Technology Challenges for Virtual Overlay Networks

Kenneth P. Birman

***Abstract*—An emerging generation of mission-critical networked applications is placing demands on the Internet protocol suite that go well beyond the properties they were designed to guarantee. Although the "next generation internet" (NGI) is intended to respond to the need, when we review such applications in light of the expected functionality of the NGI, it becomes apparent that the NGI will be faster but not more robust. We propose a new kind of *virtual overlay network* (VON) that overcomes this deficiency and can be constructed using only simple extensions of existing network technology. In this paper, we use the restructured electric power grid to illustrate the issues, and elaborate on the technical implications of our proposal.**

***Index Terms*—Critical infrastructure, distributed computing, fault-tolerance, next generation internet (NGI), quality of service (QoS), virtual overlay networks (VONs).**

I. Introduction

The basic premise of this paper is that mission-critical use of the Internet, for example, in support of the restructured electric power grid, emerging medical computing applications, or advanced avionics, will require functionality lacking in the "next generation internet" (NGI). All of these are examples of emerging distributed computing systems being displaced onto network architectures constructed from the same hardware and software components and running the same protocols employed in Internet settings. Each can point to earlier successes with networked technologies, but that depended upon special hardware, highly specialized architectures, and dedicated protocols. The challenge is to repeat and surpass these accomplishments with standard components.

Today, faced with what can only be called a revolution in networking connectivity and productivity, it has become imperative to work with off-the-shelf commercial products. Not only are older and less standard approaches unacceptably expensive, in selecting them a designer denies him or herself the best available technology, such as tools associated with building web interfaces, application-builders such as one finds on PCs, management and monitoring infrastructure, plug-and-play connectivity with thousands of powerful software products, and access to other commodity components that offer exciting functionality and economies of scale. This trend, however, is creating a daunting challenge for the designer of a mission-critical system, who will need to demonstrate the safety, reliability, or stability of the application in order to convince the end-user that it is safe and appropriate to deploy the new solution.

Traditionally, critical networked applications have exploited physical or logical separation to justify a style of reasoning in which each application is developed independently. For example, a medical computing system might be divided into a medical monitoring network, a medical database and records keeping system, a billing and paperwork system, a medical library and pharmacy system, and so forth. In hospitals built during the 1980s each of these subsystems might well have had its own dedicated infrastructure: a real-time network for the monitoring system, a more conventional one for the clinical database system, and so forth. This approach simplified the task confronting the designers because each subproblem was smaller and any interactions between subsystems occurred through well-defined interfaces.

When migrating such systems to a more standard shared network infrastructure, supported by Internet routers and protocols, applications are forced to compete for network bandwidth and switching resources in accordance with the end-to-end philosophy which governs the Internet. Protocols such as TCP are designed to be greedy, aggressively seeking the largest possible share of resources, then backing off when packet loss in the Internet signals that a saturation point has been reached. Since other applications are generally layered over TCP, they are subjected to this behavior.

TCP is a reasonable data transfer protocol for file transfer, email, and even web pages—at least after the web use becomes accustomed to the idiosyncrasies of the web. However, the unpredictable performance and extended delays that the protocol can experience are at odds with any type of "guarantee" that the application might require. Moreover, this behavior of TCP is a consequence of the connectionless, packet-oriented philosophy of the Internet. Thus, to the extent that an application implicitly depends upon isolation or other network "guarantees" for correctness, migration to a shared network—even one disconnected from the public Internet but running standard Internet protocols—has the potential to compromise safety.

Recognizing this problem, a series of reports and studies have suggested that there is a crisis in the software industry [1]. A means for supporting and validating NGI applications is urgently needed [2]. Moreover, the lack of isolation presents serious security concerns [3].

Application designers depend upon isolation to rule out *unanticipated interference*. The interpretations of "isolation" and "interference," however, vary among applications. For example, some critical applications will require security from intrusion, a property offered by virtual private networks (VPNs). We know how to build VPNs on the Internet, and the NGI will offer even stronger security because of the expected widespread deployment of public key infrastructures (PKI) and the use of security techniques to protect the Internet routing and naming protocols. If this is *all* that an application requires, it seems completely reasonable to talk about migrating it to the NGI, because such a solution would provide the necessary properties, but nothing in the pipeline offers an easy answer when isolation involves addressing other reliability goals—for example, when the need is for "virtual private" bandwidth and latency, or for network infrastructures capable of tolerating failures. In particular, although there are a number of quality-of-service proposals in the works, these prove to be inadequate when carefully evaluated against the requirements, as we will do here.

This paper considers a new networking isolation capability, termed a virtual overlay network (VON). We suggest that VONs are well matched with the need, but would be prohibitively costly to implement using contemporary technologies. However, by expanding an existing router feature and coupling it with well-understood group communication techniques, VONs could be supported at low cost, with good scalability. In what follows, we start by examining a sample application in Section II, then review the degree of match between our VON concept and emerging NGI technology in Sections III and IV. Section V presents proposed VON implementation and the associated technical challenges, and Section VI concludes the paper with a summary and review of these technology challenges. The paper leaves

Manuscript received July 2, 2000; revised March 27, 2001. This work was supported by DARPA/RADC under Grant F30602-99-1-0532 and ARO/EPRI under Contract WO8333-04. The views, opinions, and findings expressed herein are solely those of the author, and do not reflect official positions of the funding agencies. This paper is revised and extended from a less technical treatment entitled "The Next Generation Internet: *Unsafe at Any Speed?*" which appeared in the August 2000 issue of *IEEE Computer*.

The author is with the Department of Computer Science, Cornell University, Ithaca, NY 14853 USA (e-mail: ken@cs.cornell.edu).

Publisher Item Identifier S 1083-4427(01)05293-6.

open as many questions as it addresses, and hence should be seen as a technology proposal emerging from a serious problem with the current Internet and NGI, but not a definitive answer.

II. Need for VONs in the Restructured Power Grid

The restructuring of the electric power grid creates a wide variety of technical challenges associated with operation of the power grid, response to changing loads or failures, protection and dissemination of information within the grid both for management, and also for pricing and commerce. These highlight the broader problem of interest to us here.

For example, the *load following* problem involves dynamically adjusting power production to match demand. Today, this is accomplished by monitoring line frequency. If generation exceeds consumption, the frequency will rise above 60 cycles, and if generation is inadequate, frequency falls. Moreover, frequency is a global property of the grid (within any well-connected region). Thus, with little or no computer-to-computer communication, the industry can exploit the grid itself to dynamically match power production to demand: load-following generators simply increase or decrease power production, working to maintain line frequency at 60 Hz, but the approach assumes that within any region, a single company provides all the power and owns all the load-following generators.

A major goal of restructuring is to support new ways of buying and selling power. Suppose that a producer enters into a load-following contract directly with a consumer—an option restructuring will create. Implementation of this capability is not possible using the traditional method, since changes in line frequency occur throughout the grid, giving no information about the specific sources of the loads. Some form of dedicated intranet within which the producer and consumer can exchange information about load and production is needed. Utilities would use this both to implement load following and for other purposes such as protection and setting pricing. As noted before, we can assume that this intranet would run Internet protocols, although it would not be connected to the public network. The question is to identify a way to implement a load-following capability over a dedicated Internet-like system.

Unfortunately, even a dedicated, isolated intranet would behave much like the public Internet if used on a large scale. Just as in the public Internet, a power-grid network would presumably see a large volume of data transfer traffic, file transfers, web activity, and so forth. The underlying TCP rate control mechanisms would presumably be the same ones used in standard network settings, and the routers would be off the shelf. Thus, just as one can *usually* access *The New York Times* on the Web, but may be denied access from time to time, any mechanism that the electric power industry might build has the potential to encounter Internet-like limitations. Worse still, the Internet infrastructure can easily be attacked using methods like the denial of service attacks seen in early 2000.

Congress expects the industry to solve this problem, but doing so involves finding a way to superimpose a better-behaved power systems network on the Internet, with all its limitations. If we knew how to do this, the technology would be popular. While it isn't hard to design a variety of mechanisms in support of load-following contracts, were one to run such a protocol on the Internet, it would be exposed to disruption in the event of network overload, failures, mundane events like routing table updates, or even terrorist attack. Yet the application clearly requires a degree of robustness commensurate with the critical role that electric power plays in modern society and industry. In effect, while the development of suitable load-following protocols for isolated, dedicated networks is not necessarily all that hard, running them in a shared off-the-shelf network based on the technologies that underlie the public Internet is problematic. Other similarly challenging problems can be enumerated if one considers additional aspects of the restructured grid: protection against load surges or line failures, relay control, voltage control, tracking the parameters used in setting power pricing, and so forth.

What stands out in settings like this is the manner in which the Internet itself makes an otherwise straightforward problem difficult. One could easily imagine designing a dedicated Internet for the power grid with adequate redundancy, so that no two control computers would ever be "disconnected" from each other—there would always exist at least two disjoint paths between any two points. Using state-of-the-art fiber optic channels and switches, such a network could be designed to have vastly more capacity than control applications could possibly desire. Naively, one might easily believe that merely doing so would address the problems just cited. Indeed, this seems to explain why Congress believes that the industry can succeed.

Counterbalancing this observation, however, are the pragmatic considerations that emerge from the protocols that control and operate the Internet. Presumably, some power applications will transfer large data files; others will use bandwidth aggressively for monitoring and control purposes. Faced with contention for limited (even if not scarce) resources, the Internet's bandwidth-greedy protocols are designed to cause router overload in such situations, since packet loss and delay are precisely the mechanisms that trigger TCP flow control. Moreover, redundancy won't give us fault-tolerance. Internet routing takes no advantage of dual routes and changes in routing propagate slowly. While a router is down or a link is down, continued attempts to use the old route may result in periods of disconnection that would last for many minutes—*even if an alternative route is available*, and we haven't even touched on protection from deliberate attacks. Thus, no matter how the power grid network is designed and provisioned, the requirement (apparently, unavoidable) that it be constructed in a standard way using conventional components denies the industry any chance to benefit from this investment. In all likelihood, the dedicated "powernet" will suffer from the same behavior, the same lack of security, and the same flaws as does the public Internet on which it will be based.

The power grid is just one of many kinds of critical applications that are being deployed now, or will be in the near future, and that share very critical requirements. Some, like military command and control applications, air traffic control, or distributed medical computing systems, have life-critical implications. Others, like financial systems or the electric power grid are critical in nature even if the consequences of failures or disruption may not be life-threatening, but the pattern is similar. The problem is not so much that we cannot build the necessary protocols, making assumptions that would be valid in isolated, dedicated settings, but rather than were we to build them, we could not run them with confidence in a shared network populated not merely by competing applications but even potentially malicious intruders.

Our challenge is to enable coexistence between critical kinds of applications that share a single, finite capacity infrastructure. If we can solve this basic problem, we make it possible for the application developer to address each issues that arises in a complex application separately, as if the associated protocol would run on a dedicated, completely idle network. The protocol would then deployed in a shared environment that provides a virtual form of isolation, allowing us to present the one physical network as multiple virtual ones, and thereby enabling the application to run without risk of interference. Our proposal, which we call a VON, construes isolation broadly—the objective is to provide *whatever aspects of a dedicated network may be important to the correctness of the desired application*. In contrast, traditional virtual private networks provide isolation and security, but at a high cost visible to the application, and without guaranteeing any minimum level of performance.

Underlying this approach is our belief that rigorous solutions to complex distributed computing problems typically revolve around assumptions about the behavior of the network. Today, these underlying assumptions are often implicit, and one often demonstrates the adequacy of the solution by simply running it experimentally and injecting faults. Such demonstrations are only as good as the coverage of one's experiment. A way to make these implicit requirements explicit, and then to enforce them is needed. Our virtual private network mechanisms assume that the user knows what the protocol requires, but lacks a way to ask for a network with the necessary behavior.

III. Overlay Networks and Virtual Overlay Networks

We are using the term *overlay network* (ON) to describe a configuration within which a base network is used to support some second network, "layered" upon the underlying infrastructure. Each ON has associated with it the following.

- A globally unique identifier, with which traffic within that ON can be tagged. Routers use the identifier to associate traffic on different ONs with the resources reserved for use by that ON.
- Some set of access points. One can imagine the ON as a sort of virtual Ethernet, in the sense that it offers service at multiple locations, and the resulting applications share what is logically a single infrastructure.
- Some set of guaranteed properties. Notice that these properties will usually relate to the aggregated traffic on the ON and not to point-to-point paths within the ON.

An ON is intended to offer a minimal set of very basic guarantees, which may be very simple compared to the desired application-level properties. Given an underlying ON with a sufficient set of raw or base properties, it will often be possible to build layers of software that extend the ON properties into new, stronger ones. We will call such a network a VON because it refines the base properties of an ON with new properties that *appear* to hold for the underlying network, but are actually implemented on behalf of the user by means of some sort of protocol or algorithm. In general, our goal should be to employ the simplest ("weakest") ON that would still support the widest range of VONs, although this paper will not actually solve the problem of identifying such a minimal ON. Instead, we simply observe that given an ON with "sufficiently strong" guarantees of minimum bandwidth and packet loss, which secures its own infrastructure against failure and attack, it should be possible to build almost any desired property over this, provided only that the basic speed of the ON is adequate to support the desired speed of the VON even with this intermediary software in use.

Notice that VON is somewhat like an abstract data type. In this perspective, the ON can be imagined as the base type from which other types are derived by refinement. A VON is created by instantiating an ON or VON but replacing some of its methods with modified ones that change the behavior of existing methods, or extend the interface with additional operations. Were our VON proposal ever widely adopted, we believe that a natural way to present VONs to the application developer would be through ADTs available in the runtime environment of the application, much as Java treats aspects of the Java runtime environment. The author is not aware of programming language research in which the network itself is treated as an ADT, but the step seems like a small and natural one.

IV. Next Generation Internet

The NGI is a work in progress, and there is much debate about the best way to achieve even the widely accepted goals. Nonetheless, most of the technologies that will be widely available within the next few years already exist. Accordingly, this section speculates about the most probable evolution of the network. Our goal is to understand the adequacy of the NGI for hosting applications needing VON capabilities. Could one build an ON, and use it to build VONs, over the NGI? We focus on performance, security, and quality of service.

A. Achieving Higher Performance

It is certain that the NGI will be faster. Widespread deployment of broadband technologies and optical fiber is expected to yield a ten- to 100-fold improvement in the performance of the Internet. However, performance alone says nothing at all about isolation from undesired interference. In particular, the contemporary Internet is much faster than the Internet of a decade past, yet interference problems are, if anything, much more in evidence. The popular press routinely conflates speed with safety, but while speed is often necessary for safety, it rarely suffices.

B. Improving Security

The NGI will also have much better security properties than does the current public Internet. At the level of the network infrastructure, a security architecture is already being deployed. At the application level, the NGI will see widespread availability of *virtual private networks* (VPNs). A VPN is a software abstraction overlaid on a shared network, in which communication between end points belonging to the VPN is signed (for authentication), encrypted (for secrecy), or both. Given a key infrastructure, a VPN offers a way to implement one form of VON—but only one form, and only one at a time; a given machine can only belong to a single VPN. Indeed, a VPN is similar to a firewall, except that whereas a firewall acts only at the periphery (where packets are filtered), a VPN acts at the network interface of each attached computer.

In our own work, we have explored the extension of VPNs into a form of VON focused exclusively on security issues [5]. Called a dynamic virtual private network (DVPN), our solution permits a single machine to belong to multiple VPNs, and provides fault-tolerance. We also provide protocols for rapidly changing security keys when the set of participating computers or applications changes. DVPNs are limited in some respects. For example, our work does not allow services like file systems or databases to reside in multiple DVPNs, and has no support for interDVPN communication; any serious DVPN implementation would need to re-examine such design decisions. Nonetheless, it seems reasonable to assert that by extending the DVPN concept one could solve the infrastructure security needs of ONs and VONs.

C. Options for Supporting Isolation

Although expressed in terms of isolation, one can also understand an ON or a VON as offering forms of many-to-many quality of service. This creates an apparent match between trends in the Internet *QoS* domain and the needs of critical applications such as the ones we surveyed. Internet *QoS* research seeks to overcome the erratic performance of the current Internet, which represents an obstacle to migration of telephony from the current dedicated infrastructure onto a packet-based one.

Quite a few approaches to supporting quality of service have been advanced. Among the most prominent are RSVP, RED, and RIO, and a family of approaches called Diffserv [6]–[9]. All of these focus on providing guarantees of a type needed in telephony, and all are oriented toward one-to-one communication paths (some work has been done on one-to-many and many-to-many extensions of these methods, but consensus has yet to emerge).

QoS solutions can be partitioned into two categories. The first category, exemplified by RSVP, operates by reserving resources along the route from source to sender for a period of time. The second cate-

gory operates by marking packets at the time they enter the network. The idea is that any resource reservation is done abstractly, by a service that tracks overall commitments and only accepts new requests if routers should have the capacity needed to accommodate them. Incoming packets are tagged as "in profile" or "out of profile," and routers preferentially target out of profile packets when overload occurs. Clark and others have shown that this approach yields excellent statistical guarantees while avoiding a costly traffic classification problem within the network. Specifically, when using RSVP or similar schemes, if there are n endpoints in the network, a typical router may be asked to manage $O(n^2)$ connections. The lookup needed to classify each packet passing through the router (to provide it with the appropriate form of service) becomes expensive, perhaps prohibitively so, and the use of resources may be inefficient. Diffserv, while offering slightly weaker guarantees, avoids this costly lookup—a packet is classified at the edge of the network and the router need only concern itself with a single bit.

It should be stressed that the properties offered by this approach are probabilistic. For example, if network load or routing fluctuations cause network latencies to change suddenly, messages that were admitted at a steady rate (and within the sender's profile) might arrive in a burst, perhaps exceeding the profile of the sender and overloading a router on the path. In fact, one can identify any number of cases in which Diffserv would violate its guarantees. The argument advanced by the Diffserv research community is that best-effort quality of service is often adequate. Moreover, they see the approach as being more philosophically attuned to the prevailing style of the Internet community. It has an end-to-end feel to it, whereas RSVP appears to superimpose virtual circuits over the Internet's basic packet structure.

However, rather than debate such issues, we should be asking ourselves whether either category of *QoS* mechanism is suitable for implementing ONs. It seems that the answer is negative, in consequence of the need for an ON to emulate a dedicated network resource that may have many users. Imagine a power monitoring system configured to use a 10-Mbit network accessible at 100 endpoints, for example. We now wish to run on an ON implemented using a *QoS* mechanism as the underlying building-block.

It is quite possible that the burst load associated with any particular pair of endpoints (a source sending to some destination) could reach the performance limit of the ON. Thus, in our example, the ON must guarantee an aggregated bandwidth of 10 Mbits/s, but be capable of providing peak bandwidth to any pair of users that happen to generate the full load at a time when the ON is otherwise idle. A connection-based resource reservation protocol might therefore be forced to reserve $O(100^2 * 10$ Mbits$)$, or 100 Gbits, of capacity at a central router. Clearly, such a naive implementation of resource reservation would be unsatisfactory.

One can speculate about various options for improving our naïve solution. For example, we could dynamically allocate resources to endpoint-pairs, so that at any point in time, the aggregated allocation doesn't exceed 10 Mbits/s, but with the allocation shifting around as needed. Thus, if at time t_0 process a sends 5 Mbits/s to process b, the remaining 5 Mbits would be allocated to other uses. At time t_1 perhaps process a would stop sending and process b would attempt to consume the full 10 Mbits, triggering a reallocation of resources, but this would clearly entail a very sophisticated mechanism. Shifting the allocation of bandwidth will take time, and requires a form of distributed coordination protocol that is hard to support with fault-tolerance guarantees. We rule out this solution as overly complex.

Similarly, one can imagine periodically multicasting the activity profile of the endpoints so that all endpoints have a reasonably current picture of the pattern of usage within the ON as a whole, but this solution suffers from an n^2 growth in overhead messages. As we scale the size of the ON up, the number of such messages will grow at least linearly in the number of endpoints, and each is a $1 - n$ multicast. Here, the multicast could be implemented using a simple scheme because reliability is probably not critical, but the background load looks intolerable. Moreover, the higher the quality of data needed, the more overhead we pay. With any lower quality of data, the quality of profile marking or prioritization may become unacceptably poor.

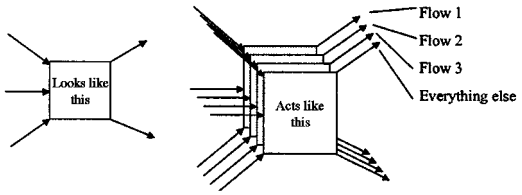

Fig. 1. Router flow partitioning.

Rather than continue on this line of speculation, however, we advance a stronger conjecture: that *any* VON solution based on a point-to-point reservation model or quality of service mechanisms will be intrinsically costly and scale poorly. Furthermore, notice that both of the *QoS* mechanisms that seem most popular as candidates for the NGI are basically probabilistic. Neither works well if the network itself comes under stress, changes routes, or experiences a failure. Using these mechanisms to implement an ON is obviously going to be a very difficult problem, at best. If our hypothesis is correct, the NGI is unlikely to represent a very friendly environment for applications seeking ON mechanisms.

V. Alternatives for ON and VON Support

The considerations just cited suggest that the NGI will be faster and more secure, but not "safer," if the safety of critical applications requires VON-like functionality. However, such an outcome could be avoided. We now present an informal proposal for an alternative way of building VONs that seems to be within the reach of current technology, and responsive to the need.

A. Using Router Partitioning to Implement ONs

Existing routers support router partitioning, whereby one internet service provider (ISP) can supply networking connectivity to another, with guarantees of minimum throughput and priority. To use router partitioning, a provider configures the router to set aside some percentage of its resources (Fig. 1), dedicating these to a particular flow. Each incoming packet is classified by means of a flow identifier and treated as if it resides entirely within a virtual router defined by the resource subset allocated on behalf of that flow. For example, if Global Domination Networks (GDNs) leases one-half the capacity of some router to Gotham Network Solutions (GNS), then each GNS packet would be labeled with the same flow identifier and the GNS routers on the network routes associated with the lease would set aside half of their resources on behalf of GNS. This is done in a way that virtualizes the router on behalf of GNS. One can even imagine a scenario in which GNS packets are dropped (due to congestion) although the GDN "side" of the same routers is idle.

It may seem that if GDN has excess resources available, it should in general offer them to GNS, but upon further reflection, it becomes clear that dropping packets when GNS reaches its contracted-for traffic level is necessary if the solution is to work correctly. The problem is that protocols such as TCP are designed to back off as congestion starts to occur. Suppose, hypothetically, that GNS was allowed to expend 20% more bandwidth than it had contracted for simply because GDN has excess capacity. TCP will now be operating at a rate that represents a substantial overload. Now, imagine that traffic surges on the GDN side,

forcing GDN to cut GNS back to its contracted-for profile of resources. The TCP protocol, rather than seeing "early warning" of impending congestion will suddenly experience high rates of packet loss, and a brief period of disruption is very likely—one that would have been avoided if GNS was limited to the profile of services it paid for, in the first place. We see, then, that for an virtualized network to operate correctly, it needs to provide a faithful imitation of a dedicated network even with respect to the limitations of such networks.

Notice that router partitioning yields a kind of ON, implemented by one ISP on behalf of another ISP. All of the traffic flowing through GDN on behalf of GNS seems to live within a single shared ON, competing with other traffic in the same ON, but not influenced in any way at all by traffic originating in other flows.

Our proposal starts by imagining that we extend this existing mechanism into one that can support perhaps thousands or tens of thousands of flows, but otherwise behaves in the same matter. In general, each ON will serve some large number of endpoints. For example, power systems applications for a large region could include several hundred monitoring devices, all sharing a single ON, with a single ON flow identifier. This is in distinction to the connection-oriented perspective that one sees in existing quality of service proposals, such as the ones reviewed earlier.

B. Building VONs over ONs

Power applications and similar critical networked applications will want more than dedicated bandwidth from its ONs. A network used to control devices such as protection relays within the power grid should be reliable even if links or routers crash, suggesting that the physical configuration of the resources used to construct the ON may need to be carefully controlled (for example, we might require redundant routes between all pairs of endpoints).

Broadly, we should imagine a world within which the core properties of the ON can be strengthened by layering network management and control software over the endpoints of the ON. For example, consider the following.

- A VON might require specialized routing.
- We may wish to authenticate the connection of a computer to the VON, and assign it a specialized session key for use while communicating in the VON.
- We may need to refresh these session keys periodically, or in cases where a computer leaves the VON and is no longer trusted.
- We may wish to coordinate the signing or encryption of data.
- We may need to monitor loads and adapt to overload.
- The VON may require some special protocol to ensure reliability, such as the duplicate transmission rule outlined above.

Abstracting, we can say that a VON resembles what the distributed computing community call a "process group." The desired group communication properties would depend upon the needs of the application, and (in distinction to classical work on distributed computing) the network itself (ON) would offer a basic level of guarantees, reflecting the dedication of resources on its behalf. This last point is important, because it offers a foundation over which much stronger properties can be guaranteed.

For example, consider the TCP protocol. In a conventional network, one cannot really assert that a TCP connection is "reliable," since such a connection can potentially break in the event of an infrastructure disruption, even if neither endpoint has failed. With a TCP connection within a VON, the identical TCP protocol might be able to offer a stronger guarantee, such as the guarantee that if no more than one network link or router fails, the TCP connection will never break unless one of its endpoints crashes. This specific guarantee would require a degree of redundancy within the ON, but one can imagine all sorts of low level guarantees, translating to all sorts of higher-level properties visible through TCP and other protocols.

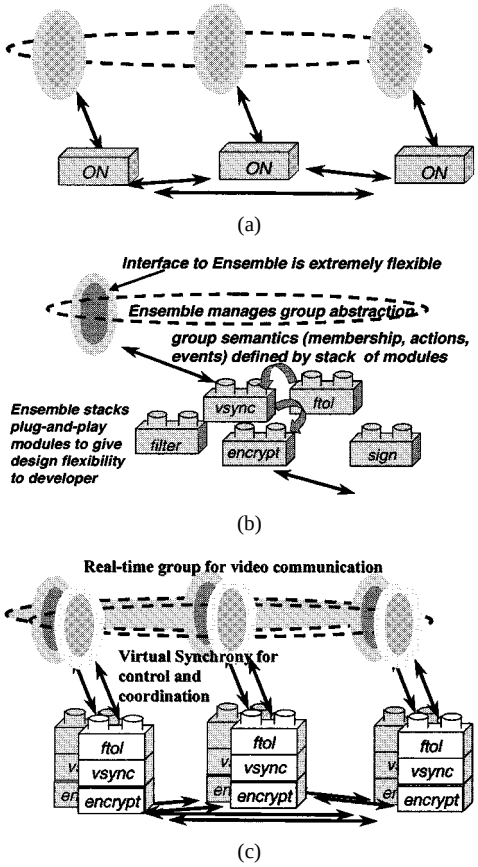

Fig. 2. ON viewed as an abstract type. Stacking microprotocols in Ensemble. Multiple VONs used in a single system.

Generalizing from this example, we can see that one builds a VON up from an ON by successively abstracting—layering one or more software-implemented abstractions over the ON, so that each layer extends or strengthens the properties of the stack of layers and ON below it. This is a way of building distributed protocols that has become popular over the past decade. It was introduced as the "streams" architecture of the Unix system then generalized by the x-Kernel [14], and adapted to group-structured applications in our work on the Horus [10] and Ensemble [11] systems. Recent work has shown how to use formal methods to reason about, optimize, and prove properties of systems structured in this manner [12], [13]. Fig. 2 illustrates the idea. In Fig. 2(a), we see the endpoints of a VON treating the ON as an abstract type. Fig. 2(b) illustrates the idea of stacking new subtypes (microprotocols) to produce new VONs with the same interface but transformed properties. Fig. 2(c) shows the result, in this case, with two VONs in use.

Notice that we've made a leap here from talking about TCP, which is a point-to-point protocol, to talking about groups of endpoints representing the virtual access points to a VON, and finally to calling these set of endpoints a form of process group. The idea is that even if the applications are unaware that we view them as participating in a process group, we might still station some small chunk of code next to the application to manage some aspects of the network connection it uses. For example, in implementing the DVPN architecture mentioned earlier, we placed a small key management application close to each DVPN participant; it was responsible for obtaining and tracking the DVPN key, which changed dynamically, and then rekeying the DVPN network driver using each new key as that key became available. Similarly, these

code stubs might take messages sent by the application and send more than one copy—perhaps one copy on each of two independent network routes. In general, any form of control involving replicated data, replicated security keys, or coordinated actions can be implemented by an appropriate process group sitting next to the application and managing some aspects of its environment.

Of course, we also have the option of making group communication more directly visible to the application. One could do so by replacing the standard IP multicast implementation with one that uses the identical interface but sending multicasts using the group communication tool. Indeed, once one has this adopted this basic approach, it becomes clear that there are a great many things we might do on behalf of an application, to actually *guarantee* the properties that might traditionally have been implicit but critical.

Given this perspective, it becomes possible to import a substantial body of knowledge about group communication into the VON arena. The author, for example, has worked on the following four styles of group communication system:

- traditional best-effort group communication environments;
- virtually synchronous group communication, with carefully managed group membership and multicast facilities (in addition to traditional point to point mechanisms of the sort normally seen in the Internet);
- secured group communication systems, providing authenticated join and keying both for the group as a whole and for point-to-point connections, as needed;
- probabilistic group communication, offering bimodal multicast and probabilistic membership tracking, for use in settings where scalability and stable throughput even when failures occur take precedence over absolute logical guarantees.

Each of these could be understood as the infrastructure one might use in supporting some class of VONs. Moreover, this list illustrates just a few options within a spectrum of possible VONs. Others include support for partition tolerance or mobility, specialized realtime guarantees, other types of security architectures, specialized protocols for video or other media tranmission, infrastructures which integrate management of the VON into application-level mechanisms such as automated restart facilities, and so forth.

The key to our proposal is the idea of successively strengthening core properties by layering software over more basic VONs and ultimately over the underlying ON, with its strong isolation and resource allocation guarantees. Lacking these basic guarantees, one could run the same sorts of protocols over the Internet, but would generally not arrive at the desired outcome, because the base case for proving many sorts of properties revolves around the properties of the underlying ON, and the isolation of the ON from external interference. Given an ON with even weak isolation properties, however, one can often find ways to strengthen them. In general, one should expect tradeoffs, such as paying communication overhead to gain higher reliability.

Notice, however, that not every form of ON can be used to support every possible VON. An ON lacking a secured infrastructure may be intrinsically insecure and hence fundamentally inadequate for supporting certain types of secured VONs. An ON that can be severely disrupted by a single router failure would be unable to support a VON requiring continuous throughput even when routers crash. Among the many open questions about VONs, issues such as these stand out as requiring further study to determine the ultimate feasibility of responding to the apparent need.

C. Building Better ONs

This observation motivates us to ask what properties ONs will need to have, and to what degree the underlying Internet mechanisms are adequate to provide them.

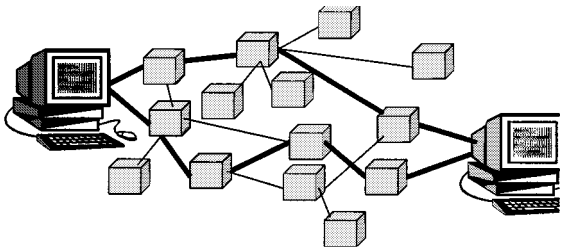

Fig. 3. Failure-independent routes in a network.

Were we merely to support the extended router partitioning scheme described above, and to combine this with IPSec mechanisms to secure the routing and DNS infrastructure, the resulting ONs would support bandwidth isolation, but the degree of fault-tolerance against router failure and link failure would be limited. Specifically, while a router or link failure will eventually trigger a change in route (assuming that the network remains connected), the temporal properties of Internet routing protocols limit the speed of adaptation. It is likely that an extended period of disrupted communication would ensure, lasting for many seconds or even minutes. Accordingly, our basic proposal is inadequate to support applications for which stronger guarantees are needed.

It can be seen that for certain classes of applications, namely those needing stability even while a router or link failure is being detected and the necessary adaptations are still being performed, it will be necessary to provide redundancy. At the time of this writing, one would normally do so using what is called a dual-IP naming architecture. In this approach, each computer is assigned two IP addresses, and the physical network infrastructure is duplicated so that each machine has two or more network interfaces. The intent of such designs is that even if machine A becomes unreachable for a period of time on one network, it may remain reachable on the second network under some other name: A' (Fig. 3). A difficulty with dual-IP architectures, however, is that the existing Internet routing protocols lack provision for creating multiple, independent routes. Thus, *even if the capability of doing so exists*, existing network routing protocols could easily chose to route messages from A to B and from A' to B' along the identical path, defeating the intentions of the dual IP configuration. Additionally, even if one had some form of guarantee that dual IP addressing results in disjoint and failure independent paths between the respective address pairs, one could question whether this is the best way for the endpoints to take advantage of the dual network. It is awkward to deal with processes using multiple names; a more natural way would be a networking infrastructure within which process A is always called A, where the infrastructure itself deals with dual addressing, redundant router paths, and the necessary mechanism.

Such a vision leads to the following proposal. Suppose that there were a way to deploy hardware redundantly, and that given an adequately redundant network, a way to obtain multiple disjoint paths between each pair of endpoints (and, for multicast applications, a way to obtain multiple, failure-independent routing trees). Then an ON might take advantage of this infrastructure by automatically duplicating packets, sending each packet once on each route (and similarly, for multicast routing). The potential for packet duplication is a property of the current Internet, so one can dispute the value of duplicate suppression within the ON, but if desired, this too would be a possibility.

The idea of supporting disjoint redundant routes represents, we believe, an exciting direction for further study. Existing Internet protocols suffer because both negative acknowledgement (NAK) packets and retransmissions typically follow the same routes that are followed by the initial data packet that was lost. When a router becomes overloaded and begins to drop packets, this means that recovery mechanisms subject

TABLE I
TECHNOLOGIES REQUIRED IN SUPPORT OF ONs AND VONs. EXTENSIONS THAT SUBSTANTIALLY EXCEED WHAT THE NGI IS ALREADY EXPECTED TO INCLUDE ARE INDICATED BY AN ASTERISK

Feature	Extends... (Existing Technology)	Observations
Packet marked as ON traffic	In or out of profile bit	Internet packet header has available bits, although many proposals are competing for them
**** ON flow id***	RSVP connection ID	While the IP header lacks an appropriate 4-byte field, the flow id is needed only when the ON traffic bit is set, hence this can be accommodated as the first 4 bytes of the payload for ON packets
**** Router classification task, resource reservations***	RSVP resource leases, router flow partitioning	The form of router flow partitioning we propose is similar to an existing router feature, but would need to support far more flows. Routers would genuinely set-aside resources on behalf of these flows.
Manage reservations	Diffserv reservation manager service	As in Diffserv and RSVP, a new flow must establish a lease on router resources needed, and renew that lease periodically.
****ON fault-tolerance***	Dual-IP networks, but these are inadequately supported in the existing Internet	Unlike existing Internet routing, a fault-tolerant VON would require fault-tolerant ON infrastructure support: the ability to create router- and link-disjoint independent paths between each pair of senders, and a mechanism for sending duplicate packets down those paths.
Authentication	Public Key Infrastructure	Emerging PKIs have rich feature sets and can be used to implement any desired authentication mechanism
Basic services	IP, TCP, IP multicast	A VON supports the normal Internet protocols, so these would run without change. Even so, these standard protocols might offer stronger guarantees emerging from the properties of the underlying VON, such as real-time properties or the guarantee that a TCP channel won't break unless an endpoint crashes.
**** Extended services***	*New*	A VON would often offer additional "new" communication protocols, such as scalable reliable multicast. The protocols supporting these new services would be implemented and proved correct by drawing on the guarantees of the underlying ON. They can be seen as offering a contract to the application: "if you run this protocol on such-and-such a VON, you will be guaranteed the property that..."
**** VON management***	*New*	As a well-defined subnetwork residing within a shared Internet, a VON offers the possibility of sophisticated management, such as monitoring services, security policy enforcement, intrusion detection and response, load monitoring and overload management, etc.

that router to additional stress, unless the protocol does some form of flow control, as in the case of TCP. The Internet, then, is designed to suffer degraded performance during overload, in part, as a consequence of routing.

However, with redundant edge- and router-disjoint paths betweens source and destination, when a packet is lost on one path, one can imagine doing recovery on the other, potentially noncongested path. Indeed, a protocol could potentially switch from a congested to a noncongested route dynamically as an alternative to choking back in the manner of current congestion-control algorithms. Such options are especially appealing when one considers reliability in multicast protocols, where large-numbers of receivers may depend upon steady data delivery: forward-error-correction methods, for example, used on disjoint paths could be of great power and might tremendously improve the reliability of even the existing unreliable multicast protocols.

Space constraints on this paper prevent us from exploring the options for building routing protocols with the desired structure. This would not be a trivial undertaking. Current protocols are designed to track shortest paths rather than independent paths, and any solution would also need to be dynamic, adapting to failures and recoveries. Yet so much is known about routing that it would be surprising if such a problem were to prove intractable.

VI. TECHNOLOGY SUMMARY

Table I summarizes the VON proposal. In this section we briefly recap the various aspects of the proposal with an emphasis on aspects that would go beyond what existing technologies currently do.

1) Packet Marking and Classification: Our proposal requires that routers be capable of identifying packets belonging to ON flows and of matching the packet to the associated resource reservation. For this purpose, we envision a single bit, per-packet, marking it as an ON packet (the IP header has room, although there are competing proposals that would use the remaining bits), a 4-byte identifier drawn from a global space of ON identifiers (this could be placed behind the IP header since it would be needed only if the bit is set), and, of course, the mechanisms needed for the router to do the necessary classification.

As seen previously, the major criticism of most router classification mechanisms is associated with their cost, both in terms of the compute time expended within the router, and also the manner in which resource reservations scale as a network grows larger. Unlike RSVP, which is the most widely criticized in this respect, our proposal would require modest numbers of ONs, because the number of ON configurations needed by an application tends to be small even if the size of the application is large. An ON is like an entire network and not like a

point-to-point circuit. Moreover, existing routers already do a similar classification on a small scale. Accordingly, while recognizing this as the most challenging aspect of our proposal, we consider it quite feasible.

Finally, we note that our VON concept offers ISPs a cost-effective way to respond to the *QoS* needs of customers and to bill for the associated services. Just as large common carriers have little difficulty billing one-another for leased resources and accounting for them, our proposal (which works in the same manner) simply scales a working facility up, in a relatively straightforward manner.

2) Resource Reservations and Management: Like Diffserv and RSVP, we do require some degree of resource management to track reservations and grant requests. Diffserv does this entirely abstractly, using centralized servers that maintain a theoretical model of resource use and grant or deny requests based on their theoretical impact on the router. Earlier, we saw that this works in a statistical sense, although network dynamics can cause the guarantees to break down when a fault occurs or a transient overload significantly changes packet latencies in ways that could cause a burst load on a router. The RSVP approach is more mechanical. Each router on a given point-to-point path is asked to set aside resources, and the resulting lease must be renewed periodically to ensure that the router will not release the associated resources for other uses.

Our proposal combines elements of both of these. Like Diffserv, we envision certain global resource management tasks, such as management of the pool of ON flow identifiers, which must be globally unique. On the other hand, the routers themselves must set the resources associated with each ON to the side, which would require a leasing mechanism similar to the one in RSVP. We presume an ON management service that would operate somewhat like the one for Diffserv, first checking the request against an approximation of the network state, then for apparently acceptable requests, but then performing a two-phase commit to actually allocate the desired router resources. It would seem that such a mechanism can be distributed using a form of loose consistency that is easily implemented (somewhat like the DNS); hence, we see no difficulty associated with this functionality, nor does it pose any major scaling challenges. Like RSVP reservations, ours should probably be implemented as leases which need to be refreshed periodically.

3) ON Fault-Tolerance Issues: We have seen that a fault-tolerant ON would need to offer path-disjoint routes between pairs of sources or, in the case of multicast, path-disjoint multicast routing trees. Although nontrivial, such a vision accords with what one builds when configuring a traditional LAN to support dual IP routing. Indeed, as discussed previously, one could argue that the handling of dual IP addressing is currently defective, since even if multiple IP addresses are used, one has no guarantee that routes between disjoint addresses will have independent reliability or latency properties. Our approach would yield substantial new options for reliability even when using traditional Internet protocols such as TCP and IP multicast, while also creating a building block of great power for use in higher-level protocols.

4) Authentication: Although our approach assumes that one could authenticate membership in VONs, we see nothing about this that couldn't be solved using emerging public key infrastructures and certification authorities. Even group keying, which involves maintaining a key shared among the endpoints of the VON and refreshing that key as membership changes, is a well understood and largely "solved" problem.

5) Basic Services: A VON would support the same basic services as a normal Internet including the following:

1) DNS;
2) routing;
3) IP;
4) UDP;
5) TCP;
6) IP multicast.

6) Extended Services: Here, we would draw heavily on the group communication mechanisms cited earlier both to offer the end-user access to new kinds of multicast and communication protocols, and also to manage the periphery of the VON in such a manner as to "precondition" traffic where the VON properties require it (for example, if an application must be forced to respect a real-time property for which it has contracted).

Earlier, we noted that it would be appealing to think of a VON as a new form of abstract data type, instantiated from a class within which the ON is the base element, and with each VON extending the ON or VON from which it was derived.

7) VON Management: The fact that a VON has a well-defined set of endpoints and mimics some form of dedicated network makes it appealing to think about new kinds of network management tools, similar to the ones used in modern LANs, but extended for management of the VON. These might include load and other performance monitoring mechanisms, intrusion detection services, facilities for reconfiguring the VON when conditions change or an attack is detected, and so forth. Network management is, today, a costly problem that grows difficult as a network grows to include Internet links; with our VON proposal, each VON has predictable behavior, making it far more reasonable to talk about detecting deviations from intended profiles of use or load, and, hence, much more reasonable to talk about managing the network in a proactive manner.

VII. Conclusion

We have sketched an ambitious proposal to redesign the Internet through a series of relatively modest interventions, but with significant implications. As seen in Table I, our proposal is constructed from existing mechanisms, although extended in nontrivial ways, and with significant new properties that result from the extensions.

Fleshing out this proposal into a full-fledged technology would be a major undertaking. However, the primary intention of the author is to identify an unrecognized requirement upon the NGI. We do wish to argue that the problem is tractable, but not that we solved the problem, and far more work would be required to do so. The contribution of our study is to suggest that critical applications often depend on a form of isolation, and that, as things stand, the NGI will not support. While we do suggest that VONs could be constructed and would respond to the need, we also identified a number of technical challenges which would need to be overcome before a working VON could be deployed. These include support for a router resource partitioning mechanism, implementation of a service to manage router bandwidth and other resources in support of reservations, new routing mechanisms supporting redundancy, and the associated management infrastructure. Our proposal, then, would require a considerable research and implementation effort.

As things stand, developers of critical applications will be challenged by insurmountable barriers. Yet it is within our technical reach to solve this problem, and this can be done in a way respectful of standards that builds on existing Internet router capabilities using services similar to ones already in the pipeline. It would be a great shame if the NGI fails to do seize this opportunity to create a better communications infrastructure.

Acknowledgment

This paper was based on a colloquium talk presented at various locations from 1998 to 1999. The author is grateful to the members of the audience at these talks, who gave extremely helpful feedback.

REFERENCES

[1] W. Gibbs, "Software's chronic crisis," *Sci. Amer.*, vol. 276, no. 3, Sept. 1994.

[2] R. Marsh, Ed., "Critical foundations: Protecting America's infrastructure," Rep. Presidential Commission Critical Infrastructure Protection, Washington, D.C., Oct. 1997.

[3] F. Schneider, Ed., *Trust in Cyberspace*. Washington, D.C.: Nat. Acad. Sci., 1998.

[4] K. P. Birman, "The next generation internet: Unsafe at any speed?," *IEEE Comput., Special Issue Critical Infrastructure Protection*, vol. 33, pp. 54–60, Aug. 2000.

[5] O. Rodeh *et al.*, "Dynamic virtual private networks," Dept. Comput. Sci., Cornell Univ., Ithaca, NY, Tech. Rep. TR98-1695, Aug. 1998.

[6] L. Zhang *et al.*, "RSVP: A new resource reservation protocol," *IEEE Networking*, vol. 7, pp. 8–18, Sept. 1993.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[8] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," MIT Lab Comput. Sci. Tech. Rep., Cambridge, MA, 1997.

[9] D. Clark and J. Wroclawski, "An approach to service allocation in the Internet," Internet Eng. Task Force Rep., July 1995.

[10] R. van Renesse *et al.*, "Horus: A flexible group communication system," *Commun. ACM*, vol. 39, no. 4, pp. 76–83, Apr. 1996.

[11] M. Hayden, "The ensemble system," Ph.D. Dissertation, Cornell Univ. Dept. Comput. Sci., Ithaca, NY, Dec. 1998.

[12] X. Liu *et al.*, "Building reliable, high-performance communication systems from components," in *Proc. 17th ACM SOSP*, vol. 33, Charleston, SC, Dec. 1999, pp. 80–92.

[13] K. Birman *et al.*, "The horus and ensemble projects: Accomplishments and limitations," presented at the Proc. DARPA Inform. Survivability Conf. Expo., Hilton Head, SC, Jan. 2000.

[14] N. C. Hutchinson and L. L. Peterson, "The horus and ensemble projects: Accomplishments and limitations," *IEEE Trans. Softw. Eng.*, vol. 17, pp. 64–76, Jan. 1991.

Simulation of Self-Similarity in Network Utilization Patterns as a Precursor to Automated Testing of Intrusion Detection Systems

David A. Nash and Daniel J. Ragsdale

***Abstract*—The behavior of a certain class of automatic intrusion detection systems (IDSs) may be characterized as sensing patterns of network activity which are indicative of hostile intent. An obvious technique to test such a system is to engage the IDSs of interest, and then use human actors to introduce the activities of a would-be intruder. While having the advantage of realism, such an approach is difficult to scale to large numbers of intrusive behaviors. Instead it would be preferable to generate traffic which includes these manifestations of intrusive activity automatically.**

While such traffic would be difficult to produce in a totally general way, there are some aspects of network utilization which may be reproducible without excessive investment of resources. In particular, real network loading often exhibits patterns of self-similarity, which may be seen at various levels of time scaling. These patterns should be replicated in simulated network traffic as closely as is feasible, given the computational ability of the simulator. This motivates interest in an efficient way to detect multiscale phenomena in network traffic, as well as a means to create simulated traffic that exhibits the desired characteristics.

We propose the use of multiresolution wavelet analysis as a technique which may be used to accomplish the desired detection, and subsequent construction of self-similarity in the simulated traffic. Following a multiresolution decomposition of the traffic using an orthogonal filterbank, the resulting wavelet coefficients may be filtered according to their magnitude. Some of the coefficients may be discarded, yielding an efficient representation. We investigate the effect of compression upon the reconstructed signal's self-similarity, as measured by its estimated Hurst parameter.

***Index Terms*—Hurst parameter, self-similarity, simulated network traffic automatic testing.**

I. INTRODUCTION

On a local area network (LAN), computers communicate with one another by exchanging messages in a particular format. Every such message is referred to as a *packet*. The semantics of a particular exchange of packets are determined by various message exchange protocols. The structure of these protocols and the physical attributes of the communications medium over which the packets are transmitted give rise to a set of characteristics which are common to all network traffic which uses the same apparatus, essentially independent of the traffic content.

We wish to address the problem of generating traffic for the purpose of simulating the performance of a local area network. Generally, this equates to the problem of describing a distribution of message arrivals which matches the distributions observed in actual computer networks. For purposes of analytic tractability, an often-used approach is to assume that the arrivals of individual messages are independent of one another, and consequently the use of an exponential distribution to model this phenomenon is used. It has become clear, however, that this assumption may not be warranted in the case of computer network traffic [1]. In fact, it turns out that the very characteristic which the exponential distribution precludes (nonzero autocorrelation of a time series of observations) is one which is desirable to preserve. This is because one

Manuscript received September 1, 2000; revised February 2, 2001.

D. A. Nash is with the Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, NY 10996 USA (e-mail:dd9875@exmail.usma.edu).

D. J. Ragsdale is with the Information Technology and Operations Center, United States Military Academy, West Point, NY 10996 USA.

Publisher Item Identifier S 1083-4427(01)05294-8.