Cascade Failure in Distributed Networks

Eric Hennigan Donald Bren School of Information and Computer Science
University of California, Irvine
Email: eric.hennigan@ics.uci.edu

***Abstract*—We review, in detail, the previous work on cascade failure, and investigate the possibilities and constraints of creating decentralized solution to the cascade failure problem. We demonstrate a few heuristics for intentional removal strategies that attempt to lighten the load on surviving nodes and maximize the size of the largest connected component after a cascade event. It is shown, via simulation, that the heuristics suggested actually do more harm than good. An intuitive proof of the limitation of algorithms using local-only knowledge is given that explains why developing a decentralized solution might be considered impossible.**

I. Introduction

Physicists have long noticed "cascading events" in physical processes such as phase transitions, sand avalanches, and nuclear decay. Such phenomena has shown its utility and has been explicitly used in the design of sensitive detectors, including photomultiplier tubes and Geiger counters. Systems engineers, by contrast, paid relatively little attention to cascading phenomena until civil engineering projects became both large enough that interdependence among system parts and functionality was unavoidable, and automated enough that the natural time scale of the system was so quick it prevented human intervention. A particularly notorious example of cascade failure was the Great Northeast Blackout of 1965, which served as a wake-up call for most of society, and civil engineers in particular. Since that time, other systems have shown similar behavior, including the Internet, which has repeatedly suffered congestion collapse since 1986 [1]. Fortunately, such outages have been primarily the result of accident or poor maintenance, and not of intentionally malicious activity. Nevertheless, we should consider ourselves as having been alerted to both our complete social dependence on such strangely fragile systems, and the security implications thereof.

The phenomenon of cascade failure has now been recognized in a wide variety of systems, including biological ecosystems [2], biochemical pathways [3], and financial markets [4]. Indeed, any system that can be modeled using an interdependence graph with limited capacity of either nodes or edges to carry flow of some resource such as information, electrons, or energy, will be exhibit the cascade failure phenomenon. Despite this generality, the problem has not been particularly well studied, even though the networks which can suffer such failure are an essential part of modern society [5]. Because these systems (power, communications, transportation, financial) which support our modern lifestyle are constructed under economic considerations, engineers naturally find great utility in preferential linking, as it avoids the high cost of adding edges to an existing network. This promotes the creation of inhomogeneous scale-free networks [6]. Though most of these systems are explicitly designed for error tolerance and are well known for reliability when subjected to random outages, they are also notorious for experiencing widespread outage or congestion failure as a result of the removal of only a few critical nodes. These are typically the nodes which play an important role in ensuring network connectivity [5] [7].

II. Summaries of Previous Work

Usually, the removal or failure of nodes or edges, either by random breakdown or intentional attack, within a stressed distributed system, will trigger a subsequent redistribution of stress within the system [5]. If the redistribution results in further outages by overloading the remaining nodes or edges, then a chain of such outages called a *cascade failure* is said to occur. It is important to note that in real-world systems, many breakdown failures in components are not statistically independent, but fall under the systems engineering terms *common cause failure* or *common mode failure*. For example, packet routers on computer networks are only manufactured and programmed by only a few companies, which share both architecture and algorithms. This can lead to potential systemic faults [8], which would share a root cause, a feature not present in fully heterogeneous systems. The term *cascade failure* does not apply in this case, because it is reserved for models covering independent failure. The term exclusively covers those faults which are spread through the system interdependency, rather than widespread faults which share a single cause. The differentiation of cascade failure from other dependent failures was first tabulated (Table. I) by the U.K. Atomic Energy Commission [9].

Because our society, and the systems that support it, are geographically distributed in a non-random fashion, outage of heavily loaded nodes is proportionally higher, indicating that an attack model with the desire to cause as much disruption as possible, would selectively target the loaded nodes of a network. Unfortunately, the consequences of such an outage will be more severe if it is able to spread, via cascade, over the entire network [5].

A. The Watts Model [7]

Watts pointed out that cascading events are extremely difficult to predict, even when the properties of the individual components are well understood. The same systems that routinely display great stability in the presence of continual small failures and shocks by incorporating fault-tolerance and redundancy into their design, also exhibit cascading failure as a direct result of their load-balancing and recovery mechanisms.

TABLE I
DIFFERENTIATION OF CASCADE FAILURE FROM OTHER DEPENDENT FAILURES.

Dependent Failure	The likelihood of a set of events, the probability of which cannot be expressed as a simple product of the unconditional failure probabilities of the individual events.
Common Cause Failure	The type of failure which occurs in redundant components, in which a single common cause simultaneously (or near simultaneously) leads to failures in different types of system components.
Common Mode Failure	The type of failure which occurs when multiple components fail in the same manner.
Cascade Failure	Dependant failures which do not share a common cause, meaning they do not affect redundant components.
Notably: The definition of "dependent failures" includes all definitions of failures that are not independent. This definition implies that an independent failure in a group of events is expressible as a simple product of the conditional probabilities of failures of a single event.	

Cascades can therefore be regarded as a specific manifestation of the *robust yet fragile* nature[1] of many complex systems [11]: a system may appear stable for long periods of time and withstand many external shocks (robust), then suddenly and inexplicably exhibit a large cascade (fragile). From this, Watts also identifies a couple of very important qualitative observations about global cascades:

- They can be triggered by exogenous events (shocks) that are very small relative to the system size.
- They occur rarely relative to the total number of shocks that the system receives, and may be triggered by shocks that are *a priori* indistinguishable from shocks that do not.

Watts' model is motivated by noticing that in all social phenomena exhibiting cascade behavior, it arises as a direct result of individual decision makers having an incentive to pay attention to the decisions of others. Networks where an individuals' state can be given as an explicit function over neighboring states, immediately suggests the use of a model for a class of problems known generically as *binary decisions with externalities* [12].

He uses a model where each node maintains a binary state, either 0 or 1, and a function for switching state that is based entirely on the state of its immediate neighbors. The dynamics of cascade are modeled by initializing the network in an all-off state, and then perturbing it by turning on a very small fraction of nodes. The network then evolves in an asynchronous fashion according to a threshold function, that causes a node to turn on if a sufficient fraction of its neighbors are on.

He compares this model with several others (percolation models of disease-spreading, random-field Ising models, bootstrap percolation, majority voting, self-organized criticality, etc..) and notes the introduction of certain features (not all present in the other models), that are essential to the dynamics of cascades — *local dependencies, fractional thresholds*, and *heterogeneity*. He also observes that network heterogeneity and threshold function heterogeneity are not equivalent and considers them separately.

His model uses random graphs as a first approximation, despite the fact that they are not representative of real-world networks. Like the other researchers discussed later, he focuses on two quantities: (i) the probability that a *global cascade* will be triggered by a single node (or small seed of nodes); and (ii) the expected size of a global cascade once it is triggered. In analyzing his model, Watts identifies as *vulnerable* those nodes which are on the verge of switching state because of the edges they share with a dynamically expanding initial seed. This situation has parallels to the social dynamics of early and late adopters, in that the successful adoption of an innovation is dependent on both the number of early adopters and how they are connected to each other.

He conjectures that the required condition for a global cascade, in his model, is that the subnetwork of vulnerable nodes must percolate throughout the network as a whole. This condition is formalized, and marks the transition between two *phases*: (1) that vulnerable clusters in the network are small and isolated such that they are unable to generate the momentum required for a cascade; and (2) that the typical size of vulnerable clusters is effectively infinite, implying that a random initial trigger will hit a vulnerable cluster capable of percolating throughout the network as a whole. The insight of identifying the vulnerability of an entire network component as the probability of hitting *any* node within a percolating cluster is aptly demonstrated in simulations of random networks, and supports the observation that though cascade failures occur rarely, they are very large when they do.

B. The Motter-Lai Model [5]

Motter and Lai were the first to address the issue of cascade failure in distributed networks. The network model that they introduced in their Rapid Communication, *Cascade-based attacks on complex networks*, is generally applicable to realistic networks such as the Internet and power grids, yet simple enough to support tractable analysis. It consists of several key elements:

- The **traffic** is simulated by the exchange of one unit of the relevant quantity (information, energy, etc.) between every pair of nodes along the shortest-hop path connecting them. The load placed on a node is then equivalent to total number of shortest-hop paths passing through the node.
- The **capacity** of a node is defined as the maximum load that the node can handle. Because real-world networks are severely limited by construction costs, the capacity C_j of node j is assumed to be linearly proportional to its initial load L_j,

$$C_j = (1+\alpha)L_j, \quad j = 1, 2, ...|\mathcal{N}|, \tag{1}$$

  where the constant $\alpha \geq 0$ is the *tolerance* parameter, and $|\mathcal{N}|$ is the initial number of nodes.
- **Cascade Failure** of the network is a result of re-balancing efforts of the network when a node fails. When all nodes are operational, the network operates steadily as long

[1]This *robust yet fragile* property was first observed of heterogeneous networks in Ref. [10]

as $\alpha \geq 0$. But the removal of a node will cause a redistribution of the shortest paths. This will generally increase the load at some of nodes. If the load increase exceeds the capacity of any node it will fail, triggering a new redistribution, and possible subsequent failures. Eventually, the cascade will stabilize, when all remaining nodes can handle their load. The resulting network is typically partitioned into a number of connected components.

- The **network** is a scale-free network, which follows a power-law distribution in node degree. The probability P that a randomly chosen node has degree κ is given by

$$P(k) \sim \kappa^{-\gamma}, \tag{2}$$

where γ is the scaling exponent.

- The **damage** caused by a cascade failure is quantified by the relative size G of the largest connected component,

$$G = |\mathcal{N}'|/|\mathcal{N}|, \tag{3}$$

where $|\mathcal{N}|$ and $|\mathcal{N}'|$ are, respectively, the number of nodes in the largest component before and after the cascade.

Using this model, Motter and Lai observe that G remains close to unity in the case of random breakdowns, but is significantly reduced under attacks targeting nodes of highest load. For example: even when every node has capacity twice that of initial load ($\alpha = 1$), the size of the largest component is reduced by more than 20%. For smaller values of α the damage is considerably greater.

Experimental simulations were also carried out on homogeneous networks (those with a more uniform distribution of node degree), does not experience cascading failure due to targeted removal or random breakdown for α as small as 0.05. From this they conclude that homogeneous networks are more robust against attacks.

C. Defense by Intentional Removal of Nodes [13]

Building upon the Motter-Lai model and previous results, Motter continued work on an investigation of the possible strategies of defense to prevent the cascade from propagating through the entire network. He begins by noticing that a cascade can be divided into two events:

I The **initial attack** where a small fraction of nodes is removed either by intention or random breakdown.

II The **propagation** of the cascade, where another fraction of nodes is removed by subsequent overload failures.

The method of defense rests in the strategic removal of nodes after (I) but before (II). This is a valid assumption, because the time scale involved during a cascade failure is typically much shorter than the time scale at which the network grows. Intentionally removing nodes via a shut-down procedure is feasible, but adding nodes or edges during a cascade event is not. Though the removal of nodes can result in an even smaller final connected component, it is possible to reduce the magnitude of the cascade by *choosing carefully* which nodes to remove.

The model is slightly improved by giving a new definition of inter-node communication, and adding an explicit reference to the evolution of the network in time.

- The **traffic** is simulated by sending one unit of flow from node i to node j, for every ordered pair of nodes (i, j) belonging to the same connected component. If there is more than one shortest path connecting nodes i and j then the traffic is divided evenly at each branching point.
- The **load** on a node k is then given by

$$L_k = \sum_{i,j} L_k^{(i,j)}, \tag{4}$$

where $L_k^{(i,j)}$ is the contribution of the ordered pair (i, j) to the load on node k.

- The **cascade simulation** is a step-wise process, starting with the network $\mathcal{N} = \mathcal{N}(0)$ at time 0, with no nodes overloaded. The initial attack is performed at time 1, by removing a fraction p of nodes from $\mathcal{N}(0)$ to form the network $\mathcal{N}(1)$. The redistribution of flow is calculated and all loads $L_k(1)$ are updated. Any nodes where load exceeds capacity, are considered to be overloaded and are simultaneously removed to form the resulting network $\mathcal{N}(2)$. Redistribution proceeds in this manner, until the cascade stops at time t' by satisfying the relation

$$L_k(t') \leq C_k \quad \forall k \in \mathcal{N}(t'). \tag{5}$$

- The **generated load** of a node i is given by

$$L_i^g = \sum_j (D_{ij} + 1) = (\bar{D}_i + 1)(|\mathcal{N}_i| - 1)\,, \tag{6}$$

where $\bar{D}_i$ is the average shortest path length from node i to all others within the connected component $\mathcal{N}_i$ containing node i.

By observing that (1) nodes whose load L_i is much larger than L_i^g contribute much more to handling than to generating load, and are therefore the most important nodes for maintaining connectivity, and that (2) the removal of any node i will increase the load on the remaining nodes by at least $L_i - 2L_i^g$, (unless it partitions the network into more than one connected component), Motter identifies a rationale for attack strategies based on the removal of highly loaded nodes. By contrast, nodes whose load L_i is smaller than L_i^g generate more load than they handle, and removing them can dramatically reduced the size of the cascade. This results in the identification of four strategies for a defense based on intentional removal (IR).

1. Nodes with smallest $\Delta_i \equiv L_i - L_i^g$ are removed first.
2. Nodes with smallest *closeness centrality* $\bar{D}_i^{-1}$ are removed first.
3. Nodes with smallest load L_i are removed first.
4. Nodes with smallest degree κ_i are removed first.

For the intentional removal defense to be effective, removals must satisfy two conditions:

- The load on the remaining nodes must be reduced.
- The fragmentation caused by the IRs must be smaller than that caused by the cascade itself.

Motter only considered strategy (1) for analysis, and demonstrated that it satisfies both of the conditions for an effective defense. The other strategies performed similarly under simulation due to the very strong correlation (in scale-free networks) of the parameters on which the strategies are based.

From this he concludes, that while the removal of the most central nodes of $\mathcal{N}(0)$ can trigger a cascade, the removal of the least central nodes of $\mathcal{N}(1)$ can drastically reduce the size of the cascade. A numerical simulation using all four strategies showed approximately equal effectiveness at stopping a cascade, and demonstrates that IRs form marked improvement over no defense at all.

D. A Proactive Defense [14]

It was noticed that the IR defense is somewhat less than ideal, both because it purposefully removes nodes and because it focuses on the minimization of damage rather than prevention. Schäfer, Scholz, and Greiner proposed a proactive measure to significantly reduce the chance of an overload avalanche and to limit its size in the case of occurrence. Their approach is based on load weights, and focuses on those flow paths with the smallest load-based length. Routing traffic in this manner turns a heterogeneous load distribution into a more homogeneous one, reduces the need to shut off nodes to stop a cascade, and simultaneously lowers the investment costs in network capacity layout.

Continuing the use of the Motter-Lai model, they notice that the load on a node k is a result of summing all shortest-hop paths traversing through that node. A contingency analysis is done with the hypothetical removal of a single failed node. For each node in the network, a redistributed load is calculated as if that node had failed. The resulting networks provide information about the minimum capacity that each node must be able to carry for the network as a whole to survive a one-node failure without triggering cascade. The capacity that a node i must have in order that the network survive any one-node failure is found by taking the maximum over all networks with single-node removal,

$$C_i = \max_{j \in \mathcal{N}_i} L_i(\mathcal{N}_i \backslash j) \tag{7}$$

where $L_i(\mathcal{N}_i \backslash j)$ is the load on node i in the connected component containing node i but with node j removed.

Load-based shortest paths, which tend to avoid the most-loaded nodes, are also considered, but the calculation proves to be much trickier, due to the introduction of a recursive relation between loads and paths. Using an iterative technique that assigns a load-based weight to each node, a stable distribution of load-based shortest paths can be discovered for single-node removals from the initial network. The resulting node capacities for preventing cascade are still given by Eqn. 7. It is noticed that those nodes which had a high hop-based load experience a significant decrease in load upon application of load-dependent weights, while nodes which carried a small load acquire a larger one. The resulting load-dependent routing is found to turn a heterogeneously loaded network into a more homogeneous one. The same technique can be used to discover the required capacity investments for two or more node removals. The required infrastructural investment for hop-based distribution is greater than that for the load-based one.

E. Assessment

The Motter-Lai model of network traffic is somewhat unrealistic in that it specifies every node to communicate one unit of flow with every other node. Though this has the advantage of being a highly regular and predictable traffic pattern for any network topology, such traffic patterns are not seen in real-world networks. In addition, the resulting traffic is computationally intensive to calculate, and thus prohibits very large networks from being simulated.

Measuring network damage by focusing on the size of the largest connected component is reasonable, as it promotes maintenance of global network cohesion. Alternatives, such as using the average size of connected components, would instead focus on local neighborhood connectivity, allowing a greater amount of network partitioning than is desirable for something akin to the Internet, which derives its utility from maximizing global connectivity.

The focus on scale-free networks is also justifiable, because such a topology appears in a wide variety of empirically observed networks, as a result of (a) the benefits arrived through "preferential attachment," [15] and (b) the economic considerations concerning node and edge additions as the network grows. A further benefit of studying such networks through simulation is their remarkable degree of self-similarity. Even though the Motter-Lai model of traffic hinders the simulation of very large networks, we are nevertheless permitted to maintain a high level of confidence in generalizing the results from smaller networks to larger ones.

The initial approach at a defense to cascading failure by the strategic and intentional removal of nodes in order to remove excessive load from causing further node failures, has the advantage of a low to non-existant incremental investment cost, as it only requires the ability to perform a remote shutdown of nodes. It has a strong disadvantage, however, in that it is both difficult to provide early detection of cascade failures and it requires knowledge and monitoring of the global topology, which is unavailable in many real-world networks.

The difficulty of early detection makes the intentional removal of nodes after an initial attack or random breakdown but prior to the cascade an unrealistic damage control strategy for many real-world networks. Some networks, though, have rebalancing dynamics slow enough to permit such intervention. For example, the power grid operates under predictable and cyclic loads and is equipped with sensors and monitoring stations which collect enough information, that overload of the distribution network is preventable through the removal of nodes in strategies such as rolling blackouts. For other networks this is not possible, either because of the cost and coordination overhead in monitoring or because the cascade propagates as quickly as the network communication itself, providing no window for intervention.

Motivated by these difficulties, we seek a proactive rather than reactive solution, to prevent cascade rather than forestall the resulting damage. The solution proposed by Schäfer, Scholz, and Greiner [14] attempts such by simulating the removal of nodes in order to calculate the resulting capacity constraint that each node must meet in order for the net-

work to survive. We consider this approach as completely impractical for most real-world networks, because it requires global knowledge of the topology and a time span between network changes long enough to run such simulations. Though it can yield valuable security information useful for resource allocation during network construction, it is not appropriate for mitigating failures when they occur in a dynamic environment.

III. Impossibility of Cascade Detection

A. The Global Problem

We notice that previous work in understanding the cascade failure phenomenon has not resulted in the creation or suggestion of any load-balancing algorithms with the goal of minimizing the cascade, but which operate in a distributed manner using only nearest-neighbor communication. We intuitively believe that a distributed algorithm for preventing cascade may not be possible. By examining what happens in the neighborhood of a node during a cascade, we can identify the following events, in a best-case scenario:

1. Self and neighbor nodes suffer a load increase.
2. One or more neighboring nodes overload and fail.
3. Self and neighbor nodes suffer another load increase, as a result of the previous failure(s).

In this timeline, a load increase immediately precedes node failures in the neighborhood. In real-world networks, this event can happen simply as a result of fluctuating network demand or traffic patterns, and is not, by itself, an indicator of cascade. The failure of neighboring nodes, by itself, is also not an adequate predictor of cascade failure. However, the combination of these events, in a recorded history, can be used as an indicator that a cascade is in progress. In fact, the more sharply the load increases during phase (1) the more confident we should be in such a prediction. It is important to note that both load and capacity must be known by all neighbors in order to make such a prediction, otherwise a node has no way of distinguishing neighboring nodes which failed due to overload from those that failed due to random breakdown. It is important to observed that because load demands on real-world networks fluctuate dynamically, these events above will have the same appearance, independent of the observed node's location with respect to the node that triggered the cascade, making the analysis applicable to all nodes in the network.

If nodes are restricted to nearest-neighbor knowledge, then the assessment that a cascade is in progress can never be made with absolute certainty. This is especially true if the wavefront of the cascade advances through the network on the same time-scale as the inter-node communication. However, by monitoring and communicating the rate of load change, it is possible to improve the assessment. This result corresponds with the observation that cascade failure is a global phenomenon, and knowledge of the local conditions will never reveal quite enough information. However, even though a distributed solution for completely preventing cascade failure with certainty is impossible, it is still worth investigating distributed load balancing strategies with heuristics that might assist in limiting network damage.

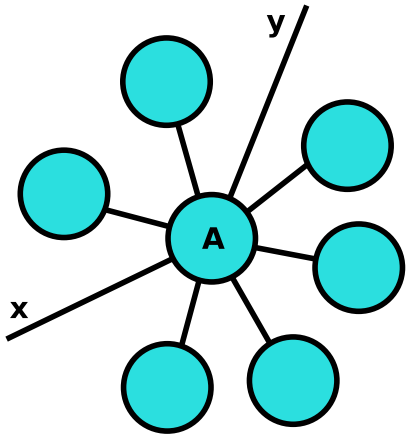

Fig. 1. *A subgraph of a typical node in a scale free network.*

B. The Local Problem

Consider a local subgraph, as in Fig. 1, of a typical node in a scale free network. It is clear that the unlabeled nodes surrounding node A generate more traffic than they route, satisfying a criterion for intentional removal given by Motter [13]. Givnen Motter's success with defensive mechanisms that will selectively remove these leaf nodes from the network, we ask: How are these nodes to receive the removal command in a decentralized network?

In Motter's simulations, node A will have been assigned a capacity capable of handling all the traffic generated via itself and its leaf nodes. This means that the only way in which A becomes overloaded is when the traffic via the two edges x and y, were to suddenly increase as a result of re-routing brought on by node failures at least 2 hops away from A. Under our restrictions, this means that none of the nodes depicted in the cluster will receive warning in time to selectively remove themselves. Unfortunately, because scale-free networks display a large amount of self-similarity, most of the local topologies will fall in this category, implying that a decentralized solution will be very difficult to obtain, if it is possible at all.

Even assuming that it is possible to detect and then give advanced communication of a cascade failure, the only strategy that succeeds in maximizing the number of connected nodes within the largest remaining component, would be for this cluster to participate in a outage-scheduling mechanism. That is, nodes can participate in sharing the time spent in voluntary removal. In the power grid, this strategy is already practiced as a rolling-blackout. Our model requires that all the nodes perform the same local computation, which results in the prompt voluntary shutdown of *all* leaf-nodes connected to A, if not A itself.

Any algorithm that attempts to solve this problem will have to identify A as more important than any of the leaves, because of its potential, using edges x and y, for joining two parts of the global network. But it is precisely this potential that makes node A vulnerable to automated re-routing of traffic. If the leaf nodes are left online, then the capacity of A to form a such a bridge is severely reduced, and it will likely

fail due to overload. Yet we still observe that, subject to the nearest-neighbor restriction, it is not possible to preserve A by voluntarily removing the leaf nodes. For, if the cascade propagates as fast as inter-node communication, there will simply not be enough time to communicate the command.

IV. Network Centrality Heuristics

We formally define our network $\mathcal{N}$, as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of verticies $\mathcal{V}$ and edges $\mathcal{E}$. Every node $n \in \mathcal{V}$ will represent a communication device (hub, router, or substation) on the network, while every edge $e = (v, w) \in \mathcal{E}$ represents a direct line of communication between devices. A node w is a **neighbor** of v if they share an edge. Every node v is also equipped with memory for storing information regarding load L_v, capacity C_v and a list of neighbors $\mathcal{V}(v)$. For convenience, the **degree** $\kappa(v)$ of node v is defined as the number of its neighbors, $|\mathcal{V}(v)|$.

A **path** $p(s,t)$ from a *source node* s to a *target node* t is an ordered k-tuple of nodes $(v_1, v_2, ..., v_k)$ such that $(s, v_1) \in \mathcal{E}$, $(v_k, t) \in \mathcal{E}$ and $(v_i, v_{i+1}) \in \mathcal{E}$ for $i = 1..k$. The *hop length* $l(p(s,t))$ of a path p is given by the number of edges on the path, thus $l(p(s,t)) = k + 1$. The **hop distance** $d(s,t)$ from node s to node t is the minimum hop length of any path $p(s,t)$, with $d(s,s) := 0$ and $d(s,t) := \infty$ if s and t belong to separate components of $\mathcal{G}$. Because there may be more than one such **shortest path**, the **number of shortest paths** is denoted by σ_{st}, with $\sigma_{ss} := 1$. The **number of shortest paths passing over node** v is denoted $\sigma_{st}(v)$.

Social network theorists have identified some common and useful metrics, based on shortest paths, for identifying the importance of a node v within a graph:

Closeness Centrality [16]

$$C_C(v) = \frac{1}{\sum_{t \in \mathcal{V}} d(t,v)} \tag{8}$$

Graph Centrality [17]

$$C_G(v) = \frac{1}{\max_{t \in \mathcal{V}} d(t,v)} \tag{9}$$

Stress Centrality [18]

$$C_S(v) = \sum_{s \in \mathcal{V}} \sum_{\substack{t \in \mathcal{V}, \\ s \neq t}} \sigma_{st}(v) \tag{10}$$

Betweenness Centrality [19]

$$C_B(v) = \sum_{s \in \mathcal{V}} \sum_{\substack{t \in \mathcal{V}, \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{11}$$

Most pertinent to our discussion is the betweenness centrality, which measures the expected routing service demands of node v if every node was communicating with every other simultaneously [20]. Using the algorithm developed by Brandes [21] allows us a very fast implementation for each of the chosen centrality metrics.

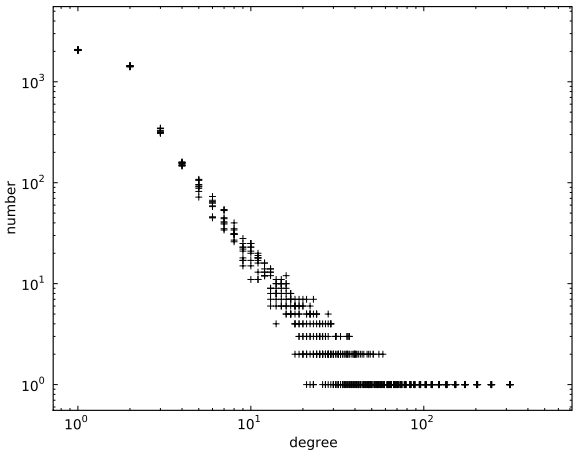

Fig. 2. *The degree distribution of nodes.*

V. Methodology

We use Inet-3.0 to generate scale-free random test networks. Despite the fact that this tool does not well represent the Internet in terms of maximum clique size and clustering coefficient, it still constructs realistic topologies with regard to resilience, pair size within h hops, outdegree vs rank power law, characteristic path length, average eccentricity and distortion [22]. As an alternative means of generating network topologies we considered Aldridge's algorithm for generating scale-free networks with preferential linking [23], but decided in favor of Inet-3.0 because it has been explicitly engineered to generate topologies resembling that of the Internet, and therefore better represents networks assembled under real-world economic constraints, providing networks qualitatively more suitable for our analysis.

However, Inet-3.0 does have a tendency to create a few nodes with unrealistic connectivity. So after generating networks of 5000 nodes, we removed the 3 nodes of highest degree. By default 30% of nodes created by Inet-3.0 have degree 1, so this removal process isolates many nodes, the network is scanned and degree 0 nodes are removed before simulation. The resulting networks have the degree distribution as shown in Fig. 2, and range in size between 4378 and 4412 nodes.

Since we are primarily focused on load distribution and intentional removal of nodes than in the routing of network traffic, we initialize the network with a worst-case scenario of load by having each node communicate in shortest-hop fashion with every other node, splitting the traffic evenly if multiple paths are available. This can be easily calculated using the betweenness centrality algorithm by Brandes [24]. We consider this a reasonable distribution of load in the sense that nodes which act as "hubs" will be initialized with more load than "leaf" nodes at the network periphery. The load will then be considered as if it were a resource demand on that node. The resource allocation could be a result of data allocated for a distributed hash table, or processes allocated in a cloud. We investigate a few local-decision based strategies

TABLE II
NUMBER OF NETWORKS THAT UNDERGO CASCADE CYCLES BEFORE STABILIZATION.

Capacity	**Number of Cascade Cycles**			
α	2	3	4	5
0.0	10			
0.1		3	4	3
0.2		7	2	1
0.3		10		
0.4	4	6		
0.5	5	5		
0.6	3	7		
0.7	5	5		
0.8	4	4	2	
0.9	6	4		
1.0	7	3		
1.1	7	3		
1.2	8	2		

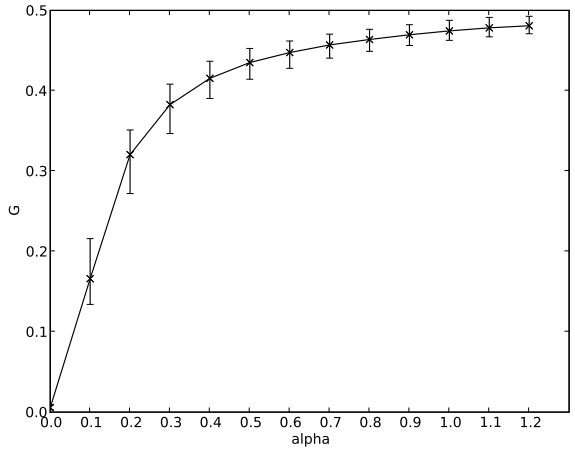

Fig. 3. *The size of the largest connected component relative to the initial network size vs. the network capacity, α, for networks without a defensive mechanism. Error bars represent the min and max size of the largest component across the 10 test networks.*

for intentional removal in the event of a failure cascade.

After creating the network, it is initialized with loads based on n^2-communication traffic. Each node is then assigned a capacity dependent on α as given by Eqn. 1. The node with the highest load is considered to have been subject to a targeted attack, and is removed from the network. This triggers a cascade, which is simulated in stages by re-routing the n^2-communications, processing intentional removals, and failing nodes that become overloaded. This proceeds until the network stabilizes, as shown in Fig. 3. A profile of the number of networks and cascade cycles that proceed before stabilization is given in Table II.

VI. RESULTS

The results of triggering a cascade without any mechanism for defense qualitatively match those of other researchers. We see that by targeting the heaviest loaded node, a cascade can be triggered that spreads throughout the network. The cascade usually proceeds with only two or three cycles before stabilizing, indicating the correctness of our assumption that the cascades progress at a rate close to that of inter-node communication.

We tested a few very simple decentralized heuristics for the intentional removal defense that are based on localized knowledge at some node n.

Defense 1. Shutdown if $L_n < \frac{\sum_{i\in\mathcal{V}(n)} L_i}{\kappa(i)}$.
A node will remove itself if it carries a load less than the average load of its neighbors.

Defense 2. Shutdown if any neighbors have failed and $L_n < \frac{\sum_{i\in\mathcal{V}(n)} L_i}{\kappa(i)}$.
A node will remove itself if it carries a load less than the average load if its neighbors and at least one of the neighbors has already failed. Our model does not distinguish the difference between voluntary removal and overload, so the term *failure* applies to any node that has transitioned to an inactive state.

Defense 3. Shutdown if more than 1/2 of neighbors have failed and $L_n < \frac{\sum_{i\in\mathcal{V}(n)} L_i}{\kappa(i)}$.
A node will remove itself only when most of its neighbors have also failed, and the condition of Strategy 1. applies.

Defense 4. Shutdown if $\frac{C_n - L_n}{\mathcal{V}(n)} < \sum_{i\in\mathcal{V}(n)} \frac{C_i - L_i}{|\mathcal{V}(i)|}$
A node will remove itself if the remaining capacity averaged over its neighbors is less than their remaining capacity averaged over their neighbors. We assume that each node will broadcast this value to neigbors at each network time-step, although the traffic that would be generated in this manner is ignored, and is not considered part of the model.

All of the defensive strategies performed much worse than no defense at all, though they did reduce the number of network cycles before stabilization. Strategies 2 and 3 performed the same, suggesting that when any neighbors fail, a majority of them do so, and both fared better than strategies 1 and 4, indicating that shutting down only when neighbors have gone offline will probably form a necessary part of any successful strategy for maximizing the size of the largest connected component. This observation is in correspondence with our earlier time-line analysis on a typical local neighborhood. Curiously, strategy 1, which was based on average load, outperforms strategy 4, which was based on an averaged remaining capacity.

Because all strategies performed performed so poorly, we find that they are overzealous in the practice of voluntary removal. In particular, strategy 1. will fail all nodes with load less than average, even if no cascade is occuring. This strategy, will in fact cause a cascade of voluntary removals. As the lighter loaded nodes drop out of the network, the average load is also brought down, which may cause further voluntary

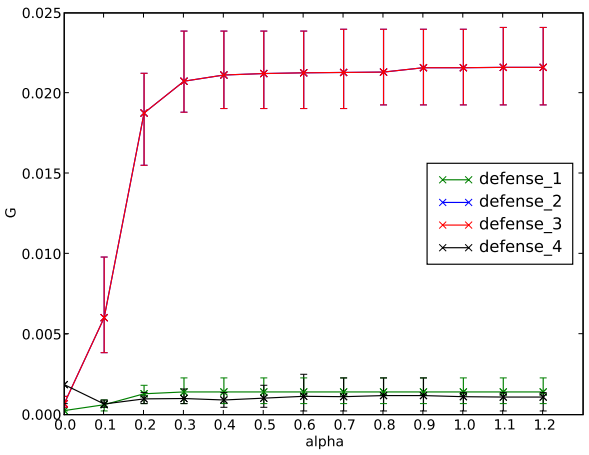

Fig. 4. *The size of the largest connected component relative to the initial network size vs. the network capacity, α, for networks with defensive mechanisms.*

removals. This demonstrates a poignant danger with regard to our goal of developing a decentralized intentional removal strategy: the strategy itself might induce a cascade. We, don't as yet, know of a self-correcting strategy that would prevent such an occurance. But we suspect that it would have to be based on self-stabilizing techniques discussed by Dijkstra [25]. It may well be that congestion aware routing strategies and rate-based load balancing are the most viable options, as they both encourage more homogeneous load patterns, which is already known to partially mitigate the effects of heterogenous linkage resposible for cascading failure.

VII. Future Work

For the purposes of easing implementation, our model made a few assumptions that should undergo reconsideration:

Omniscient Routing. By using the betweenness centrality measure, we assumed that each node was capable of omnisciently routing traffic along a shortest-hop path. A more realistic model would use discrete event simulation to more accurately depict actual routing algorithms. For the purposes of investigating the effects of routing on the cascade phenomenon, it would be useful to simulate routing or load-balancing algorithms that are congestion-aware [26], gradient-based [27], or rate-based.

n^2-based Load. Simulating traffic in this manner, though simple to calculate, should be considered as overly pessimistic. More realistic load patterns should be simulated. To the best of our knowledge, proposed traffic generators such as Swing [28] and Harpoon [29] are focused on the accurate simulation of TCP, UDP and protocol-specific simulations. As such they are not really appropriate for the study of cascade failure, and they possess too steep a cost for simulations such as those considered here. Better statistical models describing load distribution in multi-cpu systems would also allow us to forgo the assumption that traffic is equivalent to load.

Local Knowledge. Our model severely resticts the decisions at each node to be functions of nearest-neighbor properties only. Since most real-world networks actually have long up-times between global outages, it is entirely possible for nodes to obtain an idea of global parameters via distributed computations. Each of the centrality statistics mentioned in this paper are now possible to calculate via nearest-neighbor communication [30], which enables each node in the network to have a rough idea of its various topological properties. The nature of these algorithms naturally permit a node to estimate the topology of its neighbors without any communication overhead apart that of the computation protocol itself. Any future work in this area should incorporate such techniques into both routing protocols with the goal of homogenizing traffic and distributing load, as well as heuristics to detect cascades in progress, or predict that one is in danger of happening. In particular, a distributed computation of the single node removal contingency analysis mentioned in the proactive defense approach [14] would allow nodes critical to the network infrastructure to warn their operators of the situation, allowing better allocation of network infrastructure.

We would like to mention that our focus was on maximizing the size of the largest connected component rather than on minimizing the number of components into which a network is fractured. Future development could focus on maximizing a heuristic such as

$$\beta|\mathcal{N}| + (1-\beta)|\mathcal{N}_{max}|, \tag{12}$$

where $|\mathcal{N}|$ is the number of components and $|\mathcal{N}_{max}|$ is the size of the largest component, and β is a tuning paramater specifying the relative importance betwen number of components and the size of the largest component.

We also did not have time to investigate wether a calculation utilizing min-cut would be of any use. Presumably, the failure of any nodes with edges along a min-cut would be prone to generating a greater impact on the network load redistribution than the failure of nodes along the network periphery. A brief literature search did not reveal any decentralized algorithms for discovering min-cuts within the network. We think that if any such algorithm can be developed it would be of great benefit. Even if the discovery process took many network cycles, the mean time between cascades is long enough that it is safe to assume that each node knows whether it lies adjacent to a min-cut *prior* to entering a cascade. We think that this knowledge will greatly aid the awareness needed during the intentional removal process.

Many real-world networks do not cascade quite as quickly as inter-node communication, even though there is slight lead time that was not considered in our model. This situation corresponds to the percolation of vulnerable nodes throughout the network, as observed in [7], and strongly suggests that rate-estimation strategies, already known to be beneficial for load balancing and job scheduling [31], should fare much better than the simple reactive, average value based strategies considered here.

Unfortunately, the events which can trigger a cascade are *a priori* indistinguishable from events which do not trigger cascades [7]. This makes a global knowledge based solution extremely difficult, and might render a decentralized solution to the problem nigh impossible. Nevertheless, because we

often lack global knowledge of real-world network topology, and the infrastructure is almost never under a single controlling authority, we think that decentralized solutions are urgently needed.

REFERENCES

[1] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, Stanford, CA, Aug 1988, pp. 314–329. [Online]. Available: http://citeseer.ist.psu.edu/jacobson88congestion.html

[2] B. T. J. T. J. Borrvall, Charlotte; Ebenman, "Biodiversity lessens the risk of cascading extinction in model food webs," *Ecology Letters*, vol. 3, no. 2, pp. 131–136, Mar. 2000.

[3] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, "The yeast cell-cycle network is robustly designed," *PNAS*, vol. 101, no. 14, pp. 4781–4786, April 2004. [Online]. Available: http://dx.doi.org/10.1073/pnas.0305937101

[4] P. Ormerod and R. Colbaugh, "Cascades of failure and extinction in evolving complex systems," 2006. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:nlin/0606014

[5] A. E. Motter and Y.-C. Lai, "Cascade-based attacks on complex networks," *Physical Review E*, vol. 66, p. 065102, 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0301086

[6] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of networks," *Advances in Physics*, vol. 51, p. 1079, 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0106144

[7] D. J. Watts, "A simple model of global cascades on random networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 9, pp. 5766–5771, April 2002. [Online]. Available: http://dx.doi.org/10.1073/pnas.082090499

[8] E. C. Rose, "Vulnerabilites of network control protocols: an example," *ACM SIGSOFT Software Engineering Notes*, vol. 6, p. 68, January 1981. [Online]. Available: http://www.faqs.org/ftp/rfc/pdf/rfc789.txt.pdf

[9] J. B. Humphreyes P., "Dependent failure procedure guide," United Kingdom Atomic Energy Authority, Safety and Reliability Directorate, Tech. Rep. SRD-R-418, Mar. 1987.

[10] R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," *Nature*, vol. 406, p. 378, 2000. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0008064

[11] D. J. Carlson, J.M., "Highly optimized tolerance: A mechanism for power-laws in designed systems," *Phys. Rev. E*, vol. 60, pp. 1412–1427, 1999.

[12] T. C. Schelling, "Hockey helmets, concealed weapons, and daylight saving: A study of binary choices with externalities," *The Journal of Conflict Resolution*, vol. 17, no. 3, pp. 381–428, 1973. [Online]. Available: http://www.jstor.org/stable/173406

[13] A. E. Motter, "Cascade control and defense in complex networks," *Physical Review Letters*, vol. 93, p. 098701, 2004. [Online]. Available: doi:10.1103/PhysRevLett.93.098701

[14] M. Schäfer, J. Scholz, and M. Greiner, "Proactive robustness control of heterogeneously loaded networks," *Physical Review Letters*, vol. 96, no. 10, p. 108701, 2006. [Online]. Available: http://link.aps.org/abstract/PRL/v96/e108701

[15] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, p. 509, 1999. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/9910332

[16] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, December 1966. [Online]. Available: http://dx.doi.org/10.1007/BF02289527

[17] P. Hage and F. Harary, "Eccentricity and centrality in networks," *Social Networks*, vol. 17, no. 1, pp. 57 – 63, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/B6VD1-3Y6PD54-J/2/6e749011da729554919e7ed8b52e1714

[18] A. Shimbel, "Structural parameters of communication networks," *Bulletin of Mathematical Biophysics*, 1953.

[19] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, March 1977. [Online]. Available: http://links.jstor.org/sici?sici=0038-0431%28197703%2940%3A1%3C35%3AASOMOC%3E2.0.CO%3B2-H

[20] K. A. Lehmann and M. Kaufmann, "Decentralized algorithms for evaluating centrality in complex networks," 2002.

[21] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136 – 145, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6VD1-4RFJ4K1-1/2/3a32e9305e56b9844dfb72cd93c73aa6

[22] J. Winick and S. Jamin., Tech. Rep.

[23] C. H. Aldridge, "Scale-free networks using local information for preferential linking," in *MODSIM '05*, Dec. 2005. [Online]. Available: http://www.mssanz.org.au/modsim05/papers/aldridge.pdf

[24] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001.

[25] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Commun. ACM*, vol. 17, no. 11, pp. 643–644, 1974.

[26] S.-J. Lee and M. Gerla, "Dynamic load-aware routing in ad hoc networks," *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 10, pp. 3206–3210 vol.10, 2001.

[27] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: a dynamic traffic-aware routing algorithm," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 37–48.

[28] K. V. Vishwanath and A. Vahdat, "Realistic and responsive network traffic generation," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2006, pp. 111–122.

[29] J. Sommers and P. Barford, "Self-configuring network traffic generation," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 68–81.

[30] C. C. Moallemi and B. Van Roy, "Consensus propagation," *IEEE Transactions on Information Theory*, vol. 52, p. 4753, 2006. [Online]. Available: doi:10.1109/TIT.2006.883539

[31] L. M. Campos and I. D. Scherson, "Rate of change load balancing in distributed and parallel systems," *Parallel Computing*, vol. 26, 2000.