An Analysis of the Congestion Effects of Link Failures in Wide Area Networks

David Tipper, Member, IEEE, Joseph L. Hammond, Member, IEEE, Sandeep Sharma, Archana Ketan, Krishnan Balakrishnan, Member, IEEE, and Sunil Menon, Member, IEEE

Abstract—In this paper, we present the results of a study to determine the effects of link failures on the performance of a network in terms of the occurrence of congestion due to traffic restoration after a failure. The network studied is a virtual circuit based packet switched wide area network. A generic queuing framework is developed to study the effect of failures and the subsequent traffic restoration on network performance. In general, the congestion resulting after a failure is a transient phenomenon. Hence, a numerical methods based nonstationary queuing analysis is conducted in order to quantify the effects of failures in terms of the transient behavior of queue lengths and packet loss probabilities. A bounding relationship is developed whereby a network node can determine whether or not congestion will occur as the result of traffic restoration after a failure.

I. INTRODUCTION

Due to the rapidly growing demand for information transfer across communication networks, the need for reliable communication service has become increasingly important. It has been noted [1] that high exposure business such as airlines, mail order retail, and banking can lose up to 6 million dollars per hour in unrecoverable revenue due to a communication network failure. The potentially drastic effects of network failures have been demonstrated by the recent series of highly publicized network failures [2]–[4]. Such accidents have clearly shown the need for reliable networks that provide service which is robust to failures.

The general areas of communication network reliability and survivability have been under study for some time [5], [10]. Much of this work has focused on the design of networks which will minimize the impact of component failures. While there have been great strides in increasing the reliability of the physical network components, some rate of failure is inevitable. Furthermore, it has been noted by several authors [6]–[9] that as networks evolve to high bandwidth fiber optic based transmission media, the effects of even simple failures, like the loss of a single link, will become more pronounced.

A network failure, such as the loss of a link or a node, can occur due to wide variety of reasons causing service disruptions ranging in length from seconds to weeks. Typical events that cause failures [1], [4] are accidental cable cuts, hardware malfunctions, software errors, natural disasters (e.g., floods, fires, etc.), and human error (e.g., incorrect repair or maintenance, etc.). Since many of the causes of failures are outside the control of network providers, there has recently been increasing interest in the design of survivable networks [1]–[15]. This work has largely focused on planning the network to reduce the impact of link or node failures when they occur. Several techniques have been proposed to minimize the effect of failures; some common ones are multiple ingressing/egressing of users [1], [6], [13], trunk diversity [1], [6], [11], digital cross connect systems [1], [14], and self-healing ring architectures [8], [9]. An excellent discussion of current survivability techniques is given in [10].

The survivability literature primarily concentrates on network design issues, and relatively little literature exists on quantifying the performance impact of a failure. Here we present the results of a study of the effects of link failures on network performance for a virtual circuit, packet switched, wide area network. The network studied is modeled after IBM’s proposed plaNET (formerly PARIS) network architecture [16]–[18], which is a high-speed integrated packet switched network supporting a wide variety of traffic types. After a link failure, several network controls come into play, such as congestion control, cell admission control, and routing, in order to restore the lost traffic. The restored connections can result in a transient period of congestion occurring which can have a significant effect on the quality of network service.

In this paper, a general framework is developed for studying the congestion that can occur in a network as the result of a failure. Nonstationary network analysis techniques [19], [20] are applied within the general framework to model and quantify network performance after link failures.

The remainder of the paper is organized as follows. Section II discusses the effects of failures on network performance by identifying critical issues and developing a generic queuing model framework for analyzing failures. Section III describes the network studied and the development of differential equation models for studying the nonstationary behavior of the queuing models of Section II. Section IV presents a performance analysis of the effect of a link failure pointing out how the magnitude of the congestion after a failure can be estimated using a bounding relationship. In Section V, we summarize the

Manuscript received July 20, 1992; revised February 15, 1993. This work was supported by IBM, Research Triangle Park, NC. This paper was presented in part at IEEE INFOCOM ’93, San Francisco, CA, March 30–April 1, 1993.

D. Tipper, J. L. Hammond, A. Ketan, and K. Balakrishnan are with the Department of Electrical and Computer Engineering, Center for Computer Communication Systems, Clemson University, Clemson, SC 29634-0915.

S. Sharma was with the Department of Electrical and Computer Engineering, Center for Computer Communication Systems, Clemson University, Clemson, SC. He is now with Alcatel Network Systems, Raleigh, NC 27609.

S. Menon was with the Department of Electrical and Computer Engineering, Center for Computer Communication Systems, Clemson University, Clemson, SC. He is now with Codex, Mansfield, MA 02048.

IEEE Log Number 9212482.
II. Modeling the Effects of Failures on Network Performance

Consider the arbitrary packet switched wide area network shown in Fig. 1. We assume that the network uses virtual circuit service to transport packets and source node routing of the virtual circuits as in the plaNET network. In source node routing, each network node maintains a database of the network topology and determines the route through the network for all virtual circuits originating at the node.

A. Impact of Link or Node Failures

The impact of a link failure on the network performance will depend on the complex interaction of several factors, some of which are the location of the failure, the network topology, the network loading, the routing algorithm, the error control procedures, and the congestion control. Note that most of these factors will be specific to the network under study, and determining generic effects of failures on network performance is difficult. However, certain aspects of the behavior of the network can be modeled by studying the general steps of the network in reacting to failures. In the event of link failure, all the virtual circuits using the failed link are disrupted and need to be reconnected if possible. The reconnection of the virtual circuits takes place only after a time delay which consists of the time taken to detect the link failure, plus the time for the affected source nodes to get the relevant information, and the time taken to determine the new route and set up the connection. During the time delay taken to restore a connection, a backlog of packets will accumulate at the source of each virtual circuit. Hence, when these virtual circuits are reconnected, the backlog of packets from all of the virtual circuits must be transmitted, and congestion can occur at various points in the network.

One would expect that congestion control mechanisms would come into play after traffic restoration to prevent the occurrence of congestion. Congestion control schemes proposed for broadband networks are based on either end-to-end windowing schemes [21] or rate based policing mechanisms [22] (e.g., leaky bucket, virtual leaky bucket).

In the event of a failure while using the windowing schemes, the source nodes keep transmitting the packets belonging to the virtual circuits that use the affected link until their corresponding window runs out. These packets are lost and hence must be retransmitted. Further packets, however, accumulate at the access nodes awaiting an opening in the window. In broadband networks, the window size will typically be quite large [21] so as not to interrupt the packet flow when the network is operating in a normal fashion. Thus, windowing schemes may not prevent congestion as the backlog of packets passes through the window of the source if the window is large.

The virtual leaky bucket scheme would not prevent congestion as the packets would be tagged and still admitted to the network. The leaky bucket algorithm would allow a burst of packets at the peak rate to enter the network, and then packets would enter at the leaky bucket token generation rate until the backlog is worked off. It should be noted that the congestion at the network is mainly due to the rerouted virtual circuits needing to simultaneously work off their backlogs. This is, in effect, a correlated burst at the network node, yet the current congestion control schemes are designed to prevent congestion from occurring due to an individual user rather than a group. Thus, the current congestion control schemes may not prevent congestion after a failure.

B. A Framework for Studying Congestion

The occurrence of congestion and its spreading in the network can be studied by noting that a link failure partitions the network nodes into four categories, namely: primary nodes, secondary nodes, tertiary nodes, and quaternary nodes. Primary nodes are those nodes which are the source nodes for virtual circuits that were traversing the link which failed. Thus, the primary nodes are responsible for the restoration of the affected virtual circuits. Secondary nodes are nodes which are relay nodes for the rerouted virtual circuits. Tertiary nodes are defined as those nodes which handle traffic that share a link in common with the rerouted traffic. Quaternary nodes are those nodes whose traffic is not affected by the traffic of the reconnected virtual circuits.

Congestion starts at the primary nodes, which can become directly congested by the backlogged packets at the attached sources. If links at a primary node become heavily congested, the congestion can spread downstream to secondary nodes. Tertiary nodes can become congested in an indirect way caused by increased packet loss rates at primary or secondary node links that the tertiary node traffic shares with the restored traffic.

We examine the modeling of the primary, secondary, and tertiary nodes by studying a representative portion of the network, specifically the region circled in Fig. 1. Fig. 2 is a magnification of this region of the network, illustrating a typical set of network nodes. Note that Fig. 2 also shows a representative loading pattern for the network nodes. Focusing on node 3, one can see from the figure that it is the source node for virtual circuits VCI1–VCI4, whereas it is a relay
node for virtual circuits VC#5 and VC#6. Consider the effect of a link failure, in this case, link 3-2. The link failure results in node 3 being a primary node since it will be responsible for rerouting virtual circuits VC#1–VC#3. In general, the location of the link failure and the network loading patterns at the time of the failure will specify a set of primary nodes. Here we focus on modeling a generic primary node such as node 3.

After a time delay involved in the detection of the failure, a primary node will try to restore its affected virtual circuits. Note that some of the virtual circuits may be blocked from reconnection due to insufficient available bandwidth. However, these blocked circuits will not affect the level of congestion in the network and are ignored here. After the primary nodes have completed traffic restoration, a new loading pattern will exist in the network as illustrated in Fig. 3. In general, after traffic restoration, the traffic at an arbitrary primary node can be grouped into the following three categories: 1) normal traffic—traffic not on the failed link; 2) source rerouted traffic—traffic on the failed link for which this is the source node; and 3) transit rerouted traffic—traffic on failed link for which this is a relay node. In Fig. 4, this categorization of the traffic is illustrated for node 3. From the figure, we have the following groupings for node 3: VC#4 and VC#5 are normal traffic, VC#1–VC#3 are source rerouted traffic, and VC#6 is transit rerouted traffic.

Focusing on a single output link at the primary node, in this case, link 3-4, we can develop the generic queuing model of a primary node output link shown in Fig. 5(a). In Fig. 5(a), $C_{ij}$ denotes the capacity of the output link $i - j$, and the node buffer represents the finite buffer at the output link under consideration. We have assumed that the nodal buffer space is completely partitioned among the output links. The normal traffic has been grouped together and is shown to have aggregate mean packet arrival rate $\lambda_{nt}$. The source rerouted traffic is represented by the individual virtual circuit’s source queues feeding into the nodal buffer. The source queue is modeled as an infinite capacity queue, and the mean packet arrival rate of the $i$th source rerouted traffic is denoted by $\lambda_i$. Also in the figure, the transit rerouted traffic is represented as an aggregate traffic stream with mean arrival rate $\lambda_{trt}$, shown flowing through several queues.

Congestion can occur at the nodal buffer due to the backlog of packets at the source rerouted traffic queues and due to congestion propagating from the primary nodes of the transit rerouted traffic. Since the network is assumed to be source based, a node has direct control in terms of routing, call admission, and congestion control settings only on the traffic originating at that node. Thus, the primary node only has direct control over the source rerouted traffic and that part of the normal traffic originating at the node. Since the source rerouted traffic is the most likely cause of congestion at the node, we simplify the queuing model by combining the transit rerouted traffic and normal traffic into a single traffic stream termed the background traffic with $\lambda_{bg}$ denoting the mean packet arrival.
rate. The resulting queueing model is shown in Fig. 5(b) and can be used to study congestion at the primary node that results from a link failure.

In a fashion similar to the analysis above for the primary node, we can develop queueing models to study the spread of congestion to secondary and tertiary nodes. Observe that the rerouting of the virtual circuits that were on the failed link defines the set of secondary and tertiary nodes. For example, from Fig. 3, one can see that nodes 4 and 2 are secondary nodes, and nodes 5 and 6 are tertiary nodes.

Consider the modeling of secondary nodes, which by definition must be downstream from the primary nodes. They can be studied with queueing models of the form shown in Fig. 6. For example, the queueing model of Fig. 6 arises in the set of network nodes shown in Fig. 3, when we trace the path of virtual circuits VC#1 and VC#3 through nodes 3, 4, and 2. In Fig. 6, at the secondary node output links, all the traffic from sessions other than the rerouted connections are grouped into a common background traffic stream. One can see from the figure that congestion from the primary node due to its backlogged packets can propagate to the secondary node links.

Focusing on tertiary nodes, observe that two types can arise based on their location, namely: 1) downstream of a primary or secondary node, and 2) upstream of a primary or secondary node. Fig. 7 shows typical queueing models of both types of tertiary nodes. The type 1 tertiary node queueing model of Fig. 7(a) can be observed in Fig. 4 by tracing the routes of VC#6 and VC#7 through nodes 3, 4, and 6, with node 6 being the tertiary node. Congestion can occur at a type 1 tertiary node only if the upstream nodes become congested. For example, in Fig. 7(a), if the link at the secondary node becomes severely congested due to restored traffic from the primary node, the background traffic can have a backlog of packets build up, and this backlog when transmitted can congest the tertiary node. Fig. 7(b) illustrates a typical type 2 tertiary node. The queueing model of Fig. 7(b) can be observed in Fig. 4 by tracing the path of VC#5 through nodes 5 and 3 with node 5 being the tertiary node. At a type 2 tertiary node, congestion can occur when the downstream nodes become congested. For example, in Fig. 7(b), if the primary node becomes severely congested, then the tertiary node traffic will suffer an increased packet loss rate at the primary node link, and this can cause a backlog of packets to form in the tertiary node sources which may result
in congestion at the node. Obviously, tertiary nodes of either type are less likely to become congested than secondary or primary nodes.

Lastly, we note that the set of quaternary nodes in the network is defined after traffic restoration is completed. In Fig. 4, only node 1 is a quaternary node. When studying a specific network, the exact form of the queueing models of primary, secondary, and tertiary nodes will be a function of the network topology, loading, and routing algorithm. However, the general primary, secondary, and tertiary models presented here provide an overall framework for the study of failures and associated issues. For example, results on the spread of congestion in sample networks and an analysis of traffic restoration routing algorithms are given in [23] and [26] using the framework proposed here.

III. QUANTIFYING THE CONGESTION EFFECTS OF FAILURES

Utilizing the approach of Section II, one can determine the impact of link failures on the performance of the network. We focus on analyzing congestion at a generic primary node and determining when congestion will occur. The network studied is a simplified variant of IBM’s plaNET. The plaNET network is an experimental high-speed packet switching system for integrated voice, video, and data communications. The plaNET network architecture is based on the previous PARIS system, and both network architectures are described in [16]–[18]. Here we briefly describe the features of plaNET which are germane to our study, along with the simplifying assumptions that were made for our simulation and analytical models.

A. Overview of the Network Model

The plaNET is a connection-oriented network using virtual circuits for transfer of variable length packets. The call admission and routing functions are accomplished in a distributed fashion with each node responsible for the virtual circuits originating at the node. Both the call admission and routing algorithms make use of an “equivalent capacity” concept to determine the parameters of the algorithms. As a call requests entry to the network, it must provide the network access node with the traffic characteristics of the call, specifically the peak rate, mean rate, and burst length. The source node transmits the traffic characteristics into an allocation of bandwidth for the duration of the call. This allocated bandwidth is termed the “equivalent capacity” of the call. The procedure for mapping the call parameters (i.e., peak rate, mean rate, and burst length) into the equivalent capacity is given in [17] and is based on the analysis of a fluid flow model of a traffic source. Here, for the sake of simplicity, we assume the equivalent capacity of a call is the same as its mean rate. After the source node determines the equivalent capacity of a virtual circuit, the routing and call admission algorithms come into play.

The network routing algorithm is based on a distributed version of the Bellman–Ford–Fulkerson [3] least-cost routing algorithm. It is basically a shortest path routing scheme similar to the one used in the ARPANET. Whenever a call request comes in, the source node calculates the least cost path to the destination node using the current value of the link costs determined from its local database. Let \( \lambda_k \) denote the mean packet arrival rate of the \( k \)th virtual circuit which, under our assumptions, is the same as the equivalent capacity. The flow on a link is defined as the sum of the equivalent capacities of the virtual circuits using the link at that time. Specifically, the current link flow \( f_{ij} \) on the link from node \( i \) to node \( j \) is defined as \( f_{ij} = \sum_{k \in k_{ij}} \lambda_k \), where \( k_{ij} \) is the set of sessions using link \( i-j \). The cost of adding a new call with an average rate \( \lambda_n \) to link \( i-j \), which has capacity \( C_{ij} \) and current flow \( \hat{f}_{ij} \), is determined by

\[
\text{Cost}_{ij} = \frac{C_{ij}}{(C_{ij} - \hat{f}_{ij})(C_{ij} - f_{ij})} \tag{3.1}
\]

where \( \hat{f}_{ij} \) is the link flow including the new call

\[
\hat{f}_{ij} = f_{ij} + \lambda_n. \tag{3.2}
\]
Using the link costs calculated from (3.1), the source node computes the least cost path to the destination node. After the route for the virtual circuit has been identified, the call admission algorithm is invoked.

The concept of equivalent capacity enables a simple cell acceptance algorithm to be used in the network based on congestion thresholds. The call admission procedure is designed to admit only those calls which will not make the flow on any network link exceed a preset congestion threshold. This congestion threshold is set so as to guarantee a grade of service for the users while allowing for variations in the traffic among all the users as well as for the in-band control traffic. Let $TH_{ij}$ denote the congestion threshold for link $i-j$. The value of $TH_{ij}$ represents the fraction of link capacity available for transport of user traffic. The congestion thresholds are used in the admission procedure as follows. The source node makes a bandwidth request along the least cost path informing the nodes along the path of a new session request of bandwidth $\lambda_n$. Each node in the path computes the link flow resulting from adding the new connection using (3.2). If the resulting link flow is below the congestion threshold at all nodes in the path, then the call is accepted, otherwise the call is rejected. When a session is accepted, then each node in the path updates its current link flows to reflect the addition of the call $f_{ij} = f_{ij} + \lambda_n$. Note that a virtual circuit which is disrupted due to a link failure is treated as a new session request.

Observe that each node maintains local tables of the flow on each link in the network in order to make routing and call admission decisions. The addition or deletion of a virtual circuit along a path would lead to a change in the flows of all the links along that path. These updates are transmitted to all the other nodes in the network in the form of control messages. The control packets are sent out from each node along a spanning tree which allows the control packets to visit all the other nodes in the network before being discarded. A global spanning tree for the entire network is derived using a standard minimum spanning tree algorithm, and this is then localized to provide a certain fixed route for the control packets of any given node. The control messages are transmitted along with the normal data packets and are of two types: 1) asynchronous and 2) synchronous. Each type of control message has an update threshold associated with it. Asynchronous control messages are sent out at the time of set up/dowm of a virtual circuit if the change in flow exceeds the asynchronous update threshold. Its purpose is to notify the nodes of a significant change in the allotted bandwidth of a link. Normally, link flows are updated in a periodic fashion with a synchronous control message if the change in link flow exceeds the synchronous update threshold [16].

Flow control in plaNET is based on the virtual leaky bucket scheme where packets are violation tagged as discussed in [16]. End-to-end error control is not provided in the network with the retransmission of dropped packets being the responsibility of the network user. Selective repeat retransmission algorithms are currently being recommended for error control in high-speed networks since they minimize the number of retransmissions, and we assume the existence of this type of mechanism.

Since we are primarily interested in the effects of failures, several simplifying assumptions are made in the experimental model. Consider the primary node queuing model of Fig. 5(b). In this model, all links are assumed to be full duplex with the capacity of the primary node output link denoted by $C$ and the capacity of the rth source rerouted access link denoted by $C_i$. As in plaNET, variable length packets are transported via virtual circuit connections. Packets are assumed to arrive to the network according to independent Poisson processes with fixed mean rate $\lambda_i$ for the rth virtual circuit being reconnected. Furthermore, we assume exponentially distributed packet lengths with mean $1/\mu$, and that the service rate of a packet at a link is proportional to the link capacity. The buffer space at the network node output link is finite with system size $K$. We assume there is no congestion control on the sources to provide a worst-case scenario. Packets which are dropped in the network are retransmitted from the traffic source using a selective repeat mechanism.

In order to generalize the results so that they can be scaled to any channel capacity, the transmission rate of the primary node link was normalized to have a value of 1 (i.e., $C = 1$). Also, the mean packet length $1/\mu$ is normalized to a value of 1. Hence, the mean packet service rate of the output link is $\mu C = 1$, and one can unnormalize quantities by relating them to the mean service rate of the actual link $\mu C$. For example, consider a link which is a T1 line (i.e., $C = 1.544$ Mbps) in a network where the average packet length is $1/\mu = 2000$ bits/packet. Thus, in the actual network, the service rate is $\mu C = 772$ packets/s. Hence, a virtual circuit of mean rate $\lambda_i = 0.025$ in the normalized network translates to a virtual circuit of $\lambda_i \mu C = 38.6$ kbps in the actual network.

The plaNET network provides a guaranteed steady-state grade of service (GOS) for each virtual circuit that has been allowed to enter the network. The maximum flow on any network link is controlled by the link congestion thresholds $TH_{ij}$ in the call set up procedure. Here we assume the congestion threshold is $TH_{ij} = 0.85$. Hence, under the assumption that each queue can be represented by a finite M/M/1 queue, a worst case grade of service at any network link can be determined using the maximum flow. Specifically, the maximum allowable packet loss rate $PB$ and the maximum average number in the system $NS$ at any network queue can be found using the standard $M/M/1/K$ queuing formula. For example, in the networks modeled here, the maximum link utilization is $\rho = TH_{ij} = 0.85$ and the system size is $K = 21$ which results in values of $PB = 5.08 \times 10^{-3}$ and $NS = 5.03$. Thus, the guaranteed grade of service at each network link is an average number in the system $NS_{GOS} = 5.03 \approx 5$ and a packet loss rate $PB_{GOS} = 5.08 \times 10^{-3}$. The network is assumed to be performing satisfactorily only when the performance parameters are less than or equal to the GOS values.

The failure of a network link would be detected only after a certain amount of time. This detection would be done by one of the nodes to which the failed link is attached. This node would then transmit the information to all the other nodes as a control message. The virtual circuits that had been using the failed link continue transmitting packets until their source nodes receive
notice of the failure. These packets are assumed to be lost in the network and have to be retransmitted upon reconnection of the virtual circuit. Typical times for a node to detect a failure can range from a few milliseconds to two seconds. The downtime allowed before the user will disconnect the session is on the order of 2 s for a voice call and 10 s for data connections. Let τ denote the time taken by a node to detect the failure in the actual network, then the backlog of packets at the traffic source of the 4th session at the node will be \( \lambda \bar{\tau} \). Thus, for the same number of packets to be lost in the normalized network, where \( \lambda = \bar{\lambda}/\mu C \), the time to detect the failure \( \bar{\tau} \) must be given by \( \tau = \bar{\tau} \mu C \). Here we assume that the time to detect the failure in the actual network is \( \bar{\tau} = 1 \) s. Thus, in this second, a network operating at standard T1 (1.54 Mb/s) rates with the average packet size of 2 KB would have allowed the transmission of a maximum of 772 packets. Hence, a virtual circuit transmitting at an average rate of \( \lambda \bar{\tau} \) would have a backlog of \( 772 \times \lambda \bar{\tau} \) packets to be retransmitted. Upon restoration of the virtual circuit, the backlog can lead to a transient period where the grade of service is violated.

**B. A Technique for Determining the Nonstationary Behavior of Primary Nodes**

Consider the queuing model of a primary node output link shown in Fig. 5(b). Focusing on the network primary node output link queue, we assume that customers arrive to the network queue according to nonstationary Poisson processes with aggregate mean rate \( \lambda(t) \) at time \( t \). We denote the finite system size (buffer + server) existing at the queue by \( K \). Furthermore, we assume that the link can be approximately modeled as a \( M/M/1/K \) queue with mean service rate \( \mu C \), where \( 1/\mu \) is the average packet length and \( C \) is the link capacity. Defining \( p_j(t) \) as the probability of \( j \) packets being in the queueing system at time \( t \), then the Chapman-Kolmogorov differential equation for the queue can be derived as

\[
\frac{dp_j^K(t)}{dt} = -\lambda(t)p_j^0(t) + \mu C p_j(t) + \mu C p_{j+1}(t), \quad 0 < j < K
\]

\[
\frac{dp_j^0(t)}{dt} = \lambda(t)p_{j-1}(t) - \lambda(t)p_j(t) + \mu C p_j(t), \quad 0 < j < K
\]

\[
\frac{dp_{K}(t)}{dt} = \lambda(t)p_{K-1}(t) - \mu C p_K(t).
\]  

Given the time-varying link load \( \lambda(t) \) and initial conditions, one can solve (3.3) using standard numerical integration techniques. Consider the \( i \)th traffic source queue in the primary node model of Fig. 5(b). We assume the traffic source queues can be modeled as \( M/M/1 \) queues. After a link failure, the source queues will have a large number of back-logged packets to transmit. Thus, using a finite capacity detailed Chapman-Kolmogorov model, such as (3.3), to represent each queue is computationally difficult due to the large number of equations required. Hence, an approximate fluid flow model approach is adopted.

Consider the source queuing model of the \( i \)th connection at a primary node as shown in Fig. 8. Following [19], we define \( C_i \) as the capacity of the link, \( NS_i(t) \) as the number in the system (i.e., queue + server) at time \( t \), and \( x_i(t) \) as the state variable representing the average number in the system at time \( t \). Note that the state variable is the ensemble average of the number in the system at time \( t \) (i.e., \( x_i(t) = E\{NS_i(t)\} \)). Defining \( f_i^{out}(t) \) and \( f_i^{in}(t) \) as the ensemble average of flow out and flow in of the queue, respectively, then from the flow conservation principle, the rate of change of the state variable can be written as

\[
\dot{x}_i(t) = -f_i^{out}(t) + f_i^{in}(t).
\]  

Assuming that the queue storage capacity is unlimited and that customers arrive to the queue according to a nonstationary Poisson process with rate \( \lambda_i(t) \), then, \( f_i^{out}(t) = \lambda_i(t) \) since no packets are dropped. The flow out of the system \( f_i^{out}(t) \) can be related to the ensemble average utilization of the link \( \rho_i(t) \) by \( f_i^{out}(t) = C_i \rho_i(t) \). Note that \( \rho_i(t) = P(NS_i(t) > 0) \) and, as shown in [19], \( \rho_i(t) \) can be approximated by a function \( G_i(x_i(t)) \) which represents the ensemble average utilization of the link at time \( t \) as a function of the state variable. Hence, the link queue can be represented by

\[
\dot{x}_i(t) = -C_i G_i(x_i(t)) + \lambda_i(t)
\]  

with initial condition \( x_i(0) \). This type of fluid flow model has been used by several researchers [19], [24], [25] to describe queuing systems in terms of time-varying mean quantities. The utilization function \( G_i(x_i(t)) \) which accurately models the system depends on the queue under study and the data available. If experimental data from an existing system can be obtained, then the function can be determined statistically. However, such data are normally unavailable and one must determine \( G_i(x_i(t)) \) by other means such as matching the steady-state equilibrium point of (3.5) with that of the equivalent queuing theory model. Assuming that the queue can be modeled as \( M/M/1 \), then when the arrival rate to the queue is constant (i.e., \( \lambda_i(t) = \lambda_i \bar{\lambda} \)), the average number in the system at steady state is given by \( \lambda_i/(\mu C_i - \lambda_i) \) from the \( M/M/1 \) queuing formulas. Thus, requiring that \( x_i(t) = \lambda_i(t)/(\mu C_i - \lambda_i(t)) \) when \( \dot{x}_i(t) = 0 \), \( G_i(x_i(t)) = \lambda_i(t)/C_i \). Results in \( G_i(x_i(t)) = \lambda_i(t)/(1 + x_i(t)) \) and the state model becomes

\[
\dot{x}_i(t) = -C_i \left( \frac{x_i(t)}{1 + x_i(t)} \right) + \lambda_i(t)
\]  

with initial condition \( x_i(0) \). The solution of this fluid flow model has been shown to be an accurate approximation to the time-varying mean number in the system [19], [24] of an \( M/M/1 \) queue. However, to have an accurate model of the source queue after a failure, the effects of packet dropping at the primary node queue must be included in the model.

Consider a sample backlogged packet for the \( i \)th virtual circuit, the probability that the packet gets dropped on its
first attempt is \(PB\), the blocking probability at the primary node output link. Assuming independence of retransmissions and that the blocking probability remains constant over the period of interest, the probability that the same packet gets blocked on its \(n\)th attempt is \(PB^n\). Thus, the total number of retransmissions \(NR\) is given by the equation

\[
NR = PB + PB^2 + PB^3 + \cdots = PB \sum_{i=0}^{\infty} PB^i = \frac{PB}{1 - PB}.
\]

Hence, the total number of transmissions \(NT\) required for the tagged packet is \(NT = PB/(1 - PB) + 1 = 1/(1 - PB)\). Therefore, the average time \(TT_i\) required for one packet to successfully get transmitted at source queue \(i\) is given by

\[
TT_i = \left( \frac{1}{1 - PB} \right) \frac{1}{\mu C_i}.
\]

Note that the increase in the average time for the packet to be transmitted is equivalent to decreasing the capacity of the source queue access link by a factor of \((1 - PB)\). Let \(C_i\) denote the adjusted link capacity, that is, \(C_i = C_i/(1 - PB)\). Then, the fluid flow model of the \(i\)th source queue (3.6) becomes

\[
\dot{x}_i(t) = -\mu C_i \left( \frac{x_i(t)}{1 + x_i(t)} \right) + \lambda_i(t).
\]

Thus, the fluid flow model of (3.9) can be used to model the source queues taking into account packet retransmissions. Notice that the blocking probability \(PB\) at the primary node link must be known, and this can be determined from (3.3) as \(PB = p^K(t)\). Hence, the inclusion of retransmission effects provides a feedback coupling between the source queue and primary node link models.

The initial condition for the fluid flow model \(x_i(0)\) denotes the number in the system of the \(i\)th source queue at the time of reconnection of the virtual circuit. Let \(T_i\) denote the time to reconnect the \(i\)th call at the primary node in question. Then \(T_i\) is the sum of the time for the node to become aware of the failure and the time to set up the \(i\)th call. The initial condition \(x_i(0)\) is the backlog of packets that accumulates during the reconnection time which, assuming the load is stationary, is

\[
x_i(0) = \lambda_i \times T_i.
\]

Typically, this backlog is large and is primarily due to the time for the node to detect the failure rather than the reconnection set up time.

In order to evaluate the performance of the primary node queueing model, the arrival rate to the primary node output link \(\lambda(t)\) must be known. Let \(\lambda_{bg}(t)\) represent the time-varying mean arrival rate of the background traffic. Then \(\lambda(t)\) can be found in a fashion similar to that for product form queueing networks using the flow balance principal as

\[
\lambda(t) = \lambda_{bg}(t) + \sum_{i=1}^{N} \mu C_i \left( \frac{x_i(t)}{1 + x_i(t)} \right)
\]

where the second term on the right-hand side represents the arrival rate of traffic from the source queues. This equation provides a coupling between the queues, since the output rate from each source queue is given as a function of the number in the system at the source queue (i.e., flow out). Observe that although the effective service time at the source queues is reduced by retransmissions occurring, the actual flow of packets out of the source queues into the primary node link is unaffected. Also note that when steady-state conditions prevail at all source queues (i.e., \(x_i(t) = 0 \forall t\) and \(\lambda_i(t) = \lambda_i \forall i\), the arrival rate of packets to the primary node link becomes \(\lambda(t) = \lambda_{bg}(t) + \sum_{i=1}^{N} \lambda_i\).

Hence, the primary node queueing model of Fig. 5(b) can be modeled by a coupled differential equation model consisting of \(n\) equations of the form (3.9) representing the source queues along with the system of equations (3.3) to model the primary node link and the coupling equation (3.11) to determine the primary node queue arrival rate. The resulting model consists of \(n + K + 1\) differential equations and 1 coupling equation.

One can apply standard numerical integration techniques in an iterative fashion over the time intervals of interest to solve (3.3), together with (3.11) and (3.9), for the state probabilities \(p_i(t)\) of the primary node queue, and the number in the system \(x_i(t)\) at each source queue. The steps in the solution technique are as follows. Begin with some known boundary conditions at time zero \(p_i(0)\) and \(x_i(0)\), such as the steady-state number of customers in the primary node queue before the failure and the backlog of dropped packets at the source nodes. Over the first small time interval \([0 \leq t \leq t_1]\), assume a constant arrival rate at each queue [i.e., \(\lambda_i(t) = \lambda_i(0)\) and \(\lambda(t) = \lambda(0)\) from (3.11)]. Then a numerical integration technique, such as the Runge-Kutta method, can be used to solve the set of differential equations for \(p_i(t)\) and \(x_i(t)\) over the interval \([0, t_1]\). The state of the system at the end of the first time interval is given by the probability distribution \(p_i(t_1)\) and the state variables \(x_i(t_1)\), and this becomes the boundary condition for the next time interval \([t_1, t_2]\). One then selects new constant arrival rates for the new time interval and solves the differential equations again. This procedure is repeated for each time interval in the time horizon. From \(p_i(t)\) and \(x_i(t)\), one can study the performance of the system. For example, \(NS(t)\), the expected number of packets in the system at the primary node link at time \(t\), is given by \(NS(t) = \sum_{j=1}^{N} p_j(t)\).

In developing the differential equation model, a number of approximations were made; and to check the accuracy of the model, its solution was compared to a detailed discrete event simulation model. A typical experimental study is reported here for the two virtual circuit case with background traffic shown in Fig. 9. In the experimental model, the capacity of the network links was \(C = 1\), the mean packet length was one \((1/\mu = 1)\), and the system size of the network node links was \(K = 21\). As a worst case, the virtual circuits were assumed to be reconnected simultaneously with the time to reconnect being \(T_1 = T_2 = 772\) s which, as noted previously, translates to a time to detect a failure and reconnect of 1 s in a network at T1 rates. Note that in developing the analytical model, it was assumed that all traffic arriving to the source queues is Poisson, which implies under steady-state conditions that the external traffic arriving to the network nodes is Poisson. However, a component of the background traffic will be traffic from other
nodes, and thus the Poisson assumption is an approximation for this traffic. Hence, in the simulation model, the background traffic was set up to travel through two nodes mixing with additional traffic before the primary node.

An illustrative result of the model accuracy is shown in Fig. 10 where NS(t), starting at the instant that the virtual circuits are reconnected, is plotted for the analytical model along with the results of a detailed simulation. In Fig. 10, the results for two cases are shown: a) λ1 = 0.0086, λ2 = 0.0214, C1 = 0.34, C2 = 0.85, λbg1 = λbg2 = 0.05, λA = λD = 0.4; and b) λ1 = 0.013, λ2 = 0.037, C1 = 0.2975, C2 = 0.85, λbg1 = λbg2 = 0.025, λA = λD = 0.4. In case a), the background traffic load is three times the total source rerouted traffic; whereas for case b), the background traffic load is equivalent to the total source rerouted traffic. The values chosen for the traffic arrival rates were selected to illustrate the behavior of NS(t) near the GOS level, which is the range of interest in Section IV. Also note that the model parameters (i.e., K = 21, 1 s to detect failure, etc.) were selected to be representative of planET. In Fig. 10, for the simulation results, two curves are given showing the upper and lower bounds of the 95% confidence intervals determined from 3000 independent runs generated using the ensemble averaging technique of [20]. From the figure, one can see that the analytical model is reasonable accurate especially in estimating the maximum number attained in the system. The simulation was implemented in SLAM II on a SUN IV workstation and required 110 min of run time; in contrast, the differential equation model was implemented using the fifth-order Runge–Kutta numerical integration routine of MATLAB, and needed less than 1 min of run time.

IV. AN ANALYSIS OF CONGESTION AT PRIMARY NODES

A parameterized study of the primary node queuing model was conducted to determine when congestion occurs in the primary node. In the study, the normalized network model was used for the primary node output link. As noted in Section III-A, in the normalized model, the system size was K = 21, the link capacity was one (C = 1), the mean packet length was one (\( \frac{1}{\mu} = 1 \)), and the call admission congestion threshold was \( TH_{ij} = 0.85 \). The resulting grade of service at the network link was determined to be \( NS_{GOS} = 5.03 \) and \( PB_{GOS} = 5.08 \times 10^{-3} \). The primary node output link was considered congested if the average number in the system \( NS(t) \) exceeded the grade of service level (i.e., \( NS(t) > NS_{GOS} = 5.03 \)). The number in the system was chosen as the metric rather than the blocking probability since our numerical results showed that \( NS(t) \) was the more sensitive metric. Specifically, the grade of service was never exceeded by \( PB(t) \) unless it was also exceeded by \( NS(t) \), and the duration of time that the grade of service was exceeded by was always longer for \( NS(t) \) than \( PB(t) \). The time to detect the link failure and reconnect the ith session was assumed to be \( T_i = 772 \gamma_i \), which, as noted previously, is equivalent to 1 s in a network operating at T1 link rates. In order to concentrate on the effects of the rerouted traffic from the failed link and simplify the analysis, it was assumed that no background traffic exists.

The purpose of the study was to determine what combination of parameters (\( C_i, \lambda_i, \) etc.) would result in congestion at the primary node link. Note that the steady-state arrival rate to the primary node can never be greater than the congestion threshold of \( TH_{ij} \). Congestion can occur only when the arrival rate to the primary node link \( \lambda(t) > TH_{ij} = 0.85 \) for
A similar study was conducted for the case of two virtual circuits being rerouted through the primary node link. The experimental model was the same except for the additional access link $C_2$ and mean arrival rate $\lambda_2$ of the second session. The condition for congestion to be possible in this case is $C_1 + C_2 > 0.85$. Note that for the two virtual circuit case, there are four parameters to vary (i.e., $\lambda_1, \lambda_2, C_1, C_2$), and this makes an exhaustive search for bounding conditions difficult. Hence, to reduce the number of parameters, the values of $C_1$ and $C_2$ were fixed as well as the ratio $\lambda_1/\lambda_2$, and the total load $\lambda_1 + \lambda_2$ was varied to the point where the number in system exceeds the grade of service. This was repeated for different values of $C_1$ holding $C_2$ and $\lambda_1/\lambda_2$ constant, and the resulting values were plotted as a curve. The entire experiment was repeated for different values of the ratio $\lambda_1/\lambda_2$, and the resulting family of curves are shown in Fig. 12. Thus, in Fig. 12 each curve is a bounding curve for a particular ratio of the arrival rates. The effect of varying $C_2$ in the experiment can be seen in Fig. 12(b) where $C_2$ was fixed at 0.85 and the experiment repeated.

In Fig. 12, an interesting effect can be seen: notice that for a particular ratio of $C_1/C_2$, the bounding curve that is the worst case (i.e., lowest load to exceed the grade of service) is always the curve resulting from $\lambda_1/\lambda_2 = C_1/C_2$. For example, in Fig. 12(a) at the value $C_1/C_2 = 0.3$, the curve that is leftmost is the one resulting from $\lambda_1/\lambda_2 = 0.3$. The condition $\lambda_1/\lambda_2 = C_1/C_2$ is equivalent to $\rho_1 = \rho_2$, that is, the utilization of the two access links being equally balanced. In order to clearly demonstrate this worst case effect, an experimental study was conducted where the values of $\mu C_1, \mu C_2, \lambda_1 + \lambda_2$ were fixed and ratio $\lambda_1/\lambda_2$ varied. The results of a typical study of this type are shown in Fig. 13, where $NS(t)$ and $PB(t)$ are plotted versus time for various ratios of $\lambda_1/\lambda_2$. In Fig. 13, the horizontal lines represent the grade of service levels $NS_{GOS}, PB_{GOS}$. One can clearly see from the figure that when $\lambda_1/\lambda_2 = C_1/C_2 \rightarrow \rho_1 = \rho_2$, the maximum number in system and blocking probability occurs.

In general, one can show that balancing the utilization of the access links results in maximum packet loss probability at the primary node queue which is a worst case scenario. Consider the primary node queuing model shown in Fig. 5(b) for the general case of $n$ virtual circuits being connected. The initial backlog of the source access queue $x_i(0), i = 1, 2, \cdots, n$, is found from (3.10), and we assume that the time to reconnect $T_i$ is the same for all the virtual circuits (i.e., $T_i = T \forall i$). It will also be assumed that the total average load $TL$ of the disconnected virtual circuits is fixed with $TL = \Sigma_{i=1}^n \lambda_i$, and that the average background traffic $\lambda_{bg}$ is fixed. Obviously, the problem of maximizing the packet loss rate at the primary node reduces to a consideration of the transient associated with working off the backlogs $x_i(0), x_2(0), \cdots, x_n(0)$ at the source queues. Note that the total number of backlogged packets $TK$ is given by

$$\sum_{i=1}^n x_i(0) = T \sum_{i=1}^n \lambda_i = TK$$

and is a constant independent of the variable parameters. This total number of backlogged packets is delivered to the primary

![Graph showing boundary curves for single virtual circuit with background traffic.](image-url)
node regardless of the choice of parameters. Maximum packet blocking at the primary node occurs when the backlogged packets are delivered to the primary node queue in the least time. Let $TB_i$ denote the time required by the $i$th source queue to deliver its backlogged packets to the network. Note that the mean service rate seen by the backlogged packets at the $i$th source queue is just the adjusted line capacity $\mu \bar{C}_i = \mu C_i(1 - PB)$ minus the proportion of the link capacity utilized by the regular traffic $\lambda_i$. Hence, $TB_i$ can be approximated by

$$TB_i = \frac{x_i(0)}{\mu \bar{C}_i - \lambda_i} = \frac{\lambda_i}{\mu C_i(1 - PB) - \lambda_i}$$

where $\rho_i = \lambda_i/\mu C_i$ and $PB$ is the blocking probability of the primary node queue. This blocking probability is the same in each $TB_i$, and is independent of the source queue.\(^1\) The time for each source to deliver its backlog to the primary node is the same function, $TB(\rho_i) = TB_i(\rho_i)$ of $\rho_i$.

The time $TD$ to deliver all of the backlogged packets to the primary node queue is given by

$$TD = \max_i [TB(\rho_1), \ldots, TB(\rho_i), \ldots, TB(\rho_N)]$$

\(^1\)Although $PB$ is the same in the expression for each $TB_i$, it can be a function of time.

In order to achieve maximum blocking, $TD$ must be minimized by choice of the $\rho_i$. Thus,

$$\min TD = \min \max_i [TB(\rho_1), \ldots, TB(\rho_i), \ldots, TB(\rho_N)]$$

Denote the value of $\rho$ which minimizes the maximum of $TB(\rho_i)$ as $\rho^*$. Thus,

$$\min TD = TB(\rho^*)$$

Equation (4.5) requires $TB(\rho^*)$ to be the largest of the $TB(\rho_i)$. At the same time, (4.6) specifies $TB(\rho^*)$ as the minimum over all $\rho_i$ and hence as the smallest of the $TB(\rho_i)$. The only way that $\rho^*$ can make $TB(\rho^*)$ both the largest and the smallest of the $TB(\rho_i)$ is for all of the $\rho_i$ to be equal and equivalent to $\rho^*$. Hence, the maximum backoff loss occurs at the primary node when the utilization of the source access links are equally balanced. Experimental results demonstrating this effect are given in [23].

Since having balanced source access queue utilizations results in the worst case congestion, by considering the $\rho_1 = \rho_2$ case in the experimental model we can construct a conservative bounding curve similar to Fig. 11, below which the grade
of service will always be met. Specifically, the experimental study used to create Fig. 12 can be repeated to generate a worst case curve by repeating the study maintaining \( \lambda_1 / \lambda_2 = C_1 / C_2 \) at each point, then determining the load where the grade of service is just met. The resulting bounding curve for two cases of \( T \), the time to reconnect the virtual circuits, is shown in Fig. 14(a). Note that the effect of \( T \) is to change the initial backlog of packets at the source queues, and thus the range of virtual circuit arrival rates being permissible before the grade of service is exceeded. The effect of a stationary background load \( \lambda_{bg} \) is also illustrated in Fig. 14(a). A set of bounding curves for various values of \( T \) and \( \lambda_{bg} \) can easily be constructed by repeating the experiment. Note that it may be preferable from a network standpoint to allow some congestion to occur at the primary node. A similar set of bounding curves can be determined for any level of congestion using the same approach. For example, in Fig. 14(b), the bounding curves are shown for the condition that \( N S(t) \leq 10.5 \, \text{Vt} \).

Bounding curves (such as Figs. 14 and 11) are potentially quite useful for network control and could be utilized for call admission or adjustment of flow control settings after a failure. For example, after a link failure which takes 1 s to detect, a primary node with two virtual circuits to reconnect could use Fig. 14(a) to determine whether or not to admit the calls on the basis of preventing congestion.

The numerical approach above for constructing bounding curves should, in theory, be extendable to the general \( n \) virtual circuit primary node case of Fig. 5(b). However, the number of parameters to vary becomes difficult to handle with increasing \( n \) and here we propose an algorithm for determining if congestion occurs based on the results from the two virtual circuit case. Given \( \lambda_1, \lambda_2, \ldots, \lambda_n \) and \( \mu C_1, \mu C_2, \ldots, \mu C_n \), we basically compare them two at a time then combine the two together that we have just compared. The procedure can be written in algorithmic form as follows.

1) Let \( \lambda_{sum} = \lambda_1, \mu C_{sum} = \mu C_1, j = 2 \)
2) Check two call bounding curve for \( \lambda_{sum}, \lambda_j, \mu C_{sum}, \mu C_j \)
3) If congestion occurs (i.e., above curve) then
   stop; congestion will occur for \( n \) call case
else if \( j < n \) then
   \[ \lambda_{sum} = \lambda_{sum} + \lambda_j \]
   \[ \mu C_{sum} = \mu C_{sum} + \mu C_j \]
   \[ j = j + 1 \]
   go to step 2
else if \( j = n \) then
   stop; congestion will not occur
end if.

Thus, using this algorithm, we can determine whether or not congestion will occur at an arbitrary primary node for any number of virtual circuits needing to be restored. For example, consider the situation where \( \lambda_1 = 0.007, \mu C_1 = 0.85, \lambda_2 = 0.007, \mu C_2 = 0.85, \) and \( \lambda_3 = 0.007, \mu C_3 = 0.85 \). Applying the algorithm, we see that on the first iteration \( \lambda_{sum} = 0.007, \lambda_2 = 0.007, \mu C_{sum} = 0.85, \mu C_2 = 0.85 \) and the resulting point \((0.014, 1)\) lies below the \( T = 1 \) bounding curve of Fig. 17(a) thus avoiding congestion. On the second iteration,

\[ \lambda_{sum} = 0.014, \lambda_3 = 0.007, \mu C_{sum} = 1.7, \mu C_3 = 0.85 \] and the resulting point \((0.021, 2)\) lies above the bounding curve implying that congestion will occur. A numerical simulation of the differential equation model verifies the conclusion. The results of the algorithm above tend to be conservative due to the combining of the access queues after each iteration.

V. CONCLUSIONS

In this paper we have presented a study of the performance impact of link failures in a virtual circuit wide area network. It was shown that a link failure and the subsequent traffic restoration divides the network nodes in to four groups: primary, secondary, tertiary, and quaternary. A generic queueing model framework for studying the performance impact of failures at the different types of nodes was presented. It was shown that congestion after a link failure starts at the primary nodes (i.e., the source nodes of virtual circuits that were on the failed link) and spreads to the other nodes. A detailed nonstationary performance analysis of a generic primary node was conducted using a numerical methods differential equation model approach. A set of boundary curves which specify the conditions under which a primary node will become congested.
was presented, and an algorithm for its general use developed. The bounding curves can be utilized for call admission or to set congestion controls after a failure so as to prevent congestion at primary nodes and therefore the rest of the network. A possible better approach would be to allow some level of congestion at primary nodes but not enough to spread congestion to secondary and tertiary nodes. However, this requires analysis of the more complex queueing models of secondary and tertiary nodes.

REFERENCES


Krishna Balakrishnan received the B.E. degree in electronics and communication engineering from the Regional Engineering College at Tiruchirapalli, India, in 1989, and the M.S. degree in computer engineering from Clemson University in 1992. From 1989 to 1990, he was working at Tata Consultancy Services, Bombay, India, as a computer consultant. He is currently pursuing the doctoral degree at Clemson University.

Sunit Menon (S'90-M'92) received the Bachelor of Technology degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, in 1990, and the M.S. degree in computer engineering from Clemson University, Clemson, SC, in 1992.

From August 1990 to December 1992, he was a Teaching Assistant in the Department of Electrical and Computer Engineering at Clemson University. From January 1992 to May 1993 he was a Research Assistant in the Department of Electrical and Computer Engineering at Clemson University. Since August 1992, he has been with Motorola Codex, Canton, MA, where he is a member of the Research and Advanced Development Group, working on the architecture and implementation of high-speed networks.

Mr. Menon is a member of Phi Kappa Phi.