Heterogeneous Networking: A New Survivability Paradigm

Yongguang Zhang
HRL Laboratories, LLC
Malibu, California
ygz@hrl.com

Harrick Vin
Dept. of Computer Sciences
University of Texas at Austin
vin@cs.utexas.edu

Lorenzo Alvisi
Dept. of Computer Sciences
University of Texas at Austin
lorenzo@cs.utexas.edu

Wenke Lee
College of Computing
Georgia Inst. of Technology
wenke@cc.gatech.edu

Son K. Dao
HRL Laboratories, LLC
Malibu, California
skdao@hrl.com

ABSTRACT

We believe that a network, to be survivable, must be heterogeneous. Just like a species that draws on a small gene pool can succumb to a single environmental threat, so a homogeneous network is vulnerable to a malicious attack that exploits a single weakness common to all of its components. In contrast, in a network in which each critical functionality is provided by a diverse set of protocols and implementations, attacks that focus on a weakness of one such protocol or implementation will not be able to bring down the entire network, even though all elements are not be bulletproof and even if some of components are compromised.

Following this *survivability through heterogeneity* philosophy, we propose a new survivability paradigm, called *heterogeneous networking*, for improving a network's defense capabilities. Rather than following the current trend of converging towards single solutions to provide the desired functionality at every element of the network architecture, this methodology calls for systematically increasing the network's heterogeneity without sacrificing its interoperability.

Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection*; C.2 [**Computer Systems Organization**]: Computer-Communication Networks—*General,Network Architecture and Design,Internetworking*

General Terms

Design, Security, Reliability

Keywords

Network security, survivability, heterogeneity, diversity

1. INTRODUCTION

The current trend in networking is towards convergence on a single protocol, software, or technology at each layer of the network's architecture. While this trend towards homogeneity results in improved interoperability and reduced costs, it may pose serious vulnerability to the network as a whole. To draw an analogy from the biological sciences, just like a species that draws on a small gene pool can succumb to a single environmental threat, so a homogeneous network is vulnerable to a malicious attack that exploits a single weakness common to all of its components.

For example, it has been pointed out and again that the continued growth of Microsoft products across a large audience has created an environment where one exploit within a Microsoft product may impact a large number of users worldwide. On the other hand, the reason why the Internet survives the recent several rounds of e-mail attacks (e.g., the love bug) is exactly because of the heterogeneity that we are still having in today's Internet - while the love bug exploits the vulnerability in Outlook, it has no effects on Eudora or Unix e-mail clients. Therefore, it may be intuitive that if more diverse technologies are being deployed in a network and if deployed strategically, the network may be more resilient to orchestrated attacks.

Furthermore, building a network with homogeneous elements run the risk of invalidating some of the assumptions at the very core of using fault-tolerant systems to ensure continuous operations of a network even in the presence of attacks. For instance, techniques developed to tolerate arbitrary (Byzantine) failures have been proposed as a way to make a system survivable to security attacks. The basic idea behind these techniques is to replicate critical components so that, if the number of arbitrarily faulty replicas does not exceed a given threshold t, the system will continue to operate correctly. Clearly, critical to the correctness of all these approaches is the determination of an appropriate value for t. The chosen value should be such that the probability that at any point in time the number of concurrent failures exceeds t is negligible. In classical fault-tolerance literature, this probability is computed assuming that failures are independent: in other words, the failure of a replica does not affect the probability that another replica will also fail. If such fault-tolerance techniques are used to tolerate security attacks in a network with homogeneous elements, the assumption of failure independence is ill founded. In other words, for security attacks it is not reasonable to assume that identical replicas will fail independently: rather, once a successful attack is performed against one replica, the same attack can be performed successfully on all identical replicas. To restore the assumption of failure independence, we need to introduce sufficient heterogeneity back to the network.

In this paper, we propose a new paradigm that achieves network survivability through the use of heterogeneous technologies. We propose a network architecture in which each critical functional capability is provided by a diverse set of instantiations or implementations, so that attacks that focus on a weakness of any one such protocol or implementation is less likely to prevent the network from providing acceptable service.

2. HETEROGENEOUS NETWORKING

The *survivability* of a network is defined as the network's ability to fulfill its mission in the presence of security attacks. Our vision of *survivability through heterogeneity* is based on the observation that different instances of network elements that export the same functional capability are, in general, vulnerable to different security attacks. Hence, a network architecture that supports a collection of heterogeneous network elements is likely to result in higher survivability than a homogeneous network architecture. Consider, for instance, the following examples.

1. A router is an important element of network architecture. A network with homogeneous routers (and hence homogeneous router operating systems) is more susceptible to security attacks than a network architecture that employs a heterogeneous collection of routers with multiple, redundant paths through heterogeneous routers between every source-destination pair.

2. End-to-end network services rely on transport protocols for reliable, timely delivery of data packets; the survivability of such network services depends critically on the ability of transport protocols to survive attacks. Hence, a web service that can utilize UDP or SRDP (Simple Reliable Datagram Protocol) in addition to TCP for data transport can survive a TCP SYN-flood attack (which is the cause of several denial-of-service attacks on web servers today).

3. The current WWW client/server model is often the target of distributed denial of service (DDoS) attacks, in which an adversary, by controlling a large number of unsuspecting clients, issues an overwhelming number of bogus to deny legitimate clients a chance to be served. This attack is especially effective in a network with symmetric bandwidth, such as the Internet core. Since the WWW service model is often asymmetric in bandwidth requirements (with more data flowing from servers to clients than in the other direction), symmetric broadband connectivity may have an unintended negative effect: by increasing the idle bandwidth from the clients to the servers, it makes DDoS attacks more effective.

An asymmetric network infrastructure may help restrain DDoS attacks. For instance, the bandwidth from servers to client in the next generation satellite networks is expected to be around 100Mbps, but the bandwidth from clients to servers is expected to be typically limited to 128Kbps or 512Kbps, making DDoS attacks much less effective. In fact, satellite networks can completely eliminate this type of DDoS attacks by supporting WWW service through a broadcast-based information dissemination model [12] that is not driven by explicit client requests.

An heterogeneous networking paradigm could be used to build a survivable network application on two completely different sets of service models and over two different network infrastructures (see Figure 1). When one service is degraded significantly because of attacks on one or more elements involved, the application can quickly migrate to the second service.

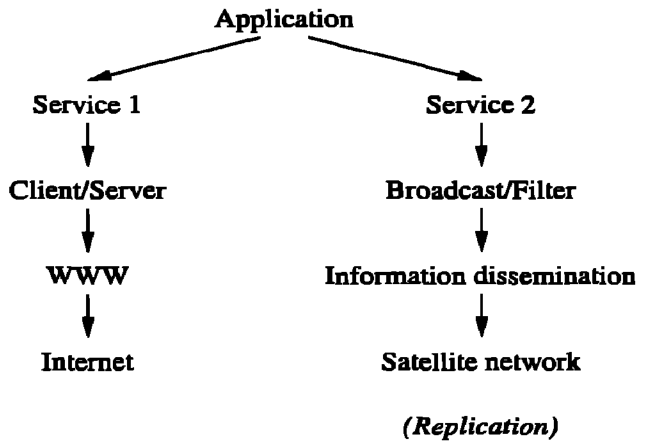

Figure 1: Replications over heterogeneous service models.

As these examples illustrate, the survivability of a heterogeneous networking framework depends critically on the differences in the vulnerability to security attacks of different instantiation of network elements at each level of functional capability. The greater the diversity in the vulnerability of network elements to attacks, the higher the survivability of the heterogeneous networking framework.

2.1 Diversity Space

Conceptually, we can represent the functional capabilities of network architecture and the heterogeneity of network elements using *diversity space diagram.* This diagram organizes functional capabilities of a network (e.g., network and transport protocols, routing protocols, router operating systems, etc.) into a multi-dimensional space. Each network element that instantiates a functional capability is represented as a point along the dimension. Figure 2 illustrates an example of such diversity space. Here, UDP, RTP and TCP are the three network elements along the dimension of transport protocols, while satellite, wireless, and fiber-optic networks are examples of elements for the communication medium (or physical network connectivity).

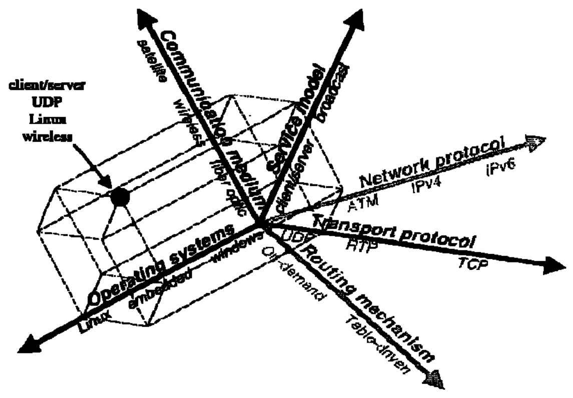

Figure 2: The diversity space for heterogeneous networking.

The *distance* between two network elements along any dimension reflects the diversity in their vulnerability to attacks; the larger the distance between two network elements along a dimension, the smaller is the overlap in their vulnerability to attacks. For example, the distance between "Linux" and "Windows" in the operating system dimension is relatively large because these two systems are independently designed and implemented, while the distance between "IPv4" and "IPv6" is relatively small because the latter is derived from the former.

2.2 Vulnerability Model and Survivability Measure

Given such a diversity space diagram, the key question one has to address in designing a survivable network is: for each of the dimensions, which and how many network elements should a survivable network framework support?

This question can be addressed by developing a *vulnerability model* for each network element, and by introducing the novel concept of "survivability measure" – a metric for capturing the diversity in the vulnerability to attacks of different network elements. In particular, we can identify, for each network element s, the set of attacks A_s that the network element is vulnerable to. Let A denote the cumulative set of such attacks: $A = \bigcup_s A_s$. Then, a survivable network framework should include, at a minimum, the set S of network elements at each level of functional capability such that at least one network element in S is not vulnerable to each of the attacks in A.

Formally, let $a \rightarrow s_\downarrow$ denote that a network element s is vulnerable to attack a. The cumulative known attacks set A can be defined as $\{a | \exists s \in S : a \rightarrow s_\downarrow\}$. Then, we can say that S is survivable to A, if "$\forall a \in A : \exists s \in S : \neg(a \rightarrow s_\downarrow)$" is true. That is, the set S may include network elements such that several network elements are vulnerable to each of the attacks in A, but for each attack there is always network elements that survive it.

We can then develop a quantifiable *survivability measure* for set S; this measure will capture the extent of redundancy required in S so as to reduce the likelihood that every element in S is vulnerable to an unknown future attack. Intuitively, the higher the survivability measure is, the more "diverse" the set is. The more "diverse" a network becomes, the more time/resources an adversary must invest to identify vulnerabilities of all elements and to plan orchestrated attacks on each of them.

Our methodology for constructing the survivable set S is guided by the following conjecture: *survivability of the network elements in set S to the set of known attacks A is a reasonable indicator of the degree to which set S will survive unknown attacks.*

There may be many ways to define a quantifiable survivability measure for a given set of network elements that export the same functional capability. One measure is the cumulative diversity distance between all pairs of elements in the set. Another measure can be the number of distinct attacks that the set can tolerate.

Once we identify the set of network elements S for each level of functional capability, we can design and implement the relevant network elements to create our heterogeneous networking framework. The key challenge is to create a systematic plan for instantiating network elements with reasonable cost and with manageable complexity. A successful instantiation of these network elements will yield a network that will be highly resilient to a vast variety of known an unknown security attacks.

3. HETEROGENEOUS NETWORK COMPOSITION

End-to-end services involve layered implementation of functional capabilities; this can be realized through composition of network elements. In our heterogeneous networking framework, each functional capability is instantiated using a set S of heterogeneous network elements. Hence, in principle, composing together different selections of network elements from each functional capability layer can yield different versions of an end-to-end network service. For example, Figure 3 depicts a composition of several network elements to create WWW and broadcast services. It is easy to see that the broadcast service can also be instantiated by using RLM (reliable layered multicast) instead of UDP as its transport protocol. In such a framework, the network can support half of the services using RLM and the other half using UDP, or it can utilize one of the two instantiations during normal operation and switch to the other instantiation on detecting an attack.

Realizing this in practice imposes several challenges. This is because not all network elements within a network layer may be functionally equivalent from the perspective of an application, even though they play the same role (functional capability) in the network. For instance, both TCP and UDP are transport protocols; however, TCP provides to an application a reliable transport with mechanisms for congestion control, while UDP does neither. Hence, even though TCP and UDP belong to the same network layer, it is, in general, impossible to switch among them in a way that is transparent to the application.

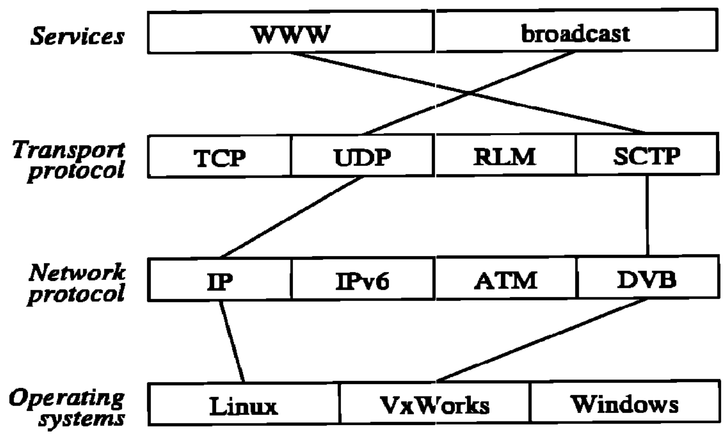

Figure 3: Interchangeable elements at each layer.

This issue can be addressed by the following four mechanisms:

- *Patching lost functionality.* This approach would be to implement any functionality that may be lost while switching from one network element to another at a higher level in the network protocol stack. For instance, on switching from TCP to UDP at the transport protocol level, the functionality of reliable transmission and congestion control can be implemented at the session or higher layer. This approach has the advantage of supporting application transparency, but has the major disadvantage that the resulting implementation may be vulnerable to the same attack of the network element that it is trying to substitute for.

- *Tolerable operation region.* The approach is to renounce transparency, at least partially, and require the application to specify an acceptable region of operation in the heterogeneous diversity space. If an attack merits network service reconfiguration that is outside the application-specified tolerance, then the application is notified through an upcall interface. The application can provide specific handlers to adapt appropriately in response to these upcalls.

  If, on the other hand, the network operates within application-specified tolerance, then any reconfiguration of network service through recomposition of network elements is transparent to the application. To enable such transparent reconfiguration, each network element must export a well-defined interface. Further, the heterogeneous networking framework should export a set of mechanisms to translate and transfer state among network elements providing the same functional capability.

- *Overlay networks.* Using the above two approaches, the heterogeneous networking framework can now support logical overlay networks with multiple physical realizations. Operating such overlay networks also present several design choices. In the simplest case, the framework can use one of the physical realizations as a default, and switch to other realizations only on detecting an attack. In somewhat more complex settings, the framework may simultaneously support multiple physical realizations of the logical overlay network; each physical realization carrying a fraction of the total overlay network traffic. Traffic can be distributed at various levels of granularity: from the packet level to flows to aggregates of flows. These design choices will have implications on the network's ability to support quality of service (QoS) guarantees. This is because, to provide end-to-end service guarantees, a network may need to reserve resources along a path, as well as initialize and maintain state information at each network element. Consequently, switching among different physical realizations on a per-packet basis may violate application's QoS requirements.

- *Multiplexing.* It is quite often that one element in one layer needs to interact with heterogeneous elements of another layer. For example, a WWW server may need to serve clients using TCP or using RLM at the same time. This requires multiplexing techniques to divide one service into multiple forms to be served by heterogeneous alternatives. As another example, a mission critical network can be overlaid on several heterogeneous networks that provide similar connectivity. The overlay mechanism will ensure that it can dynamically change its affiliation with underlying alternatives when one is under attacks.

4. NETWORK RECONSTITUTION THROU(HETEROGENEOUS REPLICATION

Replication has been used in distributed systems as a fault-tolerance measure. When a system component fails, a replicated component takes over the functionality of the failed component so that the system as a whole can accomplish its mission. As we have pointed out earlier, traditional replication measures in a computer network, such as backup routes or backup servers, can improve the network's resilient against unintentional failures, but will not improve its survivability against orchestrated attacks.

Our heterogeneous networking methodology supports a new type of replication – replication of critical network elements – such as connectivity infrastructure, resources, and services – over heterogeneous components. When a successful attack diminishes the functionality of a network element, a heterogeneous replica of the element may still function as usual. Hence, a network can switch to a different, functionally equivalent network element and continue to provide the same end-to-end service to applications. We refer to this approach to survivability as *network reconstitution through heterogeneous replication.*

This network reconstitution approach consists of the following two basic steps:

- *Heterogeneous replication.* This is to replicate the critical network functional capabilities, not by duplicating the components that export these capabilities, but by instantiating them into many different network elements. This can be done by physically duplicating the network components, and having different network elements activated at each components, or by having more than one network elements co-exist at the same physical component. To develop the mechanisms for heterogeneous replication, we will build upon the tools for (off-line) switching and migrating network elements as described in previous sections

- *Dynamic reconfiguration.* This is to reconfigure, on the fly, the composition of network elements. When an attack seriously damages a functional capability provided by a network element, the system can dynamically switch to a replicate of the same functional capability.

Furthermore, dynamic reconfiguration can be used as a preemptive measure. By frequently changing the active set of elements, the network may have taken away the ability for an adversary to identify weaknesses and time needed to plan for an orchestrated attack.

Our network reconstitution techniques are also built upon the following:

- A set of policies that define what critical elements in a network we should replicate, what type of heterogeneous components we should replicate onto, and how to coordinate between replicas during normal operations and during attacks.
- Mechanisms that mediate between intrusion detection algorithms and our heterogeneous networking platform, so that any attack detected by the intrusion detection module can trigger dynamic reconfiguration actions.

One important issue we need to address is to identify as to what we should replicate and what type of heterogeneous replications do we need. We will address this issue through the threat model and the additional intrusion detection component in the next section.

5. THE ROLE OF INTRUSION DETECTION

We will introduce an intrusion detection component in our new survivable network paradigm as an optimization measure. The role of intrusion detection here is to recognize the threats to network services and to provide information about the attacks so that appropriate recovery actions can be carried out. Threat models of network, which specify the essential services and their degrees of tolerable performance degradation or damage, are used by the intrusion detection system (IDS) to determine what to monitor and what constitutes threats. Reports of detected threats by the IDS describing the compromised services and attack techniques are then used to determine which heterogeneous replications should be activated.

In a survivable network where the mission must be fulfilled in a timely manner in the presence of attacks, a threat is an attack scenario that aims to compromise or damage the *essential components/services*. Attacks targeted at nonessential services need not be considered as threats and thus do not warrant network recovery actions, especially when there is limited response time and resources, which is normally the case when the network is under orchestrated attacks.

A *threat model* formally specifies, for a specific mission (i.e., normal usage scenario), which network component/service is critical and which isn't, and for each of these components/services their acceptable quality requirement (or its degree of tolerable performance degradation or damage). The threat models link the policies/requirements with survivability mechanisms because they enable the *recognition* of on-going threats to the network and its mission, and hence facilitate the decision-making on when and how the heterogeneous replicas can be used to recover and reconstitute the mission. As an example, the threat model for a WWW server may include 1) essential service: to provide information of upon request, and 2) minimum quality requirement: to service at least x number of concurrent requests with at most y seconds of delay. This model dictates that if the service is not up to the performance requirement, it is a threat and recovery action must be taken to recover the service.

Because there can be potentially a large number of threats, we can introduce the notion of threat taxonomy where similar threats can be grouped together. The taxonomy can reduce the system complexities because it not only provides a common terminology for referring to the threats but also allows the same recognition and recovery techniques be applied to the same category of threats. For example, we can use the following three dimensions to categorize threats: the *effect* (or goal), e.g., denial-of-service; the *target*, i.e., which essential service is targeted; and the *technique*, i.e., how is the threat carried out. For example, denial-of-service (DoS) can be accomplished by two techniques: "crashing" the server or "resource consumption". Two threats are in the same category if they have the same values in all three dimensions.

In our architecture, the intrusion detection component can list the detected on-going threats and the predicted upcoming threats, based on attack scenario analysis. Using information of the threats, i.e., the effects, targets, and techniques, appropriate recovery actions can be carried out. In particular, the technique dimension determines what type of heterogeneous replication should be used, i.e., *how* to use the heterogeneous replications, for the damaged service(s). For example, if a DoS attack is accomplished via exploiting a bug in Windows and causing the server to crash, then a Linux implementation can be activated. If the DoS attack is accomplished via exploiting TCP handshake (e.g., it is a SYN-flood attack), then other implementations using other transport layer protocol can be activated. To generalize the solution, the threat techniques should be mapped to dimensions of Diversity Space (see Section 2.1) and a heterogeneous replication should be selected automatically so that it has the longest distance from the one that was subject to the identified threat.

6. DISCUSSIONS

6.1 Related Work

The notion of survivability through heterogeneity has been suggested before. For example, in a report published in 1999, CERT proposed to use redundant modules with identical interface but different implementations to recover essential services after an attack [5]. Several DARPA Information Survivability projects, e.g., the Immunix project by OGI [3], also listed heterogeneity (different implementation from the same specification) as one of the main objectives. In their 1997 HotOS paper [6], Forrest et al argued for increasing software diversity as a security measure and suggested diversity techniques such as adding nonfunctional codes, reordering codes, changing memory layout, etc. Heterogeneity has also been exploited to achieve tolerance from software faults through N-Version Programming [1, 2]. N-Version Programming can be further extended to eliminate the effects of certain computer viruses in program handling tools [7]. We believe that our paper is the first to explore the "survivability through heterogeneity" principle in the networking arena.

It is hard to quantify the benefits of heterogeneity vis-á-vis survivability. Most research on these problems has been performed in the context of software fault-tolerance. Eckhardt and Lee [4] developed a probability model to capture the meaning of independence in N-version software development and to explain the failure correlation among versions. Mitra et al [11] also proposed a Design Diversity Metric and used it in calculating and quantitatively analyzing the overall system reliability. In their work, the notion of diversity ($d_{i,j}$) between two implementations with respect to a fault pair (f_i, f_j) is defined as the probability that the two implementations will not produce identical error patterns, in response to a given input sequence, where f_i and f_j affected the first and the second implementations, respectively. Then, for a given fault model, the design diversity metric, D, between two designs is defined as the expected value of the diversity with respect to different fault pairs: $D = \sum_{(f_i,f_j)} P(f_i, f_j)d_{i,j}$, where $P(f_i, f_j)$ is the probability of fault pair (f_i, f_j).

6.2 Is Heterogeneity Achievable?

Over the past decade, researchers have debated the following basic question: *how effective are the techniques for using heterogeneous, independently developed software components for improving software reliability?* Several researchers have advocated software diversity, such as the N-Version Programming approach [2], as a way to achieve higher levels of software reliability. Some industrial sectors (e.g., aerospace) and safety-critical systems have adopted this approach. While relatively little data has been published to date validating the effectiveness of using software diversity for improved reliability, there have also been no reports of catastrophic failure attributable to software faults in these systems [9].

Some other researchers have conjectured that, because of the intrinsic similarity in the common failure modes in software systems, software diversity and hence higher reliability is unattainable simply by using independently developed software components. One of the best known studies is the Knight-Leveson Experiment [8], which involved developing 27 versions of the same software and subjecting them to a million test cases. The experiment revealed overwhelming evidence that failures are more likely to be correlated among versions than to be independent. Although results from this and subsequent experiments are largely negative, the experiment did have positive discovery. It showed that the *average* reliability among all possible 3-version systems is an order of magnitude better than the *average* reliability of the 27 single versions, although no conclusion could be drawn for any *particular* 3-version or single-version system. A good review of the two sides of the software diversity arguments can be found in [9].

Similar issues arise even in our heterogeneous networking paradigm. For instance, it can be argued that if there is a vulnerability in TCP protocol specification, all versions of TCP can be attacked, independent of the platform (e.g., Linux or Windows) that they may be implemented on. Similarly, since many of these implementations use a common source base, the separate implementations may be vulnerable to common attacks. To address these arguments, our approach argues for the use of independent components with identical functional capability (e.g., interchanging TCP with UDP enhanced with a reliability and flow control protocol). We expect that such functionally equivalent components built from completely different building blocks are less likely to have correlated vulnerability. The objective of our study is to investigate this issue carefully. Further, to allow us to assert whether a *particular* N-version system is more reliable than a single-version system, we are investigating the use of models that allow us to quantify the gain in survivability with increase in the amount of heterogeneity.

6.3 Open Issues and Future Work

The open issues that we plan to address in our future work include the following.

- Identifying the proper orthogonal axes for our heterogeneity analysis for today's network architecture.
- Extending the model to include cascade attacks, where attacks are staged to target multiple vulnerabilities. Constant network reconfiguration (such as in [10]) and network reconstitution are the keys to defend against such attacks.
- Identifying common-mode vulnerability. For example, a successful attack on the network reconstitution mechanisms in our approach could bring down the whole network. Furthermore, interoperability often requires standardization and standardization tends to introduce "choke points" from a survivability points of view. Such common-mode vulnerabilities must be identified and properly dealt with.

7. REFERENCES

[1] A. Avizienis. The n-version aproach to fault-tolerant software. *IEEE Transactions on Software Engineering*, SE-11(12):1491–1501, December 1985.

[2] A. Avizienis and L. Chen. On the implementation of n-version programming for software fault-tolerance during program execution. In *Proceedings of International Computer Software and Applications Conference*, pages 149–155, 1977.

[3] C. Cowan and C. Pu. Immunix: Survivability through specialization. In *Proceedings of SEI Information Survivability Workshop*, San Diego, California, USA, February 1997.

[4] D. Eckhardt and L. Lee. A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Transactions on Software Engineering*, SE-11(12):1511–1517, 1985.

[5] R. Ellison, D. Fisher, R. Linger, H. Lipson, T. Longstaff, and N. Mead. Survivability: Protecting your critical systems. *IEEE Internet Computing*, 3(6):55–63, November/December 1999.

[6] S. Forrest, A. Somayaji, and D. Ackley. Building diverse computer systems. In *Proceedings of the Sixth Workshop on Hot Topics in Operating Systems (HotOS-VI)*, pages 67–72, 1997.

[7] M. Joseph and A. Avizienis. A fault tolerance approach to computer viruses. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 52–58, Oakland, California, USA, April 1988.

[8] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1):96–109, January 1986.

[9] B. Littlewood, P. Popov, and L. Strigini. Modeling software design diversity - a review. *ACM Computing Surveys*, 33(2):177–208, June 2001.

[10] J. Millen. Local reconfiguration policies. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 48–56, Oakland, California, USA, May 1999.

[11] S. Mitra, N. Saxena, and E. McCluskey. A design diversity metric and reliability analysis for redundant systems. In *Proceedings of the 1999 International Test Conference*, pages 662–671, Atlantic City, New Jersey, USA, September 1999.

[12] E. Shek, S. Dao, Y. Zhang, D. van Buer, and G. Giuffrida. Intelligent information dissemination services in hybrid satellite-wireless networks. *ACM Mobile Networks and Applications (MONET) Journal*, 5(4):273–284, December 2000.