation 1)
   Also, $d_j^i \geq d_{j-l}^i + s^i$ (Observation 3)
   So, $d_j^i = \max(d_{j-l}^i + s^i, d_j^{i-1} + t^i)$.
2. It is buffered at R_i.
   Then $d_j^i = d_{j-l}^i + s^i$ (Observation 2)
   Also, $d_j^i \geq d_j^{i-1} + t^i$ (Observation 3)
   So, $d_j^i = \max(d_{j-l}^i + s^i, d_j^{i-1} + t^i)$.

Lemma 2 *If, $\forall j > 0, a_j^i - a_{j-k}^i \geq A$, A is some positive constant, then, $\forall j > 0, d_j^i - d_{j-k}^i \geq A$.*

This is proved by induction on j. Assume, router R_i has l servers.
Base case : $d_j^i \geq a_j^i$, (departures happen after arrivals).
Note, by the assumptions, $a_r^i, d_r^i = 0, \forall r < 0$. For $j \leq k$, $a_j^i \geq A$ and hence, $d_j^i > A$.
So for $j \leq k$, $d_j^i - d_{j-k}^i = d_j^i - 0 > A$.
Inductive case : (for $j > k$) It is given that $\forall j, a_j^i - a_{j-k}^i \geq A$. Suppose $\forall j < r, d_j^i - d_{j-k}^i \geq A$.
It is required to show that $d_r^i - d_{r-k}^i \geq A$.
There are two cases -

1. If packet $r - k$ is not buffered at router R_i.
   $d_r^i \geq a_r^i + s^i$ (Observation 3)
   $d_{r-k}^i = a_{r-k}^i + s^i$ (Observation 2)
   So, $d_r^i - d_{r-k}^i \geq a_r^i - a_{r-k}^i \geq A$.

2. If packet $r-k$ is buffered at router R_i.
   $d^i_r \geq d^i_{r-l} + s^i$ (Observation 3)
   $d^i_{r-k} = d^i_{r-k-l} + s^i$ (Observation 1)
   So, $d^i_r - d^i_{r-k} \geq d^i_{r-l} - d^i_{r-k-l} \geq A$ (hypothesis).

This proves the lemma.

Corollary 1 *If $\forall j > 0, d^i_j - d^i_{j-k} \geq A$, then $\forall j > 0, d^r_j - d^r_{j-k} \geq A$, where $r \geq i$.*

This is proved by induction.
Base Case : This is trivially true for $r = i$.
Inductive case : (for $r > i$) It is given $\forall j > 0, d^i_j - d^i_{j-k} \geq A$. Assume, $\forall j > 0, d^{r-1}_j - d^{r-1}_{j-k} \geq A$.
It is needed to show that $\forall j > 0, d^r_j - d^r_{j-k} \geq A$.
For $j - k \leq 0, d^i_{j-k}, d^r_{j-k} = 0$. And $d^r_j \geq d^i_j$, for $r > i$.
So, $d^r_j - d^r_{j-k} \geq d^i_j - d^i_{j-k} \geq A$.
For $j - k > 0$ $a^r_j = d^{r-1}_j + \delta_r$, where δ_r is the transit delay between routers, R_{r-1} and R_r.
Similarly, $a^r_{j-k} = d^{r-1}_{j-k} + \delta_r$.
Hence, $a^r_j - a^r_{j-k} = d^{r-1}_j - d^{r-1}_{j-k} \geq A$. So, $\forall j, d^r_j - d^r_{j-k} \geq A$ (Lemma 2).

Corollary 2 $\forall j > 0, d^i_j - d^i_{j-l} \geq s^{max_l(1,i)}$.

If there are no l-server routers between routers R_1 and R_i (both inclusive), then $s^{max_l(1,i)} = 0$.
Since under order-preserving discipline, $d^i_j \geq d^i_{j-l}, \forall l > 0$, the corollary is true.
If $max_l(1,i) = i$, then the corollary is true (Observation 3).
Otherwise, if $max_l(1,i) = q < i$, then $\forall j, d^q_j - d^q_{j-l} \geq s^q$, (Observation 3) noting that R_q has l servers.
Since, $i > q$, $\forall j, d^i_j - d^i_{j-l} \geq s^q$, (Corollary 1) i.e., $\forall j, d^i_j - d^i_{j-l} \geq s^{max_l(1,i)}$.

Lemma 3 *In R_i and R_k are two l-server routers, with $i < k$ and $s^i > s^k$, then, no packet would be buffered in R_k.*

$\forall j, d^i_j - d^i_{j-l} \geq s^i$, (Observation 3).
$a^{i+1}_j = d^i_j + \delta^i$ and $a^{i+1}_{j-l} = d^i_{j-l} + \delta^i$
With $k > i$, $d^k_j - d^k_{j-l} \geq s^i$ (Corollary 2).
i.e., $d^k_j \geq d^k_{j-l} + s^i > d^k_{j-l} + s^k$.
However, if some packet, q is buffered at router, R_k, then $d^k_q = d^k_{q-l} + s^k$, a contradiction.
Hence, a packet cannot be buffered at router, R_k.

Corollary 3 *If the last router in which a packet j is buffered, R_i has l servers, then $i = max_l$, where $max_l = max_l(1,n)$.*

$d_j^i - d_{j-l}^i = s^i$ (Observation 1).
If $i \neq max_l$, then there are two cases -

1. $max_l < i$.
   Note, s^{max_l} must be strictly $> s^i$, from definition of max_l.
   Then no packet would be buffered in R_i, (Lemma 3), which is a contradiction.

2. $max_l > i$.
   Since, R_i is the last router in which the packet j is buffered, $d_j^{max_l} = d_j^i + T(i+1, max_l)$.
   Note, $\tau(i+1, max_l)$ is defined as $\sum_{i=i+1}^{max_l} t^i$.
   Similarly, $d_{j-l}^{max_l} \geq d_{j-l}^i + \tau(i+1, max_l)$.
   So, $d_j^{max_l} - d_{j-l}^{max_l} \leq d_j^i - d_{j-l}^i = s^i$ (Observation 1 - packet j was buffered in R_i).
   i.e., $s^i \geq d_j^{max_l} - d_{j-l}^{max_l}$.
   Also, $d_j^{max_l} - d_{j-l}^{max_l} \geq s^{max_l}$ (Observation 3).
   i.e., $s^i \geq d_j^{max_l} - d_{j-l}^{max_l} \geq s^{max_l}$.
   $\Rightarrow s^i \geq s^{max_l} \Rightarrow s^i = s^{max_l}$ (noting that R_i has l servers).
   Hence, $d_j^{max_l} - d_{j-l}^{max_l} = s^{max_l}$, i.e. the packet j was buffered at router R_{max_l} (Observation 1), contradicting that R_i was the last router where the packet was buffered.

Theorem 1 *For packets flowing through a network of n routers, and router R_i, has l_i-servers, where $1 \leq l_i \leq m, \forall i, 1 \leq i \leq n$, the departure time of packet j from the network is given by*

$$d_j^n = \max(d_j^0 + \tau(1,n), \{d_{j-i}^n + s^{max_i(1,n)}\}_{i=1}^m)$$

If packet j was not buffered anywhere in the network, then $d_j^n = d_j^0 + \tau(1,n)$.
Otherwise, packet j was last buffered at some router R_i, with l servers, where $i = max_l(1,n)$. In this case, $d_j^n = d_{j-l}^n + s^{max_l(1,n)}$.

Proof : The result is proved by induction on n, the number of routers.
Base case : Assume that router, R_1 has l servers. Note that, $s^{max_l(1,1)} = s^1$ and $s^{max_i(1,1)} = 0, \forall i \neq l$.
Also, for a single l-server router, $d_j^1 = \max(d_j^0 + t^1, d_{j-l}^1 + s^1)$ (Lemma 1).
i.e., $d_j^1 = \max(d_j^0 + t^1, d_{j-l}^1 + s^{max_l(1,1)})$.
$d_j^1 \geq d_{j-i}^1 + s^{max_i(1,1)}$ (Corollary 2). $\Rightarrow d_j^1 = \max(d_j^0 + t^1, \{d_{j-i}^1 + s^{max_i(1,1)}\}_{i=1}^m)$. Note, in this case $\tau(1,1) = t^1$.
If packet j is not buffered ar router R_1, then $d_j^1 = d_j^0 + t^1$ (Observation 2), and if it is buffered then $d_j^1 = d_{j-l}^1 + s^1 (Observation 1) = d_{j-l}^1 + s^{max_l(1,1)}$.
Inductive case : Assume that the hypothesis holds for the first n routers. It is required to show that it holds when another router, R_{n+1}, is added. Assume that R_{n+1} has l servers.
The proof is split into two cases -

1. If packet j is buffered at R_{n+1}.
   So, $d_j^{n+1} = d_{j-l}^{n+1} + s^{n+1}$ (Observation 1) and $n+1 = max_l(1, n+1)$ (Corollary 3).
   i.e., $d_j^{n+1} = d_{j-l}^{n+1} + s^{max_l(1,n+1)}$.
   Also, $\forall i, d_j^{n+1} \geq d_{j-i}^{n+1} + s^{max_i(1,n+1)}$ (Corollary 2).
   Also, $d_j^{n+1} \geq d_j^0 + \tau(1, n+1)$ (Note, $\tau(1, n+1)$ is the minimum transit time).
   Hence, $d_j^{n+1} = \max(d_j^0 + \tau, \{d_{j-i}^{n+1} + s^{max_i(1,n)}\}_{i=1}^m)$.

2. If packet j is not buffered at R_{n+1}.

   (a) If the packet is not buffered anywhere in the first n routers, then $d_j^n = d_j^0 + \tau(1, n)$ (hypothesis).
   Then, $d_j^{n+1} = d_j^0 + \tau(1, n) + t^{n+1} = d_j^0 + \tau(1, n+1)$ and the packet is not buffered anywhere in the network.
   Also $\forall i, d_j^{n+1} \geq d_{j-i}^{n+1} + s^{max_i(1,n+1)}$ (Corollary 2), we have -
   $d_j^n = \max(d_j^0 + \tau(1, n+1), \{d_{j-i}^n + s^{max_i}\}_{i=1}^m)$.

   (b) If the packet was last buffered in at the router, R_q with k servers, then $d_j^n = d_{j-k}^n + s^{max_k(1,n)}$. Also, $q = max_k(1, n)$ (Corollary 3).
   So, $d_j^{n+1} = d_{j-k}^{max_k(1,n)} + s^{max_k(1,n)} + \tau(max_k(1, n) + 1, n+1)$ (Observation 5).
   $d_{j-k}^{n+1} \geq d_{j-k}^{max_k(1,n)} + \tau(max_k(1, n) + 1, n+1)$ (Observation 4).
   So, $d_j^{n+1} - d_{j-k}^{n+1} \leq s^{max_k(1,n)}$.
   $\forall p, d_p^n \geq d_{p-k}^n + s^{max_k(1,n)}$ (Corollary 2).
   Hence, $\forall p, d_p^{n+1} \geq d_{p-k}^{n+1} + s^{max_k(1,n)}$ (Corollary 1).
   Also just shown, $d_j^{n+1} - d_{j-k}^{n+1} \leq s^{max_k(1,n)}$.
   $\Rightarrow d_j^{n+1} - d_{j-k}^{n+1} = s^{max_k(1,n)}$.
   $k \neq l \Rightarrow s^{max_k(1,n)} = s^{max_k(1,n+1)}$.
   $k = l$ and $s^{n+1} < s^{max_k(1,n)} \Rightarrow s^{max_k(1,n+1)} = s^{max_k(1,n)}$.
   $k = l$ and $s^{n+1} \geq s^{max_k(1,n)} \Rightarrow d_j^{n+1} - d_{j-k}^{n+1} = s^{n+1}$, i.e., packet j is buffered in router R_{n+1}, a contradiction.
   Hence, $d_j^{n+1} - d_{j-k}^{n+1} = s^{max_k(1,n+1)}$.
   i.e., $d_j^{n+1} = d_{j-k}^{n+1} + s^{max_k(1,n+1)}$ and the packet is last buffered in router, R_q,with k servers where $q = max_k(1, n)$.
   Also, $\forall i, d_j^{n+1} \geq d_{j-i}^{n+1} + s^{max_i(1,n+1)}$ (Corollary 2).
   Also, $d_j^{n+1} \geq d_j^0 + \tau(1, n+1)$ (Note, $\tau(1, n+1)$ is the minimum transit time).
   Hence, $d_j^{n+1} = \max(d_j^0 + \tau(1, n+1), \{d_{j-i}^{n+1} + s^{max_i(1,n)}\}_{i=1}^m)$

This proves the theorem.