Estimating Available Capacity of a Network Connection

Suman Banerjee, Ashok K. Agrawala
Department of Computer Science, University of Maryland, College Park, USA
{suman,agrawala}@cs.umd.edu

Abstract— **Capacity measures for a network connection across the Internet can be useful to many applications. Its applicability encompasses QoS guarantees, congestion control and other related areas. In this paper, we define and measure the available capacity of a connection, through observations at endpoints only. Our measurements account for variability of cross traffic that pass through the routers handling this connection. Related to the estimation of available capacity, we suggest modifications to current techniques to measure packet service time of the 'bottleneck' router of the connection. Finally, we present estimation results on wide-area network connections from our experiments to multiple sites.**

I. Introduction

In this paper we present techniques to estimate *available capacity* of an end-to-end connection. We define available capacity $\alpha(\Delta, t)$, at time, t, to indicate the amount of data that could be inserted into a network path at time, t, so that the transit delay of these data packets would be bounded by a maximum permissible delay, Δ. In this paper, we make a distinction between the terms *capacity* and *bandwidth*. By capacity, we mean data volume and not data rate, e.g., our available capacity measure indicates the volume of data that can be inserted into the network at time t, to meet the delay bound, and does not indicate the rate at which to insert. However, since the available capacity is estimated in discrete time, a bandwidth estimate can be obtained from it, by distributing the capacity over the intervals between the discrete time instants.

Some traditional techniques for available bandwidth use throughput to provide a coarse estimate of bandwidth, e.g., TCP [Pos 81c] with congestion control [Ja 88]. Hence, in this mechanism, the bandwidth available estimate is directly related to the throughput that the sender is willing to test at any instant. On a packet loss, this mechanism provides a loose upper bound for the available bandwidth. As noted in [LaBa 99], packet loss is actually a better estimate of buffer capacities in the network, than of available bandwidth.

Some other work on identifying the available bandwidth addresses the measurement of the bottleneck bandwidth e.g., [Bo 93], *bprobe* tool in [CaCr 96a], [Pa 97b] and [LaBa 99] or all link bandwidths of a network path [Ja 97]. The technique described in [LaBa 99] also estimates the changing bottleneck bandwidth due to path changes. But, bandwidth available on a network path, may often be less than the bottleneck bandwidth, and may also go to zero, due to cross traffic in the path. Our measure differs from these previous work, as we account for the capacity lost due to cross traffic, in our estimates.

The *cprobe* tool in [CaCr 96a] provides an available bandwidth measure, which accounts for cross traffic. They do so by sending a stream of packets, at a rate higher than the bottleneck bandwidth, and then computing the throughput of this stream, using simple average. In our technique, we estimate available capacity as experienced by each probe packet using the network model we describe and evaluate in section 3, and not as an average over a set of probes.

To measure the available capacity of a connection, we make measurements at the endpoints only. This technique adapts quickly with changing cross traffic patterns and, hence can be used to provide a useful feedback to applications and users.

This available capacity knowledge can be beneficial to real-time applications that require estimates of network carrying capacities. The available capacity measure that we provide are parameterized by a maximum permissible delay for packet delivery, Δ. Hence, if an application realizes that for its required transit delay, the available capacity is too low, it might decide to forgo sending packets during periods of low available capacity which may be below its minimum requirements from the network. Also, for uncontrolled non real-time data sources, this estimate would provide a feedback about the network congestion. Such sources might find such information useful to traffic generation rates. Our technique can be refined to serve as either controlling or policing mechanisms for all uncontrolled data sources. An 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.

The main contribution in this paper has been to define available capacity in terms of the packet transit delay in presence of cross traffic, and develop techniques to estimate it over wide-area connections. In this process, we also suggested extensions to current techniques for a more accurate measure of the bottleneck service time. Finally, we

have presented results of using our technique over network connections across the Internet.

In section 2 we discuss certain issues on Internet measurements. In section 3, we provide an overview of our available capacity estimation technique. In section 4, we analyze our network model and quantify available capacity. In section 5, we describe our measurement tool NetDyn [Sa 91]. We provide mechanisms to evaluate the service time of a bottleneck router in a connection, in section 6. In section 7, we present results from some of our recent experiments. Finally, in section 8, we summarize our work and outline some future directions of research.

II. Issues on Measurements

Measurements in the Internet is a difficult problem to approach, particularly, because of the large number of factors that are unknown and uncontrollable, from endpoints. 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. Our 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 deployment or special facilities was needed in the network. Through the work reported in this paper, we have explored the quality of measurements that can be made of network routes, by merely observing traffic timings from the end-points. All our measurements were made using regular UDP [Pos 80] packets. Some previous work on bandwidth measurement, e.g., `pathchar` [Ja 97], uses ICMP [Pos 81b]. However, many routers in the Internet react differently to ICMP packets. Moreover, processing time required for ICMP packets would different from regular user IP [Pos 81a] traffic and might not correctly measure the user perspective of the available capacity. Finally, in this work reported, we make active measurements, by introducing probe traffic into the network. We believe a technique based on a passive mesaurement scheme will be more useful for actual deployment in the Internet. However, since, at this time, we are trying to learn more about the Internet measurement techniques and the quality of information available through them, we feel active measurements provides us a better understanding.

Another crucial consideration in network measurements is detection and elimination of clock skews between the endpoints. Some recent work in this regard has been reported in [Pa 98] and [MoSkTo 99]. In our work, we sidestep this issue, by turning the probe packets around to the source host and then operating only on round-trip times of the packets. This is easy for us, since we make active measurements and can easily control the probe packets. However, by taking advantage of the clock skew elimination techniques in the literature, our technique should be extensible to one-way transit times.

III. Technique Overview

We use our tool, NetDyn [Sa 91], to send a sequence of probe packets between the two endpoints of a network connection. For each of these probe packets we record the time when the packet is inserted into the network and the time it arrives at the destination at the other end of the network.

We have defined a deterministic model of a network connection, which is described in the next section. In this model, it is assumed that all packets of a connection follow a fixed sequence of routers. Each router is modeled as a multi-server single queue node, (Figure 1), with deterministic service times.

Using this model, we derive a recurrance relation, (Theorem 1), of the time, d_j^n, a probe packet, j, arrives at the destination of the network connection. The deviation of the observed value from the expected value of the arrival time the packet at the destination defines the amount of delay encountered by this packet in the network. This delay will generally be due to processing delays for cross traffic packets, as well as due to router blockages to process periodic routing updates. More importantly, this difference indicates the part of the network capacity, that is unavailable to the connection under observation, e.g., for single-server routers, this value as oberserved by probe packet j is given by $C_j = d_j^n - \max(d_j^0 + \tau, d_{j-1}^n + s^{max})$, as shown in equation 3. We also define *Virtual Waiting Time*, $v(t)$, at a router, to indicate the queueing and processing delay encountered by a packet at this router that arrives at time t.

The available capacity, $\alpha(\Delta, t)$, where Δ is the transit delay bound parameter, can then be computed as shown in equation 8. The relation in Equation 8 is valid for a network connection comprising of a single router. This can be extended to a connection with multiple routers, which in disrete time can be represented as $\alpha_j(\Delta) = \max(\Delta - V_j, 0)$, (Equation 10). This uses the fact that a sequence of routers, modeled as stated, is equivalent to a single router with the service time equal to the bottleneck service time of the actual routers. This is shown in theorem 2.

This available capacity measure computed in equation 10 is in units of time. We divide this measure by the service time at the bottleneck router, to obtain the available capacity in units of data volume (i.e., in Kbits). Our technique to estimate the bottleneck service time, called Minimum RTT packet trains, is described in section 6, and this extends the

packet pair technique in both mechanism and accuracy.

In all the mechanisms that we use for our estimation, we rely only on timing measurements made at the end-points of the connection. We do not need to know the individual service times of the different routers in the network, except the service time of the bottleneck router, which is also inferred from the end-point measurements.

We should, however, add that we do not account for router buffer capacities in our model of the network connection. Clearly, packet losses due to buffer overflow will cause mismatches, since our model, as defined, cannot explain packet losses. Hence, the technique will work fine in presence of infrequent packet losses with mismatches during packet loss instances. We are addressing this issue in our future work.

IV. Network Model

In this section we describe a deterministic model for a network connection and our definitions related to available capacity.

A. Deterministic Model

For the purpose of this paper, we treat all packets between the same source and destination addresses to be part of one connection. We assume for the results below, that all packets follow a fixed sequence of routers. Each router 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.

A First-Come-First-Serve (FCFS) service discipline is followed at each router. All servers are assumed to be work-conserving. We make no restrictions about the amount of other traffic in the path.

B. Notation

- δ^i : Transit delay encountered by the packet in traversing the link between R_{i-1} and R_i.
- s^i : Service time required to serve a packet at router R_i.
- t^i : $\delta^i + s^i$
- $\tau(a,b)$: $\sum_{i=a}^{b} t^i$, with $a \leq b$.
- τ : $\tau(1,n)$, indicates the minimum transit delay for a packet.
- a_j^i : Arrival instant of packet j at router R_i.
- d_j^i : Departure instant of packet j from router R_i. We also define $d_j^0 = a_j^1$.
- $max_l(a,b)$: $\{\max(i) | R_i$ 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 $s^{max_l(a,b)} = 0$.

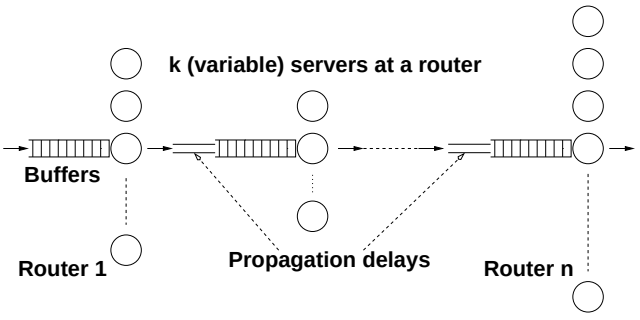

Fig. 1. Sequence of multi-server routers with order-preserving discipline

C. Without Cross Traffic

In absence of any other source of traffic in the network, we state two theorems.

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

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

If packet j was not buffered anywhere in the network, then $d_j^n = d_j^0 + \tau$.
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 is detailed in the appendix.

Theorem 2: A sequence of m-server routers $R_i, 1 \leq i \leq n$, is equivalent to a single composite m-server router, R, for end-to-end measures encountered by the packets, where service time of each of the servers of R is s^{max}, and propagation delay to the router R is τ.

This is depicted in Figure 3. It follows from theorem 1 and lemma 1 (appendix).

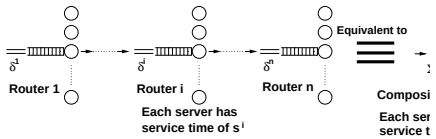

Fig. 2. Sequence of multi-server routers

C.1 Factoring Cross Traffic

In reality, packets from other connections, *cross traffic packets*, would also pass through the routers and affect delays encountered by the *probe packets* belonging to our connection.

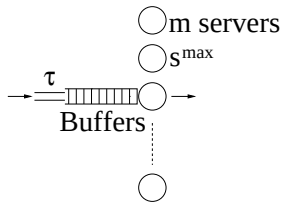

Fig. 3. Composite router equivalent

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 -

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

To account for cross traffic, we introduce an extra variable, C_j, in the above equation for each packet j. Hence, equation 1 can be modified as -

$$d_j^n = \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) + C_j \tag{2}$$

We define C_j as the *encountered delay* of the probe packet j for reasons other than processing of probe packets at the routers.

The delay is 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 probe packet j. Hence, C_j is only a lower-bound measure of the cross traffic in the network. Rewriting the equation 2 in terms of C_j, as -

$$C_j = d_j^n - \max(d_j^0 + \tau, d_{j-1}^n + s^{max}) \tag{3}$$

we can get a measure of the encountered delay for the packet j.

D. Virtual Waiting Time

We define virtual waiting time *at a router*, R_i, given by $v^i(t)$, as the amount of time a packet arriving at time, t, would have to wait before it can be serviced by the router. It, thus, gives a measure of the buffer occupancy at 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.
3. If a packet arrives at time t, then, $v^i(t^+) = v^i(t^-) + s^i$.

In this we assume infinite buffer capacities in the router. Figure 4 illustrates these above observations.

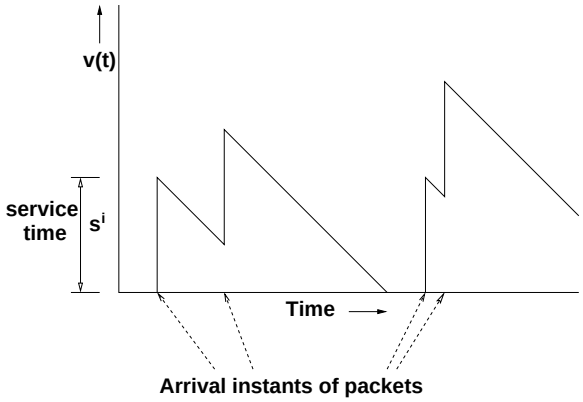

Fig. 4. Virtual waiting time at a router

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

$$v_j^i = \begin{cases} 0 & \text{if } j \leq 1 \\ \max(v_{j-1}^i + s_j^i - (a_j^i - a_{j-1}^i), 0) & \text{otherwise} \end{cases} \tag{4}$$

in absence of any cross traffic packets.

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

$$V_j = \sum_i v_j^i \tag{5}$$

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, we know that 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 4, we can state the recurrence relation of V_j terms as -

$$V_j = \begin{cases} 0 & \text{if } j \leq 1 \\ \max(V_{j-1} + s^{max} - (a_j^1 - a_{j-1}^1), 0) & \text{otherwise} \end{cases} \tag{6}$$

If we consider the encountered delay, as before, equation 4 would be modified as -

$$V_j = C_j + \begin{cases} 0 & \text{if } j \leq 1 \\ \max(V_{j-1} + s^{max} - (a_j^1 - a_{j-1}^1), 0) & \\ & \text{otherwise} \end{cases} \quad (7)$$

Note, this C_j term is the same as defined earlier in equation 3.

E. Available Capacity

In this paper, we define the *instantaneous available capacity* of a path, $\alpha(t)$, 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, we first consider the available capacity for a single router, R_i.

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, which is one packet in time s^i. However, in presence of cross traffic, the throughput would be even lower.

We define available capacity in terms of a QoS parameter, Δ, an upper-bound on the permissible transit delay for the packets.

The number of packets that can arrive at the router, R_i, at time t and encounter an end-to-end transit delay $\leq \Delta$ is given by $\lfloor(\Delta - v^i(t))/s^i\rfloor$.

This leads to the continuous available capacity relation for a single router -

$$\alpha(\Delta, t) = \max(\Delta - v^i(t), 0) \quad (8)$$

For a sequence of m -server routers, we can use theorem 2 to replace $v^i(t)$ by $V(t)$, where $V(t) = \sum_i v^i(t)$. Hence, equation 8 can be written for a sequence of multiple routers as -

$$\alpha(\Delta, t) = \max(\Delta - V(t), 0) \quad (9)$$

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

The discrete available capacity relation is expressed as -

$$\alpha_j(\Delta) = \max(\Delta - V_j, 0) \quad (10)$$

All the available capacity measures described here, indicates router processing capabilities in units of time. To get the measure in units of data, we need to determine the service time for the bottleneck router. This is the router with service time, s^{max} defined before. Hence, available capacity is redefined in discrete time per probe packet j, as -

$$\rho_j(\Delta) = \frac{\alpha_j(\Delta).P}{s^{max}} \quad (11)$$

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, we conducted a series of experiments to multiple sites.

V. Experimental Tool

We used the tool, NetDyn [Sa 91], to observe the detailed performance characteristics of an end-to-end connection. It has four main components that are shown in Figure 5 each of which runs as a separate user process. The source process running at a site, Host A, constructs a UDP packet with a sequence number (SSN) and a timestamp (STS) and sends it to the echo process running at another site, Host B. 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 Host A The sink process adds another timestamp (SiTS) and sends it via a reliable TCP connection to the logger process. The logger just collects these packets and saves the information on permanent storage.

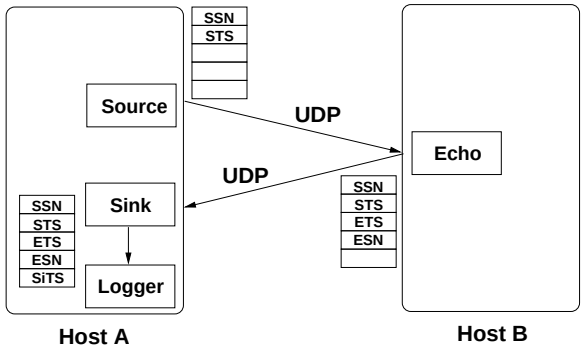

Fig. 5. Organization of NetDyn

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 6.

Along with these processes, there is a TraceRoute process that runs at Host A, 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

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

Fig. 6. Logger Information

detection of packet losses separately for the forward and reverse paths as well as packet reorders. The amount of traffic generating due to our measurements is about 25 Kbps in most cases.

VI. 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 1 and theorem 1, that if packet j is buffered somewhere in the network, then for single-server routers,

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

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 97a].

However, in presence of cross traffic, the packet pair differences are perturbed. 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 we present our extensions to the packet pair technique to estimate s^{max}.

A. Minimum RTT Packet Trains

In this technique of bottleneck service time estimation, we send a group of packets back to back. Then we identify

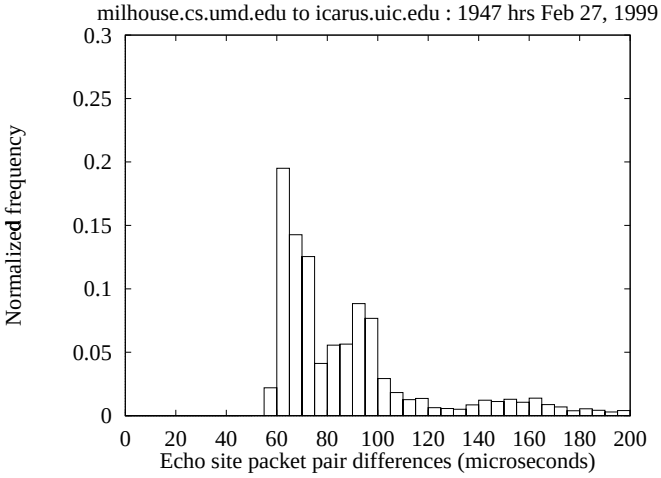

Fig. 7. Using all packet pairs

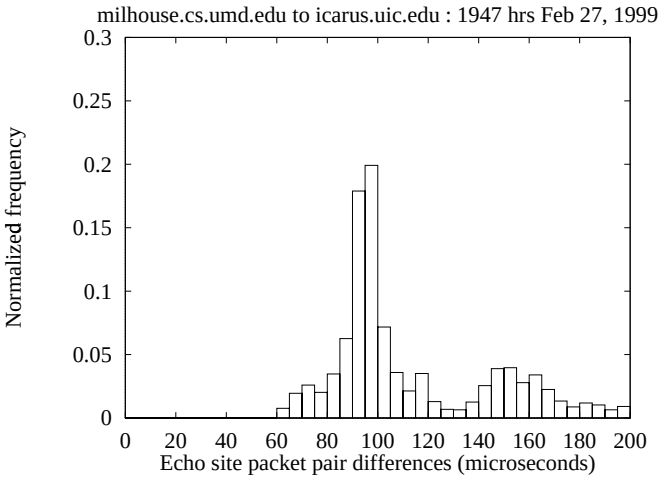

Fig. 8. Using Min RTT packet trains

the set of groups for which the transit delay of the packets have been 'close' to the minimum observed in the experiment. 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. From rough analysis of backbone router traffic characteristics [CAIDA], we expect all routers to see intermittent periods of zero queueing. Hence, we claim that the low RTT packets arrive at the receiving end-point unaffected by cross traffic. Hence, the packet pair differences of these packets are good choices to obtain the bottleneck service time estimate. In general, for our estimation, we chose packets below the 1-2 percentile of the RTT distribution of all packets.

Then we look at the histogram of these low RTT packet separations with ranges of 5 μs and pick the modal value as our estimate of the bottleneck service time. This gives a good approximation of the bottleneck service time to

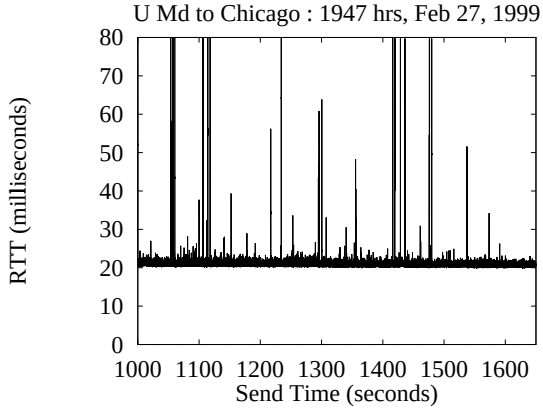

Fig. 9. Round Trip Time

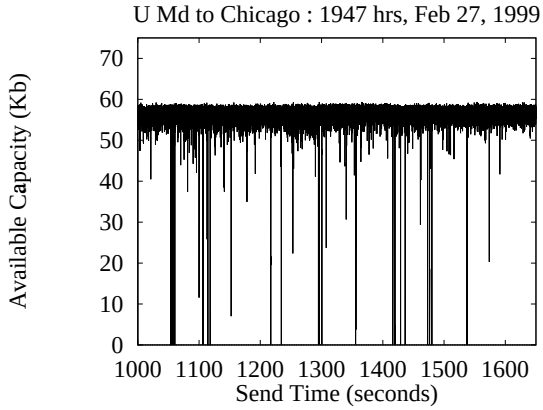

Fig. 10. $\rho\,(22ms, t)$

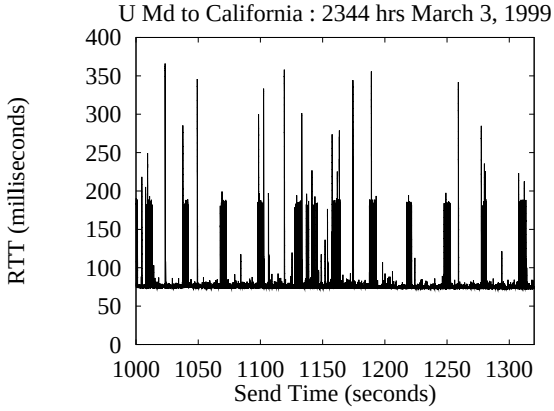

Fig. 11. Round Trip Time

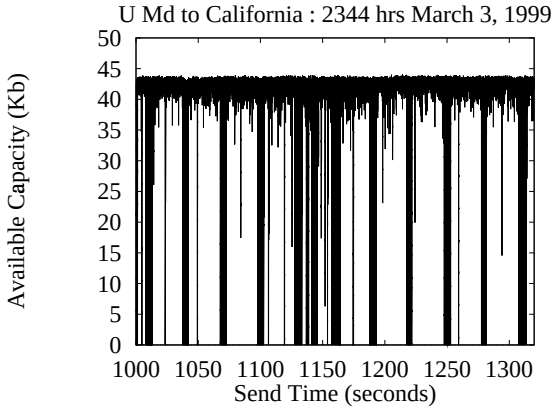

Fig. 12. $\rho\,(75ms, t)$

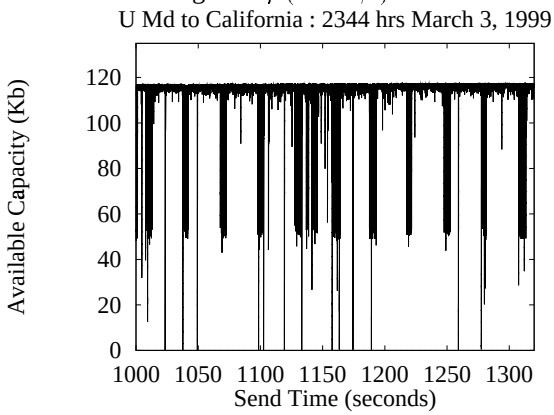

Fig. 13. $\rho\,(200ms, t)$

within about 10 μs. In the observations of the histogram, we note 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 we believe that some statistical analysis of this data might provide a more accurate estimate, the prevalent noise in the measurements make us doubt the significance of an accuracy below 10 μs.

To see that this technique, indeed, provides better results than simple packet pairs, we present estimates made from an experiment conducted between *milhouse.cs.umd.edu* and *icarus.uic.edu* on February 27, 1999. 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, we plot histograms of the packet pair differences at the echo site, in ranges of 5 μs, between 0 and 200 μs. The y-axis gives the normalized frequency of packet pair differences. When we consider all packet pairs (Figure 7), the modal packet pair difference is about 65 μs, where nearly 20% of the packet pair differences lie. In Figure 8, we consider only those packets, whose RTTs were within 0.5 ms of the minimum of 20.05 ms. In this case, the modal packet pair difference is higher at about 100 μs, with again about 20% of the packet pair differences.

We do see packet pair differences higher than 200 μs, but the normalized frequency of such packet pair differences are negligible.

VII. Results

Finally, we present the estimates that we derive using the given techniques. We have performed experiments using our techniques 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 we describe results from two of our recent experi-

ments to a couple of sites in the USA. 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 10 plot the RTT and available capacity estimate for this experiment. The minimum RTT was 20.05 ms.

A simple observation that can be made in these plots is the location of the RTT peaks in Figure 9. These are periods of high encountered delay, and using our 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 10), goes to zero, as would be expected. This is an example of how our 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 our model handles different transit delay requirements, we plot available capacities with $\Delta = 75$ ms in Figure 12 and for $\Delta = 200$ ms in Figure 13, for the same experiment. Figure 11 shows the RTT for the experiment. It can be noted that for the periods when the RTT peaks $\sim$ 185 ms, the estimate of available capacity, with $\Delta = 75$ ms, goes to zero in Figure 12. For the same RTT peaks of ~ 185 ms, the estimate of available capacity is ~ 50 Kb, for $\Delta = 200$ ms in Figure 13. Only for the RTT peaks greater than 200 ms, does the available capacity in Figure 13 go to zero.

VIII. Conclusions and Future Work

In this paper, we first stated our definition of available capacity visible to a connection endpoint. We described our network connection model and provided analytical expressions to calculate this capacity. We also described mechanisms to calculate the service time of the bottleneck router in a connection and validate improved performance over packet pair techniques proposed previously. Through the use of the NetDyn tool, we computed the available capacity, and presented results from experiments performed over the Internet.

Although, the amount of active measurement traffic that we generate is only about 25 Kbps, this technique still would not scale for Internet-wide deployment by users. An important extension of this work would be to study its applicability in passive measurement schemes.

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, in this regard.

References

[Pos 80] J. Postel. User Datagram Protocol. RFC 768, 1980.

[Pos 81a] J. Postel. Internet Protocol. RFC 791, 1981.

[Pos 81b] J. Postel. Internet Control Message Protocol. RFC 792, 1981.

[Pos 81c] J. Postel. Transmission Control Protocol. RFC 793, 1981.

[Ja 88] V. Jacobson. Congestion Avoidance and Control. Proceedings of Sigcomm, 1988.

[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.

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

[Pa 97b] V. Paxson. End-to-End Internet Packet Dynamics, Proceedings of Sigcomm, 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.

[Pa 98] V. Paxson. On calculating measurements of packet transit times. Proceedings of Sigmetrics, 1998.

[LaBa 99] K. Lai and M. Baker. Measuring Bandwidth. Proceedings of Infocom, 1999.

[MoSkTo 99] S. B. Moon, P. Skelly and D. Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. Proceedings of Infocom, 1999.

[CAIDA] Cooperative Association for Internet Data Analysis. http://www.caida.org/info.

Appendix

I. Appendix : Proof of Theorem 1

Observation 1: A packet j is buffered at a l-server router, R_i, $\Leftrightarrow d_j^i = d_{j-l}^i + s^i$.

Observation 2: A packet j is not buffered at a router R_i, $\Leftrightarrow d_j^i = a_j^i + s^i = d_j^{i-1} + \delta^i + s^i = d_j^{i-1} + t^i$.
Note, if $d_j^i = d_j^{i-1} + t^i = d_{j-l}^i + s^i$, 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.
   $d_r^i \geq d_{r-l}^i + s^i$ (Observation 3)
   $d_{r-k}^i = d_{r-k-l}^i + s^i$ (Observation 1)
   So, $d_r^i - d_{r-k}^i \geq d_{r-l}^i - d_{r-k-l}^i \geq A$ (hypothesis).

This proves the lemma.

Corollary 1: If $\forall j > 0, d_j^i - d_{j-k}^i \geq A$, then $\forall j > 0, d_j^r - d_{j-k}^r \geq A$, 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 $\forall j > 0, d_j^i - d_{j-k}^i \geq A$. Assume, $\forall j > 0, d_j^{r-1} - d_{j-k}^{r-1} \geq A$. It is needed to show that $\forall j > 0, d_j^r - d_{j-k}^r \geq A$.
For $j - k \leq 0, d_{j-k}^i, d_{j-k}^r = 0$. And $d_j^r \geq d_j^i$, for $r > i$.
So, $d_j^r - d_{j-k}^r \geq d_j^i - d_{j-k}^i \geq A$.
For $j - k > 0$ $a_j^r = d_j^{r-1} + \delta_r$, where δ_r is the transit delay between routers, R_{r-1} and R_r.
Similarly, $a_{j-k}^r = d_{j-k}^{r-1} + \delta_r$.
Hence, $a_j^r - a_{j-k}^r = d_j^{r-1} - d_{j-k}^{r-1} \geq A$. So, $\forall j, d_j^r - d_{j-k}^r \geq A$ (Lemma 2).

Corollary 2: $\forall j > 0, d_j^i - d_{j-l}^i \geq s^{max_l(1,i)}$.

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, $d_j^i \geq d_{j-l}^i, \forall l > 0$, 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 $\forall j, d_j^q - d_{j-l}^q \geq s^q$, (Observation 3) noting that R_q has l servers.
Since, $i > q, \forall j, d_j^i - d_{j-l}^i \geq s^q$, (Corollary 1) i.e., $\forall j, d_j^i - d_{j-l}^i \geq s^{max_l(1,i)}$.

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.

$\forall j, d_j^i - d_{j-l}^i \geq s^i$, (Observation 3).
$a_j^{i+1} = d_j^i + \delta^i$ and $a_{j-l}^{i+1} = d_{j-l}^i + \delta^i$
With $k > i$, $d_j^k - d_{j-l}^k \geq s^i$ (Corollary 2).
i.e., $d_j^k \geq d_{j-l}^k + s^i > d_{j-l}^k + s^k$.
However, if some packet, q is buffered at router, R_k, then $d_q^k = d_{q-l}^k + s^k$, 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.