Abstract

In this paper, a network level abstraction called \( \Phi \)-channel to support the requirements of real-time applications is proposed. A \( \Phi \)-channel represents a simplex, end-to-end communication channel between a source and a destination. The channel is characterized by a set of specific performance parameters associated with its traffic, namely packet maximum end-to-end delay and the maximum number of packets that can be sent over that delay. The primary attribute supported by the \( \Phi \)-channel is the on-time reliability.

The basic scheme that our model uses to verify the feasibility of accepting a \( \Phi \)-channel, and the run-time support to guarantee its performance are described. The results of a simulation experiment implementing the basic functionalities of the proposed scheme are presented.

1. Introduction

The technological advances in transmission systems created opportunities for a new class of real-time services and applications, such as voice, video, teleconferencing, image processing and scientific computing. These new communication services depart from the conventional telephone service in three essential aspects. They can be multi-media, multi-point and multi-rate. Besides a broad range of bit rates, certain classes of applications may have a time varying bit rate requirements over the duration of the connection while other classes may exhibit a varying amount of burstiness. Furthermore, the characteristics of these applications span a wide range of performance requirements, in terms of throughput, reliability, jitter bounds and delay.

The offering of these future services may require more than providing transmission and switching capabilities to establish a connection. More importantly, the network must provide a guaranteed level of support to the real-time applications and a flexible management of the connections and media requests.

1.1 Network Support For Real-time Applications

In order to support predictable level of performance required by real-time applications, the network must provide the means to characterize the real-time service parameters, allow some form of explicit resource allocation and maintain a degree of cooperation among its components in order to enforce the allocation policy. It is clear that without proper characterization of network parameters, and without any resource allocation, it is extremely difficult, if not impossible, to provide predictable performance [5].

Several network abstractions have been proposed and successfully implemented. The "best-effort paradigm" offered by the Internet architecture, which proved to be very successful in the realization of a universal network in a heterogeneous environment may not be adequate to support real-time traffic. The network protocol offers no guarantees about timely, reliable, or ordered packet delivery. Circuit switching, which is the standard method for providing real-time performance, does not optimize the utilization of the network resources, and may be very inadequate for short transmissions, where the connection setup overhead is prohibitive with respect to the duration of the transaction. Furthermore, the scheme offers no provision for specifying data rates or limits on delays, and supports no guaranteed performance.

1.2 Schemes For Real-time Networks

A closer look at the datagram and virtual circuit communication paradigms, however, reveals that the two models are not conceptually disjoint. Consequently, several design issues often closely associated with these communications styles can be decoupled from these network abstractions, and combined to form new and effective network implementations. Based on these views, the network design can be regarded as a set of of independent choices within multi-valued ranges of properties. Two approaches may be used to provide network performance guarantees. The first approach is to over-engineer the network to the extent that an application is certain to get the resources it needs. Thus, the applications can be unconstrained in its resource usage and still receive the guaranteed level of performance. In a high speed network, this approach would require prohibitively large amount of resources.

The second approach involves monitoring and controlling resource allocation and usage for each application. This approach requires that an application specifies...
its resources needs a priori, and unless its needs can be met, it is blocked or rejected. Once started, mechanisms are provided to ensure that the application does not use more resources than requested. Because every application uses only its share of resources, all performance needs can be met.

More recently, these views have been incorporated into a number of schemes which aimed at providing performance guarantees based on virtual-circuits, resource reservation and fixed routing have been proposed [1,2,4,8]. A rate based network control system which provides guarantees for average throughput was proposed by Zhang [8]. The scheme uses the concept of the "virtual clock", and aims primarily at regularizing the data flows according to the average throughput. The scheme does not provide guarantees to support end-to-end deadlines, and does not support an adequate buffer management scheme to limit packet loss due to buffer overflow. Another project, Multiple Congram Oriented High Performance Internet Protocol (McHIP), aims at providing variable grade services with guarantees for communication across subnets with diverse capabilities [6]. The scheme, however, does not provide a clear definition of the connection semantics and the mechanism used for implementing these connections.

The DASH communication system provides an abstraction called Session Reservation Protocol (SRP) [1]. The DASH resource model describes a set of primitives that allow a user to express workload characteristics and performance requirements.

Ferrari and Verma[4] have proposed a real-time channel that provides statistical guarantees of end-to-end performance. The real-time channel is viewed as a binding contract between the client and the network. By controlling admission of new channels, the network ensures that it can meet its guarantees. Notice, however, that the model is statistical in nature and can only provide those guarantees over an extended period of time.

Golestani proposes the stop-and-go queuing as a framework for congestion management in integrated service packet network [2]. The scheme aims to preserve the smoothness property of the network traffic, by dividing time into frames and guaranteeing that only packets of the previous frame are transmitted in the current frame. The close coupling between the frame size, the number of hops in the path and the delay, however, does not provide a flexible way to specify the delay requirements of the supported applications. In other words, the choice of the frame size and the number of hops constrains the message delay of the supported application.

In this paper, we propose a fractional network level abstraction, called \( \phi \)-channel to support real-time applications.

This abstraction is an extension of the \( \alpha \)-channel scheme [10] A \( \phi \)-channel represents a fractional simplex, end-to-end communication channel between a source and a destination. The channel is characterized by a set of specific performance parameters associated with its traffic. Upper layers treat the \( \phi \)-channel as an abstraction and specify the required performance characteristics in terms of the packet maximum end-to-end delay and the maximum number of packets that can be sent over that delay. The primary attribute supported by the \( \phi \)-channel is the on-time reliability. Our scheme differs from the schemes described above primarily in terms of the specifications characteristics of the real-time channel, the mechanism used to establish a new channel, and the mechanisms used to maintain the specified rate among the node along the routing path. Furthermore, the scheme offers a flexible way for the application to specify its end-to-end delay requirements.

In the next section of the paper, we discuss the network support for real-time applications. In section 2, we introduce the concept of the \( \phi \)-channel. The resource reservation model used to guarantee the requested performances and the issues related to the queue management of a switching node are described in section 3. The last section is dedicated to the description of the preliminary result of a simulation experiment that implements the basic components of the proposed scheme.

2. \( \phi \)-Channel, A Real-time Network Level Abstraction

A real-time communication service must provide the applications with the ability to specify their performance requirements and to obtain guarantees about the satisfaction of these requirements. Furthermore, the network must monitor these parameters and guarantee that the original agreement is not violated. Generally, however, the accuracy of evaluating the characteristics of the applications increases as the number and types of parameters used increases. Such proliferation not only complicates the implementation of the control function that monitors those parameters, but may impose some difficulties on the parameters declaration. Furthermore, if the set of traffic specification parameters involves relational quantities that are hard for the user to estimate, the resulting network abstraction may become inefficient and complicated. Consequently, the specification parameters of the network channel must be sufficient to estimate the statistical characteristics of the supported applications, easy to declare and allow an easy implementation of the control and access restriction scheme.
The $\phi$-channel is based on two parameters, namely the maximum end-to-end delay of each packet generated by the supported application, and the maximum number of packets that can be generated during this delay. These parameters have a clear and precise meaning for the user. Consequently, they are easy for the user to declare. Furthermore, based on the specified parameters, the monitoring policy can easily control that the generated traffic conforms to the parameters specified at the creation time.

The primary performance attribute supported by the $\phi$-channel is the on-time reliability. This attribute represents the minimum number of this channel's packets generated over a per-node end-to-end delay that must be delivered on time, assuming that no failures occur among the nodes of the routing path.

2.1 Specification of the Semantics of $\phi$-channels

An $\phi$-channel represents a simplex, fractional, end-to-end communication channel between two entities, a sender and a receiver. In its simplest form, a fractional guarantee expresses a bound such that over a sequence of $p$ consecutive packets, $q$ packets of this sequence satisfy some constraint [3]. This simple form of a fractional channel can be generalized to incorporate a set of parameters, a set of performance values, a set of constraints that express a relationship between the parameters and the performance values, and a desired level of support for the specified parameters. More specifically, a generalized fractional channel can be characterized by:

- A set of parameters, $p = \{ p_i, i=1,2,...,n \}$, that describes the characteristics of the real-time application, such as end-to-end packet delay, throughput and reliability. These parameters are usually specified by a transport entity or any higher level entity.
- A set of variables, $v = \{ v_i, i=1,2,...,l \}$, which represents the run-time variables of the channel.
- A set of constraints, $\Pi = \{ \pi(v,p), i=1,2,...,m \}$, that describes a relationship between the parameters $p_i$'s and the values $v_i$'s. The exact form of these constraints depends on the supported application. The constraints usually express a deadline to be met, a level of error control to be enforced, or a degree of reliability to be supported.
- A desired level of support $\Phi(s,t), i=1,2,...,m$, for each stream of $s$ consecutive packets generated by the application over a time interval $t$. The parameter $\Phi(s,t)$ represents the minimum number of packets among the $s$ packets generated over $t$ that must fulfill constraint $\pi(v,p), i=1,2,...,m$.
- A relation, $R$, between $C^1(s,t)$ and $\Phi(s,t)$, where $C^1(s,t)$ represents the number of times the network system fulfilled the constraint $\pi(v,p), i=1,2,...,m$.

Based on the above characterization, the fractional requirements of a network level channel may be formulated as

$$C^1(s,t) \leq \Phi(s,t), i=1,2,...,m$$

The value $\Phi(s,t)$ ranges between 0 and $s$. Based on the specified value of $\Phi(s,t)$, three types of $\phi$-channels may be defined. The first type, a fully guaranteed $\phi$-channel, is characterized by $\Phi(s,t) = s$. The second type, partially guaranteed, is characterized by $0 < \Phi(s,t) < s$. The third type, non-guaranteed $\phi$-channel is characterized by $\Phi(s,t) = 0$.

The semantics of these types of channels derive from the number of packets generated over $t$ that need to be guaranteed by the $\phi$-channel. The first type requires that all of the generated traffic satisfies the constraints $\pi(v,p), i=1,2,...,m$, specified on its parameters. The second class requires that only $\Phi(s,t)$ of the generated traffic satisfies the constraints $\pi(v,p), i=1,2,...,m$, while enforcing the constraints for the rest of the traffic will only be considered if the current conditions of the node permit. Finally, the third type is characterized by the absence of any guarantees. Consequently, traffic generated by this type of channel is serviced only if the network conditions permit without any guaranteed performance.

As stated above, a $\phi$-channel is characterized by the end-to-end delay any packet may suffer, the maximum number of packets that can be generated during this delay and the on-time reliability level which represents the number of packets that need to be delivered on time by the network over an end-to-end delay. Based on the general abstraction of a fractional channel, the specification of a $\phi$-channel, $k$, may be expressed by

The per-node delay, $\Delta_k(n)$, for each node, $n$, of the routing path, and the maximum number of packets, $N_k$, to be generated during this delay.

The variable, $\delta_k(n)$, representing the actual delay incurred by a given packet at node $n$.

The set of constraints, $\Pi$, which is reduced to the on-time reliability constraint, $\pi(\delta_k(n),\Delta_k(n))$. This constraint states that $\delta_k(n) \leq \Delta_k(n)$ for a packet of a $\phi$-channel to meet its on-time reliability.

A desired level of support $\Phi_k(N_k,\Delta_k(n))$. Since the primary attribute supported by the $\phi$-channel is the on-time reliability, $\Phi_k(N_k,\Delta_k(n))$ represents the minimum number of packets generated over $\Delta_k(n)$.
that need to be delivered on time. Therefore, the relation \( R \) may be expressed as:

\[
C(N_k, \Delta_k(n)) \geq \Phi(N_k, \Delta_k(n))
\]

Depending on the type of the \( \phi \)-channel, \( \Phi(N_k, \Delta_k(n)) \) varies between 0, for a non-guaranteed channel, to \( N_k \) for a fully guaranteed channel. Intermediate values between 0 and \( N_k \) characterize a partially supported channel.

In order to support the semantics of the \( \phi \)-channel, the network must use and enforce some form of explicit resource allocation policies. In the following section, we discuss the resource reservation model used by the \( \phi \)-channel abstraction to guarantee the requested performance. The basic components of the model, namely channel acceptance process and the run-time support process, will be described.

### 2.2 Resource Reservation Model Specification

The sender requests a \( \phi \)-channel by specifying the real-time performance characteristics required by the application, at the \( \phi \)-channel creation time. The sender specifies the end-to-end delay of each packet, the maximum number of packets to be sent over the specified delay, and the on-time reliability required. For each request, the network must identify a path from the source node to the destination that can support the characterization of the new channel. This identification process consists of two phases, namely route selection phase and verification phase. In the route selection phase, one or more paths from the source node to the destination node are selected. Each path identifies a set of switching nodes that can potentially support the level of on-time reliability required by the \( \phi \)-channel. A variety of routing algorithms for long haul networks have been proposed in the literature. Many of the distributed algorithms that adapt to the network state dynamically suffer from the problem of instability as changes in network conditions cause the computed best paths to oscillate between two or more alternatives. Consequently, a stable and more static routing algorithm may lead to stable paths for a \( \phi \)-channels. Models to characterize the excess capacity of the switching nodes and ways of incorporating this information in determining the set of nodes that can support the \( \phi \)-channel are currently being investigated. These issues, however, are outside the scope of this paper. It is assumed that the routing protocol is responsible for providing a path between the source node and the destination node.

Given the routing path, the second phase of the identification process consists of verifying that each node of the provided path can support the real-time requirements of the new \( \phi \)-channel without violating the guarantees provided by predecessor nodes of the routing path.

The feasibility of accepting a new \( \phi \)-channel is achieved in our scheme by the Channel Acceptance process. All accepted \( \phi \)-channels are then managed by the Run-Time Support process. The Run-Time Support process determines the order according to which the packets may be serviced in order to preserve the guaranteed on-time reliability of the \( \phi \)-channel.

### 2.3 Channel Acceptance Process

This process is responsible for determining whether to accept or reject a newly generated \( \phi \)-channel. Based on the specifications of the \( \phi \)-channel, the process determines whether the new \( \phi \)-channel can be serviced such that its real-time requirements are met, and all existing \( \phi \)-channels requirements continue to be guaranteed.

As stated above, the only information provided to the node specifies the maximum number of arrivals per node delay and does not provide any specification of the arrival pattern. Consequently, the Channel Acceptance process must consider the worst case arrival pattern. This occurs when all packets arrive simultaneously at the node, and all \( \phi \)-channels are operating at their maximum rate. Under these conditions, the Run-Time Support process must be capable of processing the offered traffic without violating the deadline requirements.

#### 2.3.1 Verification Model

The main idea underlying our verification scheme is based on the relationship among the maximum number of packets received by a given node over a given time interval, the per-packet processing time and the delay a packet may suffer at that node. This relationship may be stated as follows:

*The knowledge of both the maximum number of packets received by a given node over a given time interval and the per-packet processing time determines the maximum delay a packet may suffer at that node.*

Based on the above observation, the process used by an intermediate node, \( n \), to determine whether a \( \phi \)-channel is acceptable consists of verifying that the above relationship holds in the case where:

- The given time interval is the per-node delay, \( \Delta(n) \), of the \( \phi \)-channel, \( k \). The per node delay is the end-to-end delay divided by the length of the routing path.
- The maximum number of packets is the total number of packets received from all \( \phi \)-channels over this time interval that require a guaranteed service, and
• The processing time required by a packet of $\phi$-channel, $k$.

Recall that the traffic requiring guaranteed service results from the traffic generated by a fully guaranteed channel, and the $\Phi(s,t)$ portion of the traffic generated by a partially guaranteed channel. In the following, a more formal description of the verification scheme is provided.

2.3.2 Formal Specification of the Model

Let $n$ be the intermediate node where the verification scheme is being performed, and assume that $\phi$-channels $1,2,\ldots,N$ are the channels currently supported by node $n$. Furthermore, assume that $\Delta_i(n) \leq \Delta_j(n)$ if $i < j$. The basic idea used by the Acceptance algorithm is stated in the following Theorem.

**Theorem:**

If for all $k = 1, 2, \ldots, N$,

$$\sum_{i=1}^{k} W(i,k) + \Phi_k(N_k, \Delta_k(n)) \times p + p \leq \Delta_k(n),$$

then $\delta_k(n) \leq \Delta_k(n)$, for all $k = 1, 2, \ldots, N$,

where $\delta_k(n)$ is the delay observed by a packet, $p$, from $\phi$-channel $k$ at node $n$, and $W(i,k)$ ($i < k$), represents the amount of time required, in the worst case, to service the maximum number of packets from $\phi$-channel $i$ with deadlines occurring during any interval of size $\Delta_i(n)$. The deadline of packet $p$ at node $n$ is defined as $\alpha_p - \Delta_p - p$, where $\alpha_p$ is the arrival time of packet $p$ at node $n$, and $p$ is the packet processing time.

The worst case pattern of arrivals of two $\phi$-channels $i$ and $k$ occurs when the packets from $\phi$-channel $i$ arrive in a way such that the deadline of the first packet coincides with the arrival of a packet from $\phi$-channel $k$ and continue to arrive at their maximum predefined rate, as illustrated by Figure 1.

Based on the above observation, the quantity $W(i,k)$ may be computed by counting the maximum number of deadlines that can occur during an interval of size $\Delta_i(n)$. The number of deadlines in this interval is distributed among the three regions, $R_1$, $R_2$, and $R_3$. The first region, $R_1$, contains $\Phi_k(N_k, \Delta_k(n))$ deadlines. The cumulation of servicing these deadlines specifies the size of the region. The second region, $R_2$, contains the number of intervals of size $\Delta_i(n)$ completely embedded in an interval of size $\Delta_k(n) - R_1$. The amount of time required to service the deadlines that may occur within $R_2$ may be computed as:

$$\left(\frac{(\Delta_i(n) - R_1)}{\Delta_i(n)}\right) \times \Phi_k(N_k, \Delta_k(n)) \times p.
$$

Finally, $R_3$ can be expressed as $R_3 = \Delta_k(n) - R_1 - R_2$. Thus a deadline from $\phi$-channel $i$ will only need to be serviced during $R_3$ if $\Delta_i(n) - \Phi_k(N_k, \Delta_k(n)) \times p \leq R_3$, in which case the number of deadlines from $\phi$-channel $i$ that will require service during $R_3$ is

$$\left[\left(R_3 - (\Delta_i(n) - \Phi_k(N_k, \Delta_k(n)))\right) + 1\right].$$

Therefore, the value of $W(i,k)$, in the worst case, during an interval of size $\Delta_k(n)$, may be expressed as:

$$W(i,k) = R_1 + \left[\left(\Delta_i(n) - R_i\right)/\Delta_i(n)\right] \times \Phi_k(N_k, \Delta_k(n)) \times p +

\left\{\left(R_3 - (\Delta_i(n) - \Phi_k(N_k, \Delta_k(n)))\right) + 1\right\} \times p.$$

**Figure 1- Worst case packet arrival pattern.**

Based on the above theorem, the Acceptance Algorithm may be stated as follows. Let $C_k = \{\text{the set of } \phi\text{-channels currently supported by } n\}$. Assume that $|C_k| = N - 1$. Furthermore, let $k$ be a new $\phi$-channel requesting to be established and assume that $\Delta_1(n) \leq \cdot \cdot \cdot \Delta_{k-1}(n) \leq \Delta_k(n) \leq \Delta_{k+1}(n) \cdot \cdot \cdot \Delta_N(n)$. The new $\phi$-channel $k$ is accepted if and only if the two following conditions are satisfied:

$$k \sum_{i=1}^{k} W(i,k) + \Phi_k(N_k, \Delta_k(n)) \times p + p \leq \Delta_k(n) \quad \text{(i)}$$

and

$$k \sum_{i=1}^{k} W(i,k+j) + \Phi_{k+j}(N_{k+j}, \Delta_{k+j}(n)) \times p + p \leq \Delta_{k+j}(n) \quad \text{for all } j = 1, 2, \cdot \cdot \cdot (N - K).$$

According to the previous theorem, if conditions (i) and (ii) are verified, then for any channel $i \in C_k$, $\delta_i(n) \leq \Delta_i(n)$ for all packets, $p$, of channel $i$. Consequently $\phi$-channel $k$ must be accepted. Otherwise admission of the new $\phi$-channel into the network causes the guarantees of at least one $\phi$-channel to be violated. Consequently, the new $\phi$-channel must be rejected.
In the next paragraph, we describe the run-time support process to service deterministically packets in a switching node.

2.4 Run-Time Support Process

In order to guarantee real-time requirements, the Run-Time Support process must provide deterministic packet service. This can be accomplished if the process can determine the order in which all packets are to be serviced, given any arrival pattern. In the following sections, a description of the basic scheme used to manage different queues in the switching node and the policy used to allocate the CPU to these packets is provided.

2.4.1 Processing Node Configuration

Each packet switching node contains a service queue and a set of $\phi$-channel rate regulator queues. These queues are managed by two entities, namely a service queue manager, and a $\phi$-channel rate regulator. The configuration of the switching node is illustrated in Figure 2.

![Figure 2- Configuration of a switching node.](image)

The service queue manager waits on the service queue. This queue contains all packets that are currently eligible to be serviced by the CPU. The $\phi$-channel rate regulator monitors the rate regulator queues. The switching node provides a rate regulator queue for each $\phi$-channel currently supported by that node. The rate regulator is responsible for moving packets from the regulator queues into the service queue. In the next sections, a more detailed description of the basic operations performed by these two entities and the policies used to move packets from the rate regulator queue into the service queue is provided.

2.4.2 Service Queue Manager

As stated previously, the service queue contains all packets that are eligible to be serviced. These packets are ordered based on their per-node delays. Packets with shorter per-node delays are serviced before those with larger per-node delays. When a packet has finished receiving service from the CPU, the service queue manager provides the CPU with the packet currently holding the shortest per-node delay in the queue. This packet may be labeled as best-effort, in which case the service queue manager must first verify that no packets requiring fully guaranteed service is currently awaiting service. If there are no packets requiring fully guaranteed service in the service queue, then the best-effort packet is serviced. Otherwise, the best-effort packet is dropped, and the next packet in the queue is considered. This continues until the next most eligible packet requesting fully guaranteed service is located. Consequently, best-effort packets are only serviced if no packets requesting fully guaranteed service are awaiting service.

2.4.3 $\phi$-channel Rate Regulator

The primary responsibility of the flow regulator is to maintain the flow rate of each channel at each switching node. This is necessary to ensure that the number of packets arrival a node may see does not exceed the prescribed rate of the $\phi$-channel. The rate regulation employed is similar to those proposed by [2] and [4]. When a packet is serviced by a node, the difference between its service time and its deadline is computed. A field in the packet's header is stamped with this value and then the packet is sent to the next node. The following node, upon receiving the packet, appends it to the channel's rate regulator queue. The packet remains in this queue for the amount of time indicated by the value in the packet's header. After this amount of time has elapsed, the packet is moved from the rate regulator queue into the service queue. This scheme prevents packets from a channel from exceeding the channel's prescribed rate.

2.4.4 $\phi$-channel Network Edge Regulator

Since the network is providing performance guarantees, it must limit the amount of packets requesting fully guaranteed service that enter the network. The above mentioned verification scheme, which either accepts or rejects channels, is used to limit the long-term arrival process of packets requesting fully guaranteed service. However, the network must also protect itself from applications which have been accepted, but are generating packets requesting fully guaranteed service above their agreed to rate. These excess packets may be unintention-
ally or intentionally generated. In either case, the network must filter out these excess packets.

The network must, therefore, employ some form of channel monitoring to verify that each channel is obeying its specified rate. This can be done at each node or at the network edge. Monitoring within the network requires each node monitor each channel that it supports. This may be computationally expensive. The alternative, to monitor at the network edge, requires only the edge nodes to perform the per channel monitoring. In one possible edge monitoring scheme, the network maintains a list of the last $\Phi(N,\Delta)$ arrival times from each channel. When a packet arrives from a channel, the oldest value in the list is compared against this packet’s arrival time. If the difference in these two times is greater than $\Delta$, the packet is marked as D and this packet’s arrival time replaces the oldest packet’s arrival time in the list. Otherwise, the packet is marked as Best-Effort (BE). This scheme insures that, over any interval of size $\Delta$, not more than $\Phi(N,\Delta)$ packets will be marked as requesting fully guaranteed service.

3. Simulation Implementation

In order to test the validity of the proposed scheme, a simulation model was developed. The purpose of the simulation is demonstrate that the scheme satisfies the specified guarantees, even in the worst case situation. Furthermore, the experiment aims at demonstrating the usage of the network resource, and the efficiency of the proposed scheme to handle a wide range of applications with different load and burstiness characteristics. To achieve the latter purpose, different types of channels, namely bursty, exponential and periodic were simulated. The bursty channel generates the packets back to back. The exponential channel randomly selects a gap between two consecutive packets according to an exponential distribution. The periodic channel maintains the same gap between each pair of consecutive packets.

The simulation was written using CSIM, a simulation package that combines the C programming language and a library of routines to provide basic simulation functionalities [7]. The simulated network testbed consisted of an arbitrarily topology were each node had the same processing capacity. In this study, the links connecting nodes were assumed to have 0 propagation delay.

For each simulated node, two CSIM processes were created: one to handle the service queue maintainer requirements, and one to handle the flow regulator requirements. Additionally, each accepted channel was implemented as a CSIM process. Each channel was configured to simulate packet arrivals in one of the following ways: bursty, periodic and exponential.

In all simulation experiments, the network edge monitoring mechanism described above, was employed to monitor and enforce the specified per channel fractional rate. Packets that exceeded this rate were tagged as Best Effort, while those that did not were tagged as Deterministic. Notice, however, the packet’s arrival time was stored in memory as a double precision number and may have been subject to some rounding error. This would cause an occasional packet to be tagged best-effort which should have been tagged as deterministic. This should have negligible affect on our results.

3.1 General Network Topology

In order to exercise the performance profile of the $\phi$-channel abstraction, a simulation experiment was run with different types of network topologies. The results reported in this paper relate to the general topology specified by Figure 3, which represents a configuration of an arbitrary wide area network. Other simulation results related to linear network configurations can be found in [9].

The simulation of the general configuration network aims at demonstrating the general behavior of the scheme and test its capability and effectiveness in handling requirements specified by the supported applications. The simulated network consisted of 6 nodes. There are multiple paths between each set of nodes. Thirteen channels with varying $\Delta$ and $\Phi(N,\Delta)$ were simulated on this network. Channels A-J required fully supported service while channels K-M required only best-effort service. Channels A, D, J, and L started at node 1, F, G, respectively. Channel M started at node 2, H started at node 3, B and E started at node 4, I started at node 5, and C and K started at node 6, respectively. The entire path of each channel is shown in in the tables that report the performance profile of the channel. The three different application traffic mixes, bursty, periodic, and exponential, were simulated on each topology.

3.1.1 Performance Profile of the General Network Topology

The simulation was performed for 500000 time units. One simulation time unit is equivalent to 1 packet service time. In all the simulations, each channel was started at a random time between 0 and 10000.

The results for three simulations with different traffic generation patterns on the arbitrary topology are shown in tables 1, 2, and 3. Each of these tables shows the results for the each channel for the following fields: number of deterministic packets generated (D), number of best-effort generated (BE), number of deterministic delivered to the destination on-time (DOT), number of
best-effort delivered to the destination on-time (BEOT),
the required level of support (RLS), the observed level of
support (OLS), the channel's path (P), the per node delay
(PND), the $\Phi(N,\Delta)$, the range of end-to-end delay values
(EED), and the average network delay (AveDel).

![Figure 3 - General Topology](image)

In some cases, there is a slight difference in values
between the number of deterministic generated and the
number delivered on-time. The discrepancy is due to the
fact that some packets were still in the network when the
simulation experiment terminated, and hence were
counted as part of the packets generated, but not those
which were delivered on-time. In all simulations, all
deterministic packets, other than those in the network at
the time the simulation terminated were delivered to the
destination on-time. The per channel observed value,
OLS, is the ratio of the total number of packets delivered
to the destination on-time divided by the number of deter-
ministic packets generated. The observed value may be
slightly lower than the true value. This is a result of those
packets that are still in the network at the time the simulation
stopped. Those packets would have been already
counted towards those generated, but not yet counted as
part of those delivered on-time, since they have not
reached their destination. For a fully supported channel
that obeys its rate, its observed value should be 1. For
channels that are not fully supported, this ratio can be any
non-negative value.

The first table presents the results of the simulation
on the arbitrary topology when channel's generate bursty
traffic. From the table, one can see that the average delay
is slightly larger than the minimum end-to-end delay.
This indicates, that each packet had to wait, on the aver-
age, for only a few other packets before it received ser-
vice. However, the simulation indicated, that at times,
some packets would be delayed close to their maximum
per node delay before receiving service. Similarly, at
times, the packet received service immediately by the
node.

The second table shows the results for the arbitrary
topology and for channels generating packets in a
periodic fashion. When comparing the results from the
periodic traffic to those from the bursty traffic, it can be
observed that the average delay value changes slightly for
the channels. In most cases, the average decreased, while
in a few cases, the value increased. This is not surprising.
The amount of traffic being generated by each channel in
the bursty and periodic simulations is nearly the same
(only differing due to the randomly selected start time).
However, packets generated by a channel in a periodic
fashion are less likely to be in the same queue as another
packet from the same channel at the same time, and so the
channel’s average network delay should decrease. This is
true for all channels that have $\Phi(N,\Delta)$ values greater
than 1 (channels D, F, and H).

The third table presents the results for the arbitrary
topology and channels generating data in an exponential
fashion. Each channel generated packets with an inter-
packet gap time drawn from an exponential distribution.
The mean of the distribution was computed on a per chan-
el basis by dividing the channel’s per node delay by its
$\Phi(N,\Delta)$. It is clear, by the number of best-effort pack-
ets generated by each channel in this simulation, that the
channels frequently exceed their fractional rates, causing
a large number of the packets to be marked as best-effort.
It is interesting to note, then, that the average delay has
increased in this case compared to the bursty and periodic
simulation results presented above while the number of
packets generated per channel has decreased. While the
results observed could be partially a function of the ran-
dom start times of all the channels, it is more likely, how-
ever, that the observed effect is a function of the order
used by the algorithm to service the packets. Recall that
all eligible deterministic packets are serviced based on
their $\Delta$'s. Furthermore, it is only when no deterministic
packets are currently eligible to be serviced that best-
effort packets can be serviced. Consequently, the slight
increase in average delay is a result of more packets being
tagged, delayed, and serviced as best-effort packets.
The excess delay is caused by the manner in which best-effort
packets are serviced. This table also shows some chan-
nels with an observed value that is larger than 1. This is
a result of the fact that the observed value is computed as
the total number of packets received on-time divided by
the number of deterministic packets sent. Consequently,
an observed value that is larger than 1 indicates that the
network is servicing more packets than necessary for the
channel. Notice that for the best-effort channels, where
no deterministic packets were generated but many best-
effort were delivered on-time, the observed alpha value is
Table 1- Performance Profile of the general topology (Bursty Traffic)

<table>
<thead>
<tr>
<th>Flow ID</th>
<th>D</th>
<th>BE</th>
<th>DOT</th>
<th>BEOOT</th>
<th>RLS</th>
<th>OLS</th>
<th>P</th>
<th>PND</th>
<th>( \Phi(N, \Delta) )</th>
<th>EED</th>
<th>( \Delta \text{Del} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>49929</td>
<td>2</td>
<td>49927</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-5</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>20.00</td>
<td>30.00</td>
</tr>
<tr>
<td>B</td>
<td>49709</td>
<td>0</td>
<td>49707</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>4-2-1</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>20.00</td>
<td>30.00</td>
</tr>
<tr>
<td>C</td>
<td>49322</td>
<td>2</td>
<td>49321</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>6-4-3</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>30.00</td>
<td>40.00</td>
</tr>
<tr>
<td>D</td>
<td>45210</td>
<td>2</td>
<td>45204</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-5-6</td>
<td>22.0</td>
<td>( \text{Not Available} )</td>
<td>66.00</td>
<td>88.00</td>
</tr>
<tr>
<td>E</td>
<td>40926</td>
<td>1</td>
<td>40925</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>4-6</td>
<td>12.0</td>
<td>( \text{Not Available} )</td>
<td>12.00</td>
<td>34.00</td>
</tr>
<tr>
<td>F</td>
<td>45266</td>
<td>1</td>
<td>45260</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>2-5-6</td>
<td>33.0</td>
<td>( \text{Not Available} )</td>
<td>66.00</td>
<td>99.00</td>
</tr>
<tr>
<td>G</td>
<td>49475</td>
<td>2</td>
<td>49472</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>2-4-6-5</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>50.00</td>
<td>80.00</td>
</tr>
<tr>
<td>H</td>
<td>59360</td>
<td>0</td>
<td>59254</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>3-4-2</td>
<td>17.0</td>
<td>( \text{Not Available} )</td>
<td>34.00</td>
<td>55.00</td>
</tr>
<tr>
<td>I</td>
<td>11308</td>
<td>0</td>
<td>11285</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>5-2-3-1-3</td>
<td>44.0</td>
<td>( \text{Not Available} )</td>
<td>132.00</td>
<td>176.00</td>
</tr>
<tr>
<td>J</td>
<td>38998</td>
<td>1</td>
<td>38995</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-4-6</td>
<td>13.0</td>
<td>( \text{Not Available} )</td>
<td>39.00</td>
<td>52.00</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>19941</td>
<td>0</td>
<td>19938</td>
<td>0.000</td>
<td>( \infty )</td>
<td>1-3-5-6</td>
<td>23.0</td>
<td>( \text{Not Available} )</td>
<td>75.00</td>
<td>100.00</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>21259</td>
<td>0</td>
<td>21236</td>
<td>0.000</td>
<td>( \infty )</td>
<td>1-3-5-6</td>
<td>23.0</td>
<td>( \text{Not Available} )</td>
<td>69.90</td>
<td>99.30</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>18350</td>
<td>0</td>
<td>18349</td>
<td>0.000</td>
<td>( \infty )</td>
<td>2-3</td>
<td>27.0</td>
<td>( \text{Not Available} )</td>
<td>27.00</td>
<td>54.00</td>
</tr>
</tbody>
</table>

Table 2- Performance Profile of the general topology (Periodic Traffic)

<table>
<thead>
<tr>
<th>Flow ID</th>
<th>D</th>
<th>BE</th>
<th>DOT</th>
<th>BEOOT</th>
<th>RLS</th>
<th>OLS</th>
<th>P</th>
<th>PND</th>
<th>( \Phi(N, \Delta) )</th>
<th>EED</th>
<th>( \Delta \text{Del} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>49929</td>
<td>3</td>
<td>49668</td>
<td>3</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-5</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>30.00</td>
<td>50.00</td>
</tr>
<tr>
<td>B</td>
<td>49709</td>
<td>0</td>
<td>49707</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>4-2-1</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>20.00</td>
<td>30.00</td>
</tr>
<tr>
<td>C</td>
<td>49322</td>
<td>2</td>
<td>49321</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>6-4-3</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>30.00</td>
<td>40.00</td>
</tr>
<tr>
<td>D</td>
<td>45210</td>
<td>2</td>
<td>45204</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-5-6</td>
<td>22.0</td>
<td>( \text{Not Available} )</td>
<td>66.00</td>
<td>88.00</td>
</tr>
<tr>
<td>E</td>
<td>40926</td>
<td>1</td>
<td>40925</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>4-6</td>
<td>12.0</td>
<td>( \text{Not Available} )</td>
<td>12.00</td>
<td>34.00</td>
</tr>
<tr>
<td>F</td>
<td>45266</td>
<td>1</td>
<td>45260</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>2-5-6</td>
<td>33.0</td>
<td>( \text{Not Available} )</td>
<td>66.00</td>
<td>99.00</td>
</tr>
<tr>
<td>G</td>
<td>49475</td>
<td>2</td>
<td>49472</td>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>2-4-6-5</td>
<td>10.0</td>
<td>( \text{Not Available} )</td>
<td>50.00</td>
<td>80.00</td>
</tr>
<tr>
<td>H</td>
<td>59360</td>
<td>0</td>
<td>59254</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>3-4-2</td>
<td>17.0</td>
<td>( \text{Not Available} )</td>
<td>34.00</td>
<td>55.00</td>
</tr>
<tr>
<td>I</td>
<td>11308</td>
<td>0</td>
<td>11285</td>
<td>0</td>
<td>1.000</td>
<td>1.000</td>
<td>5-2-3-1-3</td>
<td>44.0</td>
<td>( \text{Not Available} )</td>
<td>132.00</td>
<td>176.00</td>
</tr>
<tr>
<td>J</td>
<td>38998</td>
<td>1</td>
<td>38995</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>1-3-4-6</td>
<td>13.0</td>
<td>( \text{Not Available} )</td>
<td>39.00</td>
<td>52.00</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>19941</td>
<td>0</td>
<td>19938</td>
<td>0.000</td>
<td>( \infty )</td>
<td>1-3-5-6</td>
<td>23.0</td>
<td>( \text{Not Available} )</td>
<td>75.00</td>
<td>100.00</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>21259</td>
<td>0</td>
<td>21236</td>
<td>0.000</td>
<td>( \infty )</td>
<td>1-3-5-6</td>
<td>23.0</td>
<td>( \text{Not Available} )</td>
<td>69.90</td>
<td>99.30</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>18350</td>
<td>0</td>
<td>18349</td>
<td>0.000</td>
<td>( \infty )</td>
<td>2-3</td>
<td>27.0</td>
<td>( \text{Not Available} )</td>
<td>27.00</td>
<td>54.00</td>
</tr>
</tbody>
</table>

Table 3- Performance Profile of the general topology (Exponential Traffic)

The fourth table shows the per node utilization for the arbitrary network. From this table, one can see that the utilization for the all the nodes is between 40 and 70 percent. For bursty and periodic traffic, the per node utilization is nearly the same. This similar node utilization is a result of the amount of traffic offered to the node. In the bursty and periodic case, the amount of traffic given to each node was nearly identical, yielding similar utilization values. In addition, this helps confirm our belief the higher observed average queuing delay in the bursty simulation is a result of the traffic arrival pattern instead of some random effect. In all nodes, the utilization of the nodes in the exponential case is less than the utilization values for periodic or bursty traffic. This results from the

infinity.
fact that less traffic was generated by each channel using an exponential arrival pattern.

4. Conclusion

In this paper, we introduced a network level abstraction called φ-channel to support the requirements of real-time applications. An φ-channel represents a simplex, end-to-end communication channel with guaranteed service characteristics, between two entities. We described the basic scheme that our model uses to verify the feasibility of accepting a new φ-channel with guaranteed performance. A formal description of the proposed model and a proof sketch of its correctness was provided. In addition, the preliminary results from a simulation experiment implementing the basic functionalities of the proposed scheme and were presented. The results show that the guarantees of the supported φ-channels were preserved. Furthermore, the observed guarantees of the statistical φ-channels most of the time exceeded their required guarantees. We are currently investigating models to characterize the excess capacity of the switching nodes and ways of incorporating this information in determining the set of nodes that can support the φ-channel.

5. Bibliography


<table>
<thead>
<tr>
<th>Traffic Pattern</th>
<th>Node</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bursty</td>
<td>1</td>
<td>0.570</td>
</tr>
<tr>
<td>Exponential</td>
<td>1</td>
<td>0.531</td>
</tr>
<tr>
<td>Periodic</td>
<td>1</td>
<td>0.570</td>
</tr>
<tr>
<td>Bursty</td>
<td>2</td>
<td>0.672</td>
</tr>
<tr>
<td>Exponential</td>
<td>2</td>
<td>0.621</td>
</tr>
<tr>
<td>Periodic</td>
<td>2</td>
<td>0.672</td>
</tr>
<tr>
<td>Bursty</td>
<td>3</td>
<td>0.380</td>
</tr>
<tr>
<td>Exponential</td>
<td>3</td>
<td>0.354</td>
</tr>
<tr>
<td>Periodic</td>
<td>3</td>
<td>0.380</td>
</tr>
<tr>
<td>Bursty</td>
<td>4</td>
<td>0.572</td>
</tr>
<tr>
<td>Exponential</td>
<td>4</td>
<td>0.535</td>
</tr>
<tr>
<td>Periodic</td>
<td>4</td>
<td>0.573</td>
</tr>
<tr>
<td>Bursty</td>
<td>5</td>
<td>0.521</td>
</tr>
<tr>
<td>Exponential</td>
<td>5</td>
<td>0.481</td>
</tr>
<tr>
<td>Periodic</td>
<td>5</td>
<td>0.521</td>
</tr>
<tr>
<td>Bursty</td>
<td>6</td>
<td>0.619</td>
</tr>
<tr>
<td>Exponential</td>
<td>6</td>
<td>0.576</td>
</tr>
<tr>
<td>Periodic</td>
<td>6</td>
<td>0.620</td>
</tr>
</tbody>
</table>

Table 4: General Topology Node Utilization