Packet Trains—Measurements and a New Model for Computer Network Traffic

RAJ JAIN, SENIOR MEMBER, IEEE, AND SHAWN A. ROUTHIER

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

Abstract—Traffic measurements on a ring local area computer network at the Massachusetts Institute of Technology are presented. The analysis of the arrival pattern shows that the arrival processes are neither Poisson nor compound Poisson. An alternative model called "packet train" is proposed.

In the train model, the traffic on the network consists of a number of packet streams between various pairs of nodes on the network. Each node-pair stream (or node-pair process, as we call them) consists of a number of trains. Each train consists of a number of packets (or cars) going in either direction (from node A to B or from node B to A). The intercar gap is large (compared to packet transmission time) and random. The intertrain time is even larger. The Poisson and the compound Poisson arrivals are shown to be special cases of the train arrival model.

Another important observation is that the packet arrivals exhibit a "source locality." If a packet is seen on the network going from A to B, the probability of the next packet going from A to B or from B to A is very high.

Implications of the train arrivals and of source locality on the design of bridges, gateways, and reservation protocols are discussed. A number of open problems requiring development of analysis techniques for systems with train arrival processes are also described.

Introduction

MANY system design problems are essentially resource management problems which can be done more efficiently if the resource requirements can be accurately predicted. In computer networks, a common assumption is that packet arrivals are independent and unpredictable. However, if we could somehow predict something about future arrivals, we could design better packet-handling strategies or at least implement current ones in a more effective manner.

The problem of predicting packet arrivals in a computer communication system is analogous to that of predicting memory page references in a paged memory computer. If the page references are assumed to be independent, analysis would indicate that a random page replacement strategy is as good as any other. On the other hand, actual page references have been observed to be correlated such that the probability of a page being referenced decreases as the time to its previous reference increases. This observation leads to a *least recently used* (LRU) policy, which is at present the most commonly used page replacement policy. Similarly, if we find that the probabilities of packets going to different destinations in a computer network are not the same, it may lead us to use different strategies than if we assume the probabilities to be the same.

In designing computer networks, we have a choice of at least two models of packet arrival patterns: a "car model," which assumes independent single packet arrivals, and a "train model," which assumes that a group of packets travel together. A protocol design based on the assumption of a train arrival would be quite different from one based on independent arrivals. In the car model, each car has to decide at each intersection (or exit) whether to take that exit or not. Even if all packets are going to one destination, they each make an independent decision, which may result in unnecessary overhead. The overhead is apparent on computer networks in which all intermediate nodes (routers, gateways, or bridges [8]) must make this decision for all packets, therefore resulting in long queues at each node. In a train model, on the other hand, the locomotive (the first packet of the train) may make the routing decision, and all other packets may follow it.

The size of data objects being transported over computer networks has increased substantially compared to the increase in packet sizes. Packet sizes have generally been limited by the buffer sizes and by the need to be compatible with old networks. Transfer of a graphic screen may involve on the order of two million bits. This increase in information size means that most communications involve a train of packets, not just one packet. This fact precipitated a closer look into actual traffic on the networks to determine whether there is a "train" effect. This paper is a result of that inspection.

A number of studies of LAN traffic exist in the literature [9], [6], [13], [17], [23], [27]. Also, a number of authors have discussed issues related to measurement of LAN traffic [3], [1]. For wide area networks traffic studies, we refer readers to an excellent survey by Tobagi *et al.* [29]. The measurements presented in this paper differ from other measurements in that we are looking for burstiness, predictability, locality, and correlations in the traffic.

In the next section, we describe the network on which the measurements were done. We then describe the commonly used arrival patterns such as Poisson and com-

Manuscript received December 2, 1985; revised March 25, 1986. This work was supported in part by the Defense Advanced Research Projects Agency of the U.S. Department of Defense and was monitored by the Office of Naval Research under Contract N00014-83-K-0125.

R. Jain is with Digital Equipment Corporation, Littleton, MA 01460, and was on leave at the Massachusetts Institute of Technology, Cambridge, MA 02139.

S. Routhier was with the Massachusetts Institute of Technology, Cambridge, MA 02139. He is now with Prime Computer, Inc., Framingham, MA 01703.

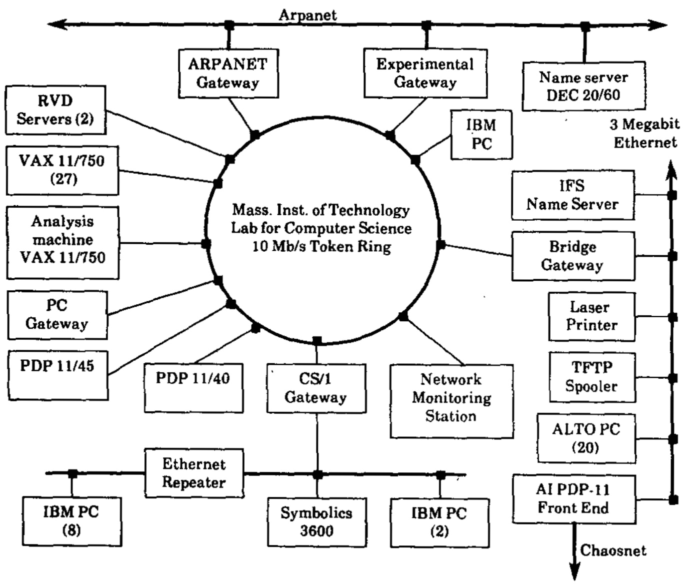

Fig. 1. The M.I.T. LCS 10 Mbit/s ring network.

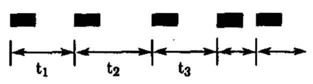

Fig. 2. The Poisson model treats each packet as a black box.

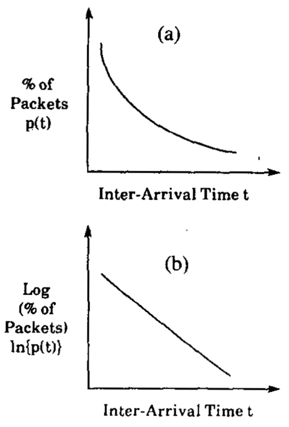

Fig. 3. The histogram and log histogram of interarrival times of a Poisson process.

pound Poisson and introduce the concept of packet train arrivals. In the third section, we present an analysis of actual data that shows the existence of the train phenomenon. Finally, we present many applications of the train concept in the design and implementation of protocols.

Environment

All the measurements presented in this paper were done on a token ring network [4] at the Massachusetts Institute of Technology (M.I.T.) Laboratory for Computer Science (LCS). The ring configuration is shown in Fig. 1. The ring operates at 10 Mbits/s, connecting 33 computers and 5 gateways on 4 floors of a building. It has a star shape with one wire center on each of the four floors [22]. It has two gateways to ARPAnet, one gateway to a 3 Mbit/s Ethernet™, another gateway to a 10 Mbit/s Ethernet™, and a dial-up gateway used by personal computers. There are three disk servers, which are used by many time-sharing VAX™ systems that use a *remote virtual disk* (RVD) protocol [7].

The predominantly used higher-level protocols on the ring include DARPA Internet's *transmission control protocol* (TCP) [19] used mostly for remote terminal (TELNET) [5] applications, *remote virtual disk* (RVD) protocol used by the disk servers, and *user datagram protocol* (UDP) [18] used generally in a request–response mode by network inquiries and for a file transfer protocol named *trivial file transfer protocol* (TFTP) [24].

The traffic on the ring is continuously monitored by a station which is described in [6]. The monitor extracts the first 16 bytes of each packet's header, which contains source, destination, packet length, and protocol type. Since the monitoring station does not have the power necessary for detailed analysis, it combines headers of 67 successive packets and prepares a *monitor packet*, which is sent over the ring to a more powerful analysis machine. The monitor packets are sequentially numbered, and therefore the analysis machine can recognize any packets that are sent from the monitoring station and lost on the way. We lose about 1–9 percent of the monitor packets. This is because the monitor uses a simple protocol without retransmissions, it sends most of its packets when the network is busiest, and the receiver is a time-sharing Unix™ system that may be busy with other tasks, and its buffers may overflow. This introduces some discrepancies in the numbers presented, but this should not change any of our conclusions since they do not depend upon exact numeric values.

We analyzed the traffic on the ring at different occasions in many different ways. The analysis presented here is for data collected during the week of December 10–17, 1984. During that week, the ring carried a total of 11 million packets.

Models of Packet Arrival

Model 1: Poisson Arrival Model

The most commonly used model for arrivals in analytical modeling is "Poisson arrival" [14], [21], [28]. In this model, the interarrival times t_i (between arrival of packets i and $i + 1$; see Fig. 2) have the following characteristics.

1) *They are independent.*

2) *They are exponentially distributed, i.e., probability density function*

$$p(t) = \lambda \exp(-\lambda t).$$

If we plot a histogram of the interarrival times, it would be an exponentially decreasing function, as shown in Fig. 3(a). There are many statistical techniques to verify if a particular arrival process is Poisson. One simple way to visually verify whether the interarrival times are exponentially distributed (the second condition above) is to plot a log histogram, as is shown in Fig. 3(b). Since the prob-

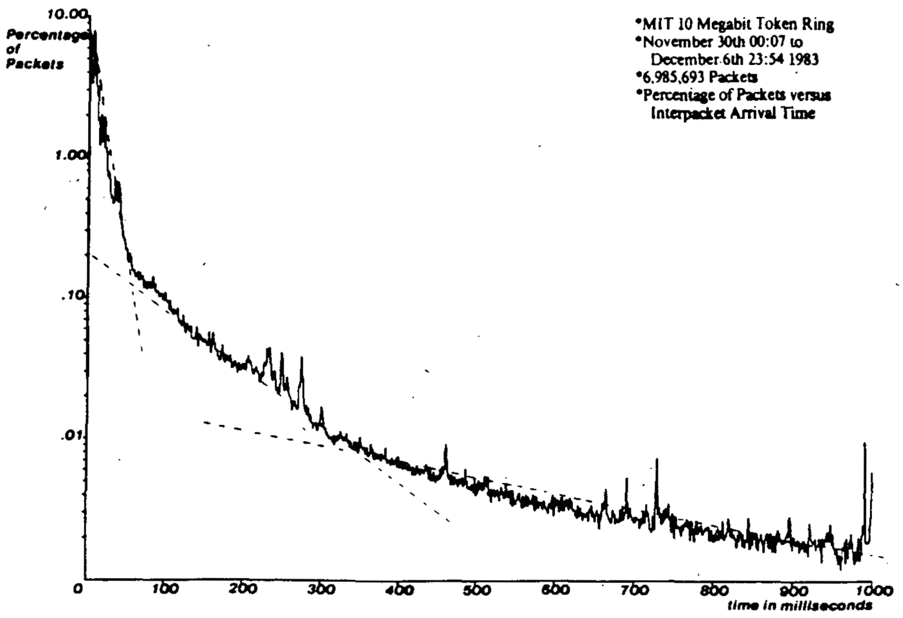

Fig. 4. The log histogram of interarrival times of packets as measured.

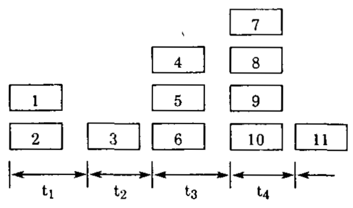

Fig. 5. A compound Poisson arrival process consists of a sequence of bursts arriving in a Poisson manner. Each burst (batch) consists of several simultaneous arrivals.

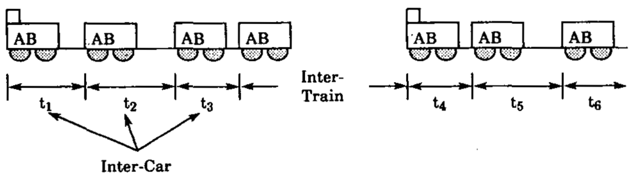

Fig. 6. The packet train model consists of a sequence of packets traveling between a given pair of nodes. The intercar interval is much smaller compared to the intertrain interval.

ability is an exponential function, the logarithm of the probabilities would be a linear function:

$$\text{Log}\,\{p(t)\} = \log\,\{\lambda\} - \lambda t.$$

Another property of the exponential distribution is that its coefficient of variation (the ratio of standard deviation to the mean) is one.

In order to verify if the interarrival times are exponentially distributed, Feldmeier [6] plotted a log histogram of an earlier week's activity, which is reproduced in Fig. 4. It is obvious from this figure that the log histogram is not linear. Rather, it consists of three distinct straight-line segments. This deviation from the Poisson is what eventually led us to the research presented in this paper.

Model 2: Compound Poisson Arrivals

An extension of the Poisson arrivals is the compound Poisson arrival process [10], [15], [16]. As shown in Fig. 5, in this model the arrivals occur in batches. The batch arrival process is Poisson in the sense that the interbatch times are independent and exponentially distributed. The batch size is random. If the batch size is assumed to be geometric, it is possible to derive simple analytical results for the process.

On a log histogram, compound Poisson arrivals would result in a straight line with a spike near the origin. From Fig. 4, we see that this is not the case. Our measurements also confirmed that simultaneous (or back-to-back) arrivals are rare. This is because the time required to prepare packets (on the order of milliseconds) is generally much longer than the time required tro transmit the packet on the network (on the order of 100 μs). Furthermore, most network nodes are shared by nonnetwork activities. This makes the time between successive packets from a single node large as well as random.

Model 3: Train Arrival Process

Our measurements (presented later in this paper) led us to a new model of arrival, which we named the *train model*, shown in Fig. 6. Imagine that every node on the network is connected to every other node via a railroad track (sometimes called a logical link). Consider the track between two nodes A and B. All packets on the track are flowing either from A to B or from B to A. A train consists of packets flowing on this track with the intercar time between them being smaller than a specified maximum, referred to as the *maximum allowed intercar gap* (MAIG). If no packets are seen on the track for MAIG time units, the previous train is declared to have ended and the next packet is declared to be the locomotive (first car) of the next train. The intertrain time is defined as the time between the last packet of a train and the locomotive of the next train.

Notice that the train packets flow in both directions and that there may be several different trains traveling simultaneously on the network. For example, in between packets of a train traveling between nodes A and B, there may be seen packets of another train traveling between nodes C and D.

Before coming to the above definition of trains, we experimented with other possible models, such as: *source trains*—the train of packets starting from a given source, *destination trains*—the train of packets destined to a given node, etc. However, the analysis showed that these alternatives do not characterize the traffic well. This is because the sequence of packets going in one direction on a track is closely related to the sequence going in the reverse direction on the same track. In fact, in many protocols (e.g., in request-response protocols), given the sequence of packets going in one direction, it is possible to predict the sequence of packets going in the reverse direction.

The Poisson as well as compound Poisson models treat packets as black boxes. They do not distinguish between packets coming from different sources or those going to different destinations. They therefore lose some information which is easily available at the network layer. By dividing the packets into different tracks, we are trying to use this information. An analogous example is the problem of predicting employee arrival times. If we stand at the gate and measure interarrival times of employees, we may conclude that the successive interarrival times are independent and exponentially distributed; we therefore cannot predict arrivals. On the other hand, if we note the badge numbers and their arrival times, we can accurately predict arrival times for the next day, as people generally arrive around the same time each day. Ignoring the source and destination of packets on the networks is like ignoring the *badge numbers*. The packets on different tracks are independent, yet packets on the same track may be correlated.

Analyses of Measured Traffic

Analysis 1: Packets as Black Boxes

The first analysis that one can perform on a stream of packets is to treat them as black boxes. We do not look into the packet header fields or distinguish packets based on their source or destination. The time intervals between successive packets form a *time series*, whose mean, standard deviation, and coefficient of variation can be calculated. Another important quantity for a time series is its *autocorrelation function* (ACF). This function shows the relationship of an element of a time series, say, t_i, with a previous element, say, t_{i-k}. The covariance between t_i and t_{i-k} normalized by the variance of t gives the ACF at lag k [2]. The ACF always lies between -1 and $+1$. A negative ACF implies an *inverse relationship*, i.e., when t_{i-k} goes up, t_i is expected to go down, and when t_{i-k} goes down, t_i is expected to go up. A positive ACF implies a *direct relationship*, i.e., if t_{i-k} is high, t_i is also expected to be high. A zero ACF indicates no relationship, or statistical independence.

TABLE I
ANALYSIS WITH PACKETS AS BLACK BOXES

Number of Intervals	Mean (ms)	ACF 1	ACF 2	ACF 3	Stand. Dev.	Coeff. Var.
11,022,088	65.8	0.015	0.046	0.043	2835.3	43.1

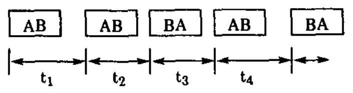

Fig. 7. A node-pair process consists of all packets traveling between a pair of nodes.

If the packet arrivals on the network were a Poisson process, the interarrival times would have a coefficient of variation of one, and the ACF would be zero at all lags.

Table I shows the results of such an analysis for our data. From the table, we see that the ACF is small, which indicates that successive time intervals are independent. However, the coefficient of variation is very high compared to unity, leading us to conclude that interarrival times are not exponentially distributed. Hence, *packet arrivals are not a Poisson process.*

Analysis 2: Node-Pair Processes

Low autocorrelation as well as high variability are both bad news to a network designer, as they both reduce predictability. Designers prefer high predictability because it helps improve the efficiency of resource management. If one could exactly determine the future resource requirements, the resources could be assigned optimally. Therefore, in analyzing the packet stream, we started looking for ways to increase the *predictability.*

One alternative that comes to mind is to divide the stream into several node-pair processes, as is shown in Fig. 7. For each pair of nodes, say (A, B), on the network, all packets traveling between A and B form a time series (process) which can be analyzed separately. For example, in a network with four nodes A, B, C, and D, there would be a maximum of six node-pair processes, i.e., those belonging to AB, AC, AD, BC, BD, and CD. Given n nodes on the network, the packet stream can be divided into $n!(n-1)!/2$ node-pair processes. However, not every pair of nodes communicates and therefore the actually observed number of node-pair processes is rather low. We divided the packet stream into individual node-pair processes, and we analyzed each node-pair process in exactly the same manner as in the previous section. We computed mean, standard deviation, coefficient of variation, and ACF at lags 1, 2, and 3. In addition, 90 percentiles of the time series were calculated.

The results for the ten most active node-pair processes are shown in Table II. The first two columns of the table list node identifiers of the two nodes of the pair. The third column is the number of interpacket intervals for packets that were sent between the nodes. The remaining columns give statistics of the interarrival times (measured in milliseconds). From the table, we see that the ACF is still

TABLE II
ANALYSIS WITH ARRIVALS DIVIDED INTO SOURCE-DESTINATION NODE-PAIR PROCESSES

Nodes		Number of	Mean	90-	Coeff.	ACF		
1	2	Intervals	(ms)	Perc*	Var.	1	2	3
68	86	1,320,555	123.9	70	60.0	0.1	0.1	0.0
4	9	1,275,500	381.3	435	22.3	0.0	0.1	0.1
9	86	1,258,595	28.1	25	85.1	0.0	0.2	0.0
4	75	981,888	187.4	80	63.7	0.0	0.0	0.0
6	65	427,892	447.2	500+	25.5	0.1	0.1	0.0
67	68	412,316	317.5	230	47.7	0.0	0.0	0.0
65	87	397,095	733.2	85	32.9	0.0	0.1	0.0
75	86	395,635	25.9	45	27.9	0.0	0.0	0.0
68	87	349,953	988.3	275	27.2	0.0	0.1	0.0
6	66	294,332	459.5	500+	22.1	0.0	0.0	0.0
Overall		10,850,688	1411.5	245	20.6			

[a]If a 90 percentile value is greater than 500 ms, it is shown as 500+.

TABLE III
ANALYSIS WITH NODE-PAIR PROCESSES DIVIDED INTO TRAINS (WITH MAIG = 500 ms)

Nodes		Number of	Mean	90-	Coeff.	ACF		
1	2	Intervals	(ms)	Perc.	Var.	1	2	3
68	86	1,315,298	31.0	70	1.6	0.0	-0.1	0.0
9	86	1,257,178	16.2	25	1.9	0.2	0.0	0.0
4	9	1,177,654	85.3	255	1.3	0.1	0.3	0.2
4	75	956,195	30.1	65	2.0	0.0	0.1	0.1
67	68	401,008	59.1	195	1.6	0.1	0.0	0.0
75	86	395,174	22.1	40	1.5	0.1	0.0	0.0
65	87	388,007	35.0	75	1.4	0.1	0.1	0.0
6	65	382,232	84.5	200	1.1	-0.1	0.1	0.0
68	87	332,011	55.4	155	1.6	0.0	0.0	-0.1
6	66	255,689	87.5	205	1.1	-0.2	0.2	-0.1
Inter-Car		10,228,405	51.1		1.6			
Inter-Train		622,283	23,773.0		5.0			

small, indicating negligible correlation. The coefficient of variation for some processes is more than that in Table I, and for others it is less than that in Table I. Overall, the coefficient of variation is now smaller than that in Table I. However, it is still high compared to that for a Poisson process. The node-pair processes are therefore neither Poisson nor any more predictable than the packet stream as a whole.

The 90 percentile column in Table II provides some new information. Notice that for most node-pair processes, the 90 percentile is lower than the mean. This implies that the distribution of interarrival times is highly *skewed to the right* (i.e., with a long tail on the right). Most packets arrive within a short interval of the previous arrival. However, in a few cases, there is a considerable delay leading to a *tail* in the distribution. This raises the mean above the 90 percentile value. This observation leads us to the train model discussed next.

Analysis 3: Train Model

Each node-pair process can be divided into a number of trains by specifying a MAIG. We experimented with a few different MAIG values. Table III shows the analysis with a MAIG of 500 ms. Although the choice of MAIG does impact numerical results, the final conclusions about traffic characteristics remain unchanged. We prefer the chosen value primarily because 90 percentiles for most node-pair processes in our initial measurements were well below this value.

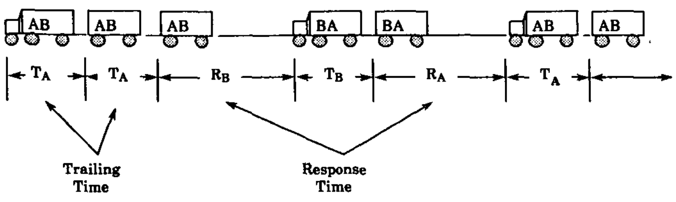

Fig. 8. A train can be subdivided into several tandem trailers. Each tandem trailer consists of several packets going in the same direction.

There are a number of observations that one can make from Table III. First, the coefficient of variation is very near one. Ninety percentiles are two to three times the mean value. Both these observations lead us to believe that the interpacket time in a train is exponentially distributed. However, the ACF is now generally nonzero. Nonzero correlations indicate that the intercar periods are dependent. One explanation for this is that the network nodes have other tasks going on in parallel with networking activities and the nodes have their busy periods and idle periods. During a busy period, it takes long to send/route a packet, and all intercar intervals are long. During idle periods, all successive intercar intervals are short. Some of the correlations are negative, indicating that sometimes short intervals are followed by large intervals, and vice versa. This happens particularly in the request-response type of protocol, in which there is an alternating sequence of data (which takes a long time to generate) and requests or acknowledgments (which are generated quickly). The bottom two lines of the table indicate that the mean intercar interval is about 51 ms, which is small compared to the mean intertrain interval, which is about 23.7 s. Although the variance in Table III is considerably smaller than that in Tables I and II, it is still far from zero. At this point, we wonder if we can further reduce the variance and increase the correlation. Doing so will help increase the predictability of resource demands and lead to the design of more efficient packet-handling strategies. To see if this can be done, we need to look further into the trains. This leads us to our next model, called the *tandem-trailer model.*

Analysis 4: Tandem-Trailer Model

A packet train consists of packets going in both directions. A train between A and B, as shown in Fig. 8, for example, consists of one or more AB packets (packets going from A to B) followed by one or more BA packets, followed again by more AB packets, and so on. A sequence of successive packets going from one source to the same destination is called a tandem trailer. The train consists of several tandem-trailer trucks going in alternate directions. A tandem trailer may consist of *segments* of the same user message. It takes some time for a node to generate the data initially; they may have to be read from a disk, for example. However, once it has read a *chunk* of say eight blocks, it can send successive packets (of one block each, say) rather quickly. The leading packet (first packet) of the truck, therefore, may take longer than the trailing packets in the truck. Actually, the leading packets

TABLE IV
ANALYSIS WITH TRAINS DIVIDED INTO TANDEM TRAILERS

Prot.	Node # A	Node # B	Trailing Time A	Trailing Time B	Response Time A	Response Time B	Truck Size A	Truck Size B	Total Intervals
RVD	68	86	11.3	16.0	88.0	53.1	3.8	2.6	1,314,272
RVD	65	87	10.1	13.9	58.0	57.4	1.8	2.2	387,747
RVD	67	68	14.8	47.5	91.6	189.9	3.8	3.1	380,766
RVD	68	87	12.6	10.0	169.4	66.3	1.3	2.9	331,624
RVD Overall			14.0		74.4		2.4		3,999,668
UDP	4	5			220.3	26.4	1.0	1.0	31,069
UDP	5	75			51.0	27.0	1.0	1.0	9,500
UDP	5	74			51.0	23.0	1.0	1.0	8,392
UDP	6	65			129.1	31.1	1.0	1.0	7,096
UDP Overall					51.3		1.1		101,559
ICMP	195	195	92.1				94759.0		94,759
ICMP	4	67	141.1				20308.0	7.0	20,315
ICMP	4	86	44.6				6883.0		6,883
ICMP	6	65	140.0				2818.0	1.0	2,819
ICMP Overall			99.6				2679.5		128,615
TCP	9	86	7.1		22.9	11.5	2.7	1.2	1,256,940
TCP	4	9	87.0	55.2	148.9	53.0	1.5	2.0	1,176,827
TCP	4	75		31.2	51.0	7.6	1.0	1.3	955,656
TCP	75	86	14.9	83.4	23.7	12.2	1.2	1.2	395,122
TCP Overall			53.0		61.6		1.5		5,965,835
All Overall			34.2		65.2		1.8		10,202,774

are of two types: simple responses such as acks, which required neither I/O nor any significant computation, and complex responses (such as long user messages), which may require I/O, computation, and possibly process schedulings. The interarrival times before a simple response, complex response, and trailing packets would be expected to be small, large, and medium, respectively.

It is obvious that the validity of the tandem-trailer model depends heavily on the networking protocol. Some protocols are purely request–response type with packets going in one direction that are data packets (responses) and packets going in the other direction that are either acknowledgments or requests for more data. To identify tandem trailers in the network traffic, we have to look in the protocol field as well and analyze separate packets of different protocols.

Table IV shows the results for four different protocols. The four most active node pairs are shown for each of the protocols. The protocols are RVD, UDP, *internet control message protocol* (ICMP), and TCP. For each node pair AB, the average time between adjacent A-to-B packets is shown under the column marked "Trailing Time A." The average time between adjacent B-to-A packets is shown in the next column. The "Response Time A" column shows the average time between a packet going from B to A and the adjacent A-to-B response. The truck size is the number of packets going in one direction before a packet is seen in the opposite direction.

Table IV shows that the average truck size in RVD is 2.4, i.e., every response is followed by one or two trailers. The trailing time is much shorter than the response time in each case.

UDP is DARPA Internet's transport layer protocol used generally for the request–response type of application. In these applications, every packet traveling in one direction is followed by a packet traveling in the other direction. The average truck size is close to one; that is, there are no trailers.

ICMP is another protocol from the DARPA Internet suite. Periodically, each node sends a message to a control node. All packets flow in one direction only. There are no responses or leaders. Every packet is a trailer. The truck size is large.

TCP is also from the DARPA Internet suite. It is another transport layer protocol. This protocol is used by many different applications. The average truck size is 1.5; that is, about one-half of the packets are followed by a second packet which is going in the same direction. Both simple and complex responses can be seen in the table.

Overall, the truck size is 1.8. The trailing times are about half of the response time. The tandem-trailer model seems to be valid, but the variance is still not zero. The predictability is basically the same as with the train model. Although the truck model gives a little more understanding of the underlying phenomenon, it still does not give understanding sufficient for use in protocol design or implementation.

Analysis 5: Source Locality

One additional phenomenon that we observed in our traffic was that of source locality. We have borrowed the term *locality* from the field of memory reference modeling. It is well known that successive references to memory have a tendency to cluster at the *most recently referenced page*. We observed a similar phenomenon in the network traffic. We found that successive packets have a tendency to belong to the same train. We termed this phenomenon source locality.

Most analyses on network modeling assume a uniform probability of a packet coming from all sources on the network. Under this assumption, given a network with m nodes,

$$\text{Probability}\{\text{the next packet will come from a given source } i\} = 1/m,$$

$$\text{Probability}\{\text{the next two packets will come from a given source } i\} = 1/m^2,$$

and

$$\text{Probability}\{\text{the next two packets will come from the same source}\} = 1/m.$$

In the packets that we monitored, we found that 21 nodes either sent or received more than 100 000 packets. Even if we ignore other nodes, which were not as active, the probability that two packets will come from the same source would equal 1/21 or 5 percent. However, we found that the probability of a packet going from A to B being followed by another packet going from A to B is 29 percent, and that of an A-to-B packet being followed by a B-to-A packet is 31 percent. Approximately one-third of the packets followed a packet from the same source, and another third followed packets from the destination of the previous source. This happened when we did not break the traffic into different node pairs. This shows that the assumption of uniform distribution does not represent the real-world traffic.

High source locality shows that the trains from different source–destination pairs do not overlap much. Of course, the amount of train overlap depends on the total load on the network. During periods of high utilization, one would expect a higher overlap and less source locality. During periods of low utilization, there is lower overlap and a high source locality. The networks are designed for peak loads, which occur rarely, and therefore the network utilization is generally low. The measurements were done several times. Each time we noticed the same phenomenon, including at peak periods, such as just before the final exams when all term papers and theses were due.

For bulk data transfer, Zwaenepoel [30] compared a number of protocols and concluded that the best protocol was a *blast* protocol, in which the source blasts the network with a large number of packets going to the same destination. If such protocols become common, the source locality will be even more pronounced in the future.

Applications of the Train Model

In this section, we describe a few potential applications of our findings to protocol modeling/analysis, protocol implementation, and protocol design. We present only a brief discussion of these applications and show that the results based on the packet train model would be quite different from those based on traditional assumptions. Solving systems models with train arrivals is currently an open problem.

Application 1: Protocol Modeling

The analysis has shown that an appropriate model of packets on a network is a train model. The traditional Poisson model has only one parameter—*the mean interarrival time*, which is a cumulative result of many underlying phenomena. The train model, on the other hand, has many parameters, each of which explains a different phenomenon in the network. The intertrain time depends upon how often users transfer data objects. The parameter depends upon the user behavior. The intercar time does not depend on user behavior, but it depends solely on the system (hardware/firmware/software) and the protocols. The train size is related to data object sizes. Given a distribution of object sizes, one can come up with a distribution of train sizes. The tandem-trailer model provides a yet more detailed insight. The average size of the truck is related to the flow control window sizes used in the protocols. The response time may or may not include I/O time, depending upon whether or not it is a simple or complex response. It may therefore depend upon the characteristics of the I/O device. The trailer time, on the other hand, does not depend upon the I/O device, but is rather a function of the protocol and system characteristics.

An analog of this problem happens in the modeling of time-sharing systems. The average service time at the system (in the machine-repairman model) was the only parameter that the initial models of time-sharing systems had. This was later replaced by a more detailed (central server) model with explicit modeling of time spent at disks, CPU, paging device, etc. The train model provides a more detailed understanding of user and system effects in a networked environment. The measured train parameters are summarized in Table V. Exact numerical values of interarrival times (e.g., 51 ms) are expected to vary significantly for different systems, media, protocol implementations, and load levels. It is their relative values (orders of magnitude difference) that are important to the model.

TABLE V
Measured Train Characteristics

Inter-Car Time:		
Mean	=	51.1 ms
C.O.V.	=	1.6
ACF(1)	=	0.2
ACF(2)	=	0.3
ACF(3)	=	0.2
Inter-Train Time:		
Mean	=	23.8 sec
C.O.V.	=	5.0
Number of Cars/Train	=	17.4
Number of Cars/Truck	=	1.8

Some idea of probability distribution functions, e.g., the distribution of train sizes or the distribution of interarrival times, would be useful for modeling purposes. Unfortunately, at the time of this study we did not have facilities to measure them. Efforts are currently underway to study these distributions.

Application 2: Path Caching in Protocol Implementation

The present analysis has shown that successive packets have a source locality. Given a packet, we can predict with high probability that the next packet will be destined either to the destination or the source of the previous packet. Normally, a network has several thousand nodes. Finding the link on which to forward a packet requires sophisticated table search and hashing procedures at each node. The existence of source locality indicates that considerable savings in table lookup can be obtained by simply saving (caching) the table entry for the last packet. The overhead can be further reduced by prefetching the table entry for the source and the destination of a packet. Thus, there is a high probability of having the table entry in our cache even before the next packet arrives. Even a two-entry cache would give an approximately 60 percent hit ratio.

Application 3: Number of Buffers in Routers/Gateways/Bridges

A key parameter in network design is the number of buffers required at intermediate nodes. Any node connecting a high-speed network with a low-speed network tends to have a queue of packets to be forwarded to the low-speed network. Queueing theorists often use an

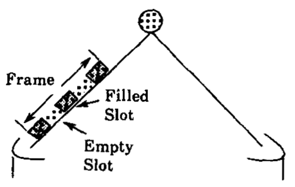

Fig. 9. The reservation switching as used on satellites.

*M/M/*1 model for the node to determine the number of buffers that are required to make the probability of buffer overflow (probability of losing a packet) sufficiently small. The result is based on the average interarrival time and average service time only. The train model indicates that the buffers would also depend on the train size. Also, the trains may create more congestion than predicted by an *M/M/*1 model.

Application 4: Dynamic Circuit Management

A network path using a telephone connection is a dynamic circuit, such that the circuit is established only for the duration of the transfer. To minimize costs, it ought to be closed as soon as we are sure that no immediate traffic is expected to arrive. In network architectures with connectionless orientation (e.g., DNA and DARPA Internet), the end nodes originate the traffic but do not know the path a packet takes. The intermediate node originating the telephone call has to *dynamically* open and close the circuit based on the traffic. With the Poisson model, the packet arrivals are independent, and the fact that we have not seen any packet for the last 1 h has no effect on the probability of arrival in the next 1 s. With the train model, one could set the cutoff point based on the distribution of intercar arrivals and close the connection when the idle time exceeds this cutoff value. The optimal cutoff point will depend on the train parameters as well as on the tariff structure, e.g., cost of opening and closing a telephone call.

Application 5: Reservation Switching

Reservation switching is a form of switching commonly used in satellite links [11]. As shown in Fig. 9, the time in this switching method is divided into equal size frames, and each frame consists of several slots. An empty slot can be obtained by contention. If a node succeeds in obtaining a slot without collision, the slot that is in the same position in the next frame is reserved for the node. If the node does not use the slot in the next frame, the reservation is cancelled and the slot is again made available to other nodes via contention. Reservation switching is based on the belief that a node sending one packet is very likely to send more packets. This is especially true for voice traffic. Active connections send packets every 20 ms or so.

An example of reservation switching in LAN environments is seen in the Cambridge fast ring [26]. The ring has two types of slots: normal slots and channel slots. Once a normal slot has returned to its source after transmission, it must be passed on empty. Channel slots remain reserved for a source until the source passes it on empty.

The efficiency of reservation switching depends upon the correct selection of the frame size. For example, if the frame size for voice traffic were chosen to be 10 ms, the nodes would not have a packet to send in successive 10 ms intervals, and all reserved slots would go empty. For data traffic, the determination of optimal frame size is not straightforward because the data packets do not arrive in a perfectly regular pattern. Had the train measurements shown a zero variance for intercar intervals, the ideal frame size would be the fixed intercar interval. The network measurements have shown that the intercar intervals are not fixed; nonetheless, the ideal frame size is a function of the intercar interval distribution. Further, the observations show that a packet going from A to B should result in a slot being reserved for packets coming from A *as well as for those coming from B*. The measurements (Table IV) show that the direction changes on the average after 1.8 packets, and therefore, for data traffic, we need *bidirectional reservation* protocols.

Hierarchy of LAN Workload Models

We have presented several models of LAN traffic. The appropriate model depends upon the level of detail desired. The Poisson model is the least detailed model of the traffic. It can represent the traffic with the single parameter of mean interarrival interval. This model treats packets as black boxes in that we do not look into the packets. The next level is that of node-pair processes, which requires separating the traffic into several streams based on the source and destination. The train model is the next level down from node-pair processes. At this point, we further subdivide a node-pair sequence into several trains. A new train starts if a car is not seen for a MAIG interval. Each train consists of several tandem trailers, which is the next level model. Each trailer truck consists of leading packets (responses) and trailing packets. The responses can be simple or complex. The complete hierarchy tree is shown in Fig. 10. As we go down the tree, the variance decreases, skewness decreases, and correlation increases, leading to higher predictability.

Summary

The packet train research has shown that the packet arrival process is neither a Poisson process nor a compound Poisson process. The packet arrivals follow a train model. A train consists of packets traveling in both directions between a given node pair. Although the packets of a train are close to each other, they are too far apart to be considered as simultaneous arrivals. The intercar interval is much smaller than the intertrain interval.

The intertrain time is a user parameter, and it depends upon the frequency with which applications use the network. The intercar interval is a system parameter and depends primarily on the network hardware and software. In Poisson arrival models, these parameters are merged to give a single parameter: *mean interarrival time.*

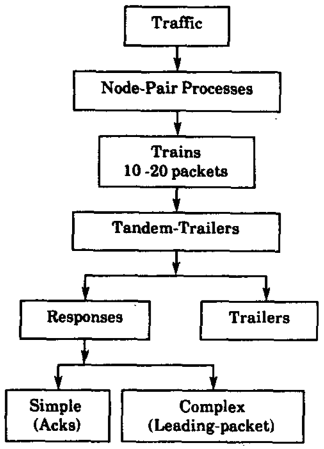

Fig. 10. Hierarchy of LAN workload models.

TABLE VI
SPECIAL CASES OF THE PACKET TRAIN MODEL

Inter-Car Interval Distribution	Inter-Car Interval Auto-Correlation	Inter-Train Interval Distribution	Network Traffic Model
Exponential(λ_1)	Zero	Exponential(λ_1)*	Poisson
Zero	Zero	Exponential	Compound Poisson
Constant	Zero	Exponential	Regular Train
General	Non-Zero	General	Train

*The two exponential distributions have the same mean.

The train model is a generalization of which other models are special cases. If we find ways to analyze network protocols using a train model, the results can be used for other models by simply setting the intercar and intertrain time distributions to values such as are shown in Table VI. Also shown in the table is a simplification of the train model called the *regular train.* In this the trains arrive in a Poisson process and consist of a random number of cars with constant intercar intervals. This type of train, which represents voice traffic, is simple to analyze as well as helpful to network designers since the car arrivals can be easily predicted. We encourage other researchers to attempt modeling with train arrivals and applying these for real-world applications discussed earlier in this paper.

The packets have a high source locality such that, given a packet going from A to B, there is a high probability that the next packet will be going either from A to B or from B to A. The probability of packets from other sources is small.

The lessons learned from the train model can be used to improve protocol analysis, design, and implementation. Cars are good for transporting a small number of people to a large number of independent destinations, whereas trains are good for bulk transfers.

To verify the existence of trains, we need to repeat the analyses in other environments. That will help determine typical values of train parameters for today's traffic. These parameter values are obviously of interest to network analysts. Even without that verification, network designers argue that the amount of information being transported across computer networks is increasing and that they ought to look at ways of making bulk transfers more efficient. We therefore hope that in the near future we will see more *railroad tracks* along with the *highways.*

ACKNOWLEDGMENT

Prof. Jerry Saltzer provided ideas and encouragement during the project. D. Feldmeier designed the monitor and helped maintain it. Dr. D. Clark helped in getting the resources together.

The following are trademarks of Digital Equipment Corporation: DEC, VAX, VAX 11/750, PDP, PDP 11/45, PDP 11/40, and PDP 11. Ethernet™ is a trademark of Xerox Corp. PRONET™ is a trademark of Proteon Associates.

REFERENCES

[1] P. D. Amer, "A measurement center for the NBS local area computer network," *IEEE Trans. Comput.* vol. C-31, pp. 723-729, Aug. 1982.
[2] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting & Control.* San Francisco, CA: Holden-Day, 1970.
[3] I. Chlamtac, "Issues in design and measurement of local area networks," in *Proc. Comput. Measurement Group Conf. CMG XI*, Dec. 1980, pp. 32-34.
[4] D. D. Clark, K. T. Pogran, and D. P. Reed, "An introduction to local area networks," *Proc. IEEE*, vol. 66, pp. 1497-1517, Nov. 1978.
[5] J. Davidson *et al.*, "The ARPANET Telnet protocol: Its purpose, principles, implementation, and impact on host operating system design," in *Proc. 5th Data Commun. Symp.*, Snowbird, UT, Sept. 1977.
[6] D. C. Feldmeier, "Empirical analysis of a token ring network," Mass. Inst. Technol., Lab. Comput. Sci., Cambridge, MA, Tech. Memo. MIT/LCS/TM-254, Jan. 1984.
[7] M. Greenwald, "Remote virtual disk protocol specifications," Mass. Inst. Technol., Lab. Comput. Sci., Cambridge, MA, Tech. Memo., in preparation.
[8] W. R. Hawe, A. Kirby, and B. Stewart, "Transparent interconnection of local networks with bridges," *J. Telecommun. Networks*, vol. 3, no. 2, pp. 116-130, Sept. 1984.
[9] J. Herskovitz, "Evaluating local network performance," *Comput. Sci. Technol.*, pp. 389-396, Oct. 1982.
[10] D. P. Heyman, "An analysis of the carrier-sence multiple-access protocol," *Bell Syst. Tech. J.*, vol. 61, no. 8, pp. 2023-2051, Oct. 1982.
[11] L. Kleinrock, "Performance of distributed multi-access computer-communication systems," in *Proc. IFIP Congr.*, 1977, pp. 547-552.
[12] M. T. Liu, W. Hilal, and B. H. Groomes, "Performance evaluation of channel access protocols for local computer networks," in *Proc. Compcon*, 1982, pp. 417-426.
[13] P. J. Lloyd and R. H. Cole, "Transport protocol performance over concatenated local area and satellite networks," in *Proc. Conf. Data Networks Satellites*, Sept. 1982.
[14] M. Marathe and W. Hawe, "Predicted capacity of Ethernet in a university environment," in *Proc. Southcon '82*, Mar. 1982, pp. 1-10.
[15] B. Meister, "Waiting time in a preemptive resume system with compound-Poisson input," *Comput.*, vol. 25, no. 1, pp. 17-28, 1980.
[16] N. C. Mohanty, "Buffer behavior for batch arrivals and single output in satellite channel," in *Proc. IEEE 1978 Nat. Telecommun. Conf. NTC'78*, Birmingham, AL, Dec. 1978, pt. III, pp. 40.4/1-40.4/3.
[17] D. N. Murray and P. H. Enslow, Jr., "An experimental study of the performance of a local area network," *IEEE Commun. Mag.*, vol. 22, pp. 48-53, Nov. 1984.
[18] J. Postel, "User datagram protocol," ARPA RFC-768, 1980.
[19] J. Postel, Ed., *Transmission Control Protocol—DARPA Internet Program Protocol Specification*, ARPA RFC-793, Sept. 1981.
[20] J. Postel, "Internet control message protocol," ARPA RFC-821, 1982.
[21] K. K. Ramakrishnan and S. K. Tripathi, "A common framework for studying the performance of channel access protocols," in *Proc. Comput. Performance Evaluation Users Group (CPEUG'82)*, Oct. 1982, pp. 365-373.
[22] J. H. Saltzer and K. T. Pogran, "A star-shaped ring network with high maintainability," *Comput. Networks*, vol. 5, no. 4, pp. 239-244, Oct./Nov. 1980.

[23] J. F. Shoch and J. A. Hupp, "Performance of the Ethernet local network," *Commun. ACM*, vol. 23, no. 12, pp. 711-721, Dec. 1980.
[24] K. Sollins, "The TFTP protocol (revision 2)," ARPA RFC-783, June 1981.
[25] J. Sventek *et al.*, "Token ring local area networks: A comparison of experimental and theoretical performance," in *Proc. IEEE Comput. Networking Symp.*, Silver Springs, MD, Dec. 1983, pp. 51-56.
[26] S. Temple, "The Cambridge fast ring," in *Proc. IFIP WG6.4 Int. Workshop Ring Technol.*, Univ. Kent, England, Sept. 1983.
[27] D. Terry and S. Andler, "Experience with measuring performance of local network communications," in *Proc. IEEE Comput. Conf., Compcon'83*, Apr. 1983, pp. 203-205.
[28] F. A. Tobagi and V. B. Hunt, "Performance analysis of carrier sence multiple access with collision detection," in *Proc. LACN Symp.*, May 1979, pp. 217-244.
[29] F. A. Tobagi *et al.*, "Modeling and measurement techniques in packet communication networks," *Proc. IEEE*, vol. 66, pp. 1423-1447, Nov. 1978.
[30] W. Zwaenepoel, "Protocols for large data transfers over local networks," in *Proc. ACM 9th Data Commun. Symp.*, 1985, pp. 22-32.

Raj Jain (S'73-M'78-SM'86) received the B.E. degree from A.P.S. University, Rewa, India, the M.E. degree from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree from Harvard University, Cambridge, MA, in 1972, 1974, and 1978, respectively.

Since 1978 he has been with Digital Equipment Corporation where he has been involved in performance modeling and analysis of a number of computer systems and networks including VAX clusters, DECnet, and Ethernet. Currently, he is a Consulting Engineer in the Distributed Systems Architecture and Performance Group. He spent the 1983-1984 academic year on a sabbatical at the Massachusetts Institute of Technology (M.I.T.) doing research on the performance of networks and local area systems. For two years he also taught a graduate course on computer systems performance techniques at M.I.T. and is writing a textbook on this subject.

Dr. Jain's Ph.D. dissertation entitled "Control Theoretic Formulation of Operating Systems Resource Management Policies" was published by Garland Publishing, Inc., New York, in their "Outstanding Dissertations in the Computer Sciences" series. He is a member of the Association for Computing Machinery.

Shawn A. Routhier received the B.S. degree in computer science and engineering from the Massachusetts Institute of Technology (M.I.T.), Cambridge, MA, in 1983.

Since 1983 he has been a member of the research staff at the Laboratory for Computer Science, M.I.T. His research interests include networks, network security, and network services.