Analyzing single-server network inhibition

Tuomas Aura
Helsinki University of Technology
Lab. for Theoretical Computer Science
FIN-02015 HUT, Finland
Tuomas.Aura@hut.fi

Matt Bishop, Dean Sniegowski
University of California
One Shields Avenue
Davis, CA 95616-8562, USA
{bishop,sniegows}@cs.ucdavis.edu

Abstract

Network inhibition is a denial-of-service attack where the adversary attempts to disconnect network elements by disabling a limited number of communication links or nodes. We analyze a common variation of network inhibition where the links have infinite capacity and the goal of the attacker is to deny connections from a single server to as many clients as possible. The problem is defined formally and shown to be NP complete. Nevertheless, we develop a practical technique for network-inhibition analysis based on logic programming with stable-model semantics. The analysis scales well up to moderate-size networks. The results are a step towards quantitative analysis of denial of service and they can be applied to the design of robust network topologies.

1 Introduction

Network inhibition is a denial-of-service attack where the attacker disables network elements in order to disconnect communicating parties. We discuss a variation of the attack where the goal is to cut a single server (or a group of replicate servers) apart from as many clients as possible. This is a common scenario for analysis because most systems have only few mission-critical services and the analysis is initiated by the concern about their availability. We assume the network to have adaptive routing and links with infinite capacity so that a node is fully connected to the server as long as at least a single good route exits.

Network inhibition is one of the rare instances where there are well-defined quantitative measures for security against denial-of-service attacks. The goal of this paper is to add to this body of knowledge and to show that quantitative measures of robustness can be implemented in practice. We encode the networks as logic programs and use a general-purpose model finder *smodels* [13] for evaluating the seriousness of the attacks.

Most models of denial of service define the availability formally and then either attempt to prove that a particular system is fair in the presence of any malicious adversaries or propose an access control policies that guarantee the availability under all circumstances [18, 1, 12]. These models have been designed with the multi-user operating system of a single computer in mind. We argue that a more fruitful approach for distributed systems such as open communications networks is to evaluate the degradation of the services as a function of the cost to the attacker. It should be possible to compare degrees of security even if availability cannot be guaranteed under all circumstances. That way, the analysis methods will benefit real communications systems such as the Internet that can never be made provably secure against denial of service.

Sec. 2 overviews related work. Sec. 3 defines the networks and attacks against them. In Sec. 4, the problem of finding an optimal attack is shown to be NP complete. The following Sections 5 and 6 show how to encode the networks as a logic programs and how to find optimal attacks from the programs. Sec. 8 reports on an implementation and practical modelling techniques. The general principles for quantitative analysis of denial-of-service resistance are summarized in Sec. 9. Sec. 10 concludes the paper.

2 Background and related work

Although inhibition attacks on communications networks are a computer security issue, they have mostly been studied in graph-theoretical papers. In the graph model of a network, the edges of the graph represent communications links and the vertices of the graph are network nodes. The edges may have capacities that limit the flow of data though them. Representing the network as an abstract model has been advantageous for the construction of mathematical theories and algorithms.

An attack against the network is defined as removal of edges from the graph or reduction of their capacity. In practice, one should be equally worried about the possibility of vertices (network nodes) being removed. However, destruction of nodes can be reduced to attacks against links to and from them.

The goal of the attacker is to disconnect communicating parties or, in some models, to reduce the maximal transfer capacity of between them. In the time of wideband communications and adaptive routing, any single good route between two network nodes is often sufficient to carry all high-priority traffic until the failed components have been repaired. This is particularly true for military communications where the systems are designed to be survivable with redundant capacity and strict priorities for critical data. The reduction of capacity might be more serious for a commercial operator of an open network whose customers will experience a drop in service quality.

Naturally, an attacker with unbounded resources can defeat any network. We are interested in the cases where the attacker's resources are limited. The limitations are represented in the models by a cost for disabling each communication link (or node) and a fixed budget for the attacks. The malicious adversary will, assuming it has enough knowledge of the network structure, use its resources in a way that maximizes the damage. This is different from statistical models of network reliability where network elements are assumed to fail randomly and independently, and the worst case scenario is unlikely to occur.

In order to measure the damage, we also need to know the value of the lost connections to the defender. The models from the literature mentioned below differ mostly in the way the damage is evaluated. Otherwise, all the models build on the same theory of minimal graph cuts and maximum flow.

The simplest problem is to deny the connection between two given nodes. An optimal attack can be found in polynomial time with any MIN CUT algorithm. The best known algorithm is by Stoer And Wagner [16]. Cutting all connections between two groups of nodes is equally easy. It is reduced to the minimal cut problem by merging both groups into single nodes.

Cunningham [4] solves another polynomial problem: how to partition a network into separate components at the lowest cost. Each edge has an associated cost. Optimality is defined as the lowest cost per created network component. Cunningham also discusses the optimal reinforcement of a network against the attack.

A more complex problem is the *multi-way cut*. The goal is to disconnect 3 or more given nodes from each other. Dahlhaus & al. [5] show this problem to be NP complete. It follows that the more general problem of disconnecting three or more arbitrary pairs of nodes is also intractable.

Phillips [15] introduces the *network inhibition* problem that is like MIN CUT but the links can be partially disabled with linearly increasing cost. Rather surprisingly, it turns out that finding an attack that minimizes the residual capacity of the network between the two nodes is NP complete. Like [4], this paper is interesting because it models the damage to the network as a function of the investment by the attacker.

In the reminder of this paper, we will analyze another variation of the network inhibition problem that we find to be especially relevant in protecting Internet services. We will try to determine *how many communication links or network nodes need to be removed from the communications network to disconnect a given number of nodes from a single center node*, called the *source*. The idea is to analyze the damage to the connectivity from a single host's point of view. When the network is modelled as a (directed) graph, this translates to the question if finding a minimal weight cut that reduces the size of the graph partition with the source node in it below a given threshold. Despite of the similarities with the other network inhibition problems, this appears to be an independent question. We have named this the *single-server network inhibition* problem although it might also be called *minimum multicast cut* since the attacker might be trying to minimize the number of receivers for a multicast (or broadcast) transmission. The links and nodes in our model may have have two kinds of weights: a cost of disabling and a value of being connected to the source. The links are assumed to have infinite capacities. Partial destruction of an edge is thus not possible. The following section formalizes the network and attack models.

3 Network and attack models

We model the communications network as a directed graph. To accommodate communications terminology, the vertices are called *nodes* and the directed edges are called *links*. The nodes and links can be given weights to denote their relative importance and robustness.

Definition 1 (communications network) A *communications network* is a quadruple $CN = \langle N, L, s, c, d \rangle$ where

1. N is set of *nodes*,
2. $L \subseteq N \times N$ is a set of *links*,
3. $s \in N \cup \{\epsilon\}$ is called the *source* node,
4. $c : N \cup L \to \mathbb{Z}_+ \cup \{\infty\}$ is called a *cost* function, and
5. $d : N \to \mathbb{N}$ is called *damage* function.

□

The source node is a server or a client from whose perspective the analysis will be done. The source can also be ϵ which means that the source itself has been disabled. The cost function gives the cost of disabling nodes or links for the attacker. Disabling these components will be modelled by removing vertices and edges from the directed graph. The costs are positive integers, or infinite meaning that particular network components cannot be disabled. The damage function tells how valuable it is to the defender to have each node connected to the source. The damage values are nonnegative integers.

In an attack, some nodes and links are disabled. Formally, an *attack* on a network is a set of *disabled* nodes and links. For networks where all links are bidirectional, an attack that succeeds in partitioning the network corresponds to a *cut* of the graph $\langle N, L\rangle$. The total cost of the attack for the attacker is the sum of the costs of disabling the individual nodes and links.

Definition 2 (cost of attack, remaining network) Let $CN = \langle N, L, s, c, d\rangle$ be a communications network and $A \subseteq N \cup L$ an attack on it. The *cost* of the attack is

$$\mathit{Cost}_{CN}(A) = \sum\{c(x) \mid x \in A\}.$$

Denote the disabled nodes by $A_N = A \cap N$, the disabled links by $A_L = A \cap L$, and the reverse links by $A_L^{-1} = \{\langle n, m\rangle \mid \langle m, n\rangle \in A_L\}$. The *remaining network* after an attack A is

$$CN^A = \\ \langle N \setminus A_N,\ L \setminus (A_L \cup A_L^{-1} \cup A_N{\times}L \cup L{\times}A_N),\ s', c', d'\rangle$$

where c' and d' are, respectively, the restrictions of c and d to the nodes and links of CN^A. $s' = \epsilon$ if $s \in A$ and $s' = s$ otherwise. □

The communications links are modelled as directed edges. However, most real links are bidirectional. We have chosen to represent bidirectional links as two unidirectional edges. The above definition ensures that both directions of a bidirectional link fail at the same time. It is enough to include one of them in the attack because including both links in the attack would increase the cost of the attack but not affect connectivity. This is a practical choice and it has no bearing on the generality of the model: two independent links in opposite directions can be represented by adding dummy nodes with zero damage values on the links.

Our model of the network assumes the use of adaptive routing algorithms and it ignores the capacity limitations of the communication links. As long as there at least one good route from the source to a node, the service is available. The service is denied when the last connection from the source fails. The lack of capacity bounds is probably the greatest limitation of the model.

We say that a node is *connected to the source* if there is a path in the directed graph $\langle N, L\rangle$ from s to the node. That is, the nodes that are available for the source (or for which the source is available), are in the closure of the source node with respect to the link relation. The nodes that are not connected to the source are *disconnected*. When all links are bidirectional, the network can be interpreted as an undirected graph. In that case, nodes become disconnected when the network is partitioned. Only the nodes in the same partition with the source are connected to it.

In practice, all nodes in the original network will be connected to the source. When some nodes and edges are disabled in an attack, the disabled nodes and possibly some other nodes become disconnected from the source. The success of the attack is measured by the number and importance of the nodes that it manages to disconnect from the source.

Definition 3 (damage) Let $CN = \langle N, L, s, c, d\rangle$ be a communications network and A an attack on it. The *damage* caused by A is defined as

$$\mathit{Damage}_{CN}(A) = \\ \sum\{d(n) \mid n \in N \text{ and } n \text{ is disconnected from } c \text{ in } CN^A\}.$$

□

Obviously, if the source is disabled in an attack (i.e. the source of CN^A is ϵ), all nodes become disconnected. The cost of disabling the source should usually be very high or infinite.

In order to find the best attacks or to assess the reliability of the network, we need to find an attack that causes maximal damage with a given cost.

Problem 4 Maximize $\mathit{Damage}_{CN}(A)$ over all attacks $A \subseteq N \cap L$ for which $\mathit{Cost}_{CN}(A) \le C$. □

In the graph terminology, this means minimizing the total weight of the nodes reachable from the source (or the total weight of the source partition for undirected cases) with a given total weight of disabled edges.

An equally interesting problem is to minimize the cost for a desired damage:

Problem 5 Minimize $\mathit{Cost}_{CN}(A)$ over all attacks $A \subseteq N \cap L$ for which $\mathit{Damage}_{CN}(A) \ge D$. □

Since the allowed cost or damage is a parameter, we would actually want to plot the optimal solutions for different parameter values. That is, we want to compute the damage caused by optimal attacks as a function of the cost to the attacker.

4 Complexity of computing the cost-damage curve

In this section, we will show that the problem of determining whether a given attack causes maximal damage for

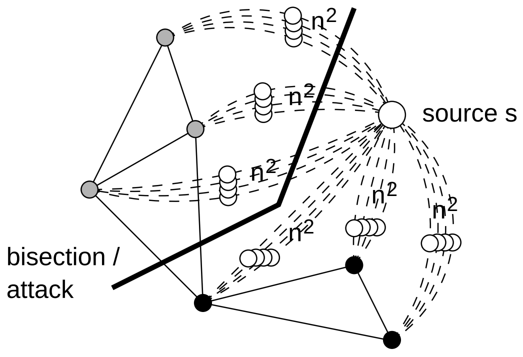

Figure 1. Reduction of graph bisection to Problem 6 (n=6)

its cost is an NP complete problem. The proof is done by a reduction from the minimal graph bisection problem. This is not surprising given similar results on other closely related graph problems (see Sec. 2). Furthermore, the problem remains NP complete even if all links are bidirectional, only links can be disabled, the cost of disabling a component is constant and the damage value of a node is constant.

Since NP completeness is proven for decision problems (ones that return answer yes or no), we have to restate the optimization problem in the following way.

Problem 6 Let $CN = \langle N, L, s, c, d\rangle$ be a communications network. Does an attack $A \subseteq N \cap L$ exist such that $\mathit{Cost}_{CN}(A) \leq C$ and $\mathit{Damage}_{CN}(A) \geq D$? □

If we can find an optimal solution for a fixed C or D, clearly we can answer the decision problem. Thus, the optimization is at least as hard as the decision.

Theorem 7 Problem 6 is NP complete with the size of the network even if all links are bidirectional, $d(n) = 1$ for all $n \in N$, and $c(n) = \infty$ for all $n \in N$ and $c(l) = 1$ for all $l \in L$. □

Proof First, the problem is in NP. A naive non-deterministic program could guess the state of each link (disabled or not), count the disabled links to check that there are at most C and to check by depth-first search from the source that at most $|N| - D$ nodes remain connected to the source.

To show NP hardness, we will describe a polynomial-time reduction from the *minimal graph bisection problem* (i.e. *minimum cut into bounded sets* or *bisection width*) [7, 8, 14] that is known to be NP complete. The goal in graph bisection is to divide a graph into two equal-size partitions so that the number of edges between the two partitions is less than some B.

Let a graph $G = \langle V, E\rangle$ with some even $n = |V|$ be given for the bisection problem. We add to the graph a new node s (source) and connect it to each of the the original vertices v through n^2 routes. Note that n^2 is larger than the number of edges in any bisection of G. (Actually, $n^2/4 + 1$ routes would suffice but it will not hurt to be generous.) Each route consists of a new auxiliary node a_v^i, an edge from the source to the auxiliary node, and an edge from the there to the original vertex. The resulting graph will be considered as network and, therefore, it is denoted by $\langle N, L\rangle$. Formally,

$$N = V \cup \{s\} \cup \{a_v^i \mid v \in V \text{ and } i = 1 \ldots n^2\}$$

$$L = E \cup \{\langle s, a_v^i\rangle, \langle a_v^i, s\rangle, \langle a_v^i, v\rangle, \langle v, a_v^i\rangle \mid \\ v \in V \text{ and } i = 1 \ldots n^2\}.$$

(E includes both $\langle v, v'\rangle$ and $\langle v', v\rangle$ since the original graph is undirected.) $\langle N, L\rangle$ has $n^3 + 1$ new vertices and $2n^3$ new edges. This may seem like a lot but it is, nevertheless, a polynomial increase.

To define a communications network $CN = \langle N, L, s, c, d\rangle$, let c be 1 for all links and infinite for all nodes, and let d be 1 for all nodes.

We claim that the original graph G has a bisection of size B, $1 \leq B \leq n^2/4 < n^2$, or less *if and only if* the new network CN has an attack with damage at least $n^3/2 + n/2$ and with cost not higher than $n^3/2 + B$. The situation is illustrated in Fig. 1.

(*only if*) Assume first that such a bisection exists. Include in the attack the B or less edges of the bisection (the ones that cross between the two partitions). Moreover, select one of the partitions and include in the attack the $n^3/2$ edges that connect the auxiliary nodes of that partition to the source. This disconnects $n^3/2$ auxiliary nodes plus half ($n/2$) of the original vertices from the source.

(*if*) Assume that an attack satisfying the conditions $\mathit{Damage}_{CN}(A) \geq n^3/2+n/2$ and $\mathit{Cost}_{CN}(A) \leq n^3/2+B$ exists. It is impossible for the attack to disconnect more than half of the vertices V from the source. This is because to disconnect a vertex, all n^2 routes through auxiliary nodes must be disconnected and $n^3/2 + B < (n/2 + 1)n^2$. Suppose then that the attack would disconnect $k < n/2$ of the vertices V. We will see that the attack cannot cause enough damage. The routes to the k vertices through corresponding auxiliary nodes must be cut thus disabling kn^2 links and disconnecting, at no extra cost, up to kn^2 auxiliary nodes in addition to the k vertices. At least $(n^3/2 + n/2) - (kn^2 + k)$ nodes must still be disconnected by disabling at most $(n^3/2 + B) - kn^2 \leq n^3/2 - kn^2 + n^2/4$ links.

Since no more original vertices may be disconnected, these nodes must all be auxiliary nodes whose corresponding vertex in V is still connected. Therefore, links on

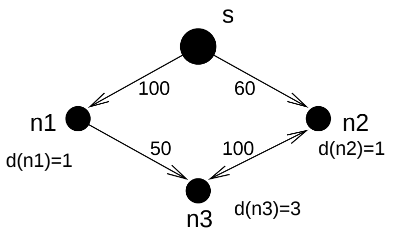

Figure 2. A simple communications network

both sides of these auxiliary nodes must be disabled which would cost a total $2 \cdot ((n^3/2 + n/2) - (kn^2 + k)) = n^3/2 - kn^2 + (n/2 - k)(n^2 + 2)$. But since $n/2 - k \geq 1$ and $n^2 + 2 > n^2/4$, this is more than we can afford. The conclusion is that that to get the desired damage, the attack must disconnect exactly $n/2$ original vertices V.

Disconnecting the routes to the $n/2$ vertices through their auxiliary nodes costs $n^3/2$ disabled links, leaving a budget of only B for further work. However, the $n^3/2$ auxiliary nodes become disabled at no additional cost. All that still needs to be done is to separate the $n/2$ disconnected vertices of V from the $n/2$ connected ones. If this can be done by disabling at most B edges of the original graph, there is a bisection of weight B or less in G.

This suffices to show that there is an attack with the given properties exactly when there a bisection meeting the given budget. Hence, any algorithm for deciding the existence of the attack can also decide the existence of the bisection. Since the transformation was polynomial, Problem 6 is also NP complete. □

Theorem 7 remains true also in networks where the nodes break and links are unbreakable. This can easily be shown by adding a breakable node in the middle of each link.

5 Networks as a logic programs

In this section, we show how a communications network can be described with a logic program. Consider, for example, the (artificially small) network in Fig. 2. The network has three unidirectional links and one bidirectional link. The cost of disabling the links is marked on them. Nodes cannot break (i.e. their cost is infinite). The damage values of the nodes are also marked on them. (The damage value of the source node s does not matter because the nodes here cannot break and the source will therefore never be disconnected.)

The corresponding logic program is in Fig. 2. Lines 1–10 describe the network structure. The other lines give rules about how nodes are connected to the source. These lines will be the same for all networks. This simple program is all it takes to model the behavior of the network with breaking components.

The rules on Lines 12-15 force each node and link to be either broken or ok but not both. As we will explain in Sec. 6, without these rules, the program could have models where some network components are neither broken nor ok.

Lines 17-19 of the program give the basic rules of connectedness. For any link, if the node at the beginning of the link is connected, the link is not broken and the node at the end is not broken itself, then the end node is also connected. These rules force the predicate `connected` to be true for all the nodes that are connected to the source. We will choose a semantics for the logic programs in such a way that the predicate will be true only for this minimal set of nodes and not for any other ones. Thus, we don't need any special rules for saying when a node is disconnected.

Note that our formal definition allows a bidirectional link to be broken in one or both directions. On the other hand, the connectedness rule (line 18 in Fig. 3) says that breaking in one direction is enough to disable communication in both directions. It may seem that one could mistakenly double the cost of disabling the link by putting both directions of the link in the attack set A. This problem is avoided because we are interested only in attacks that with minimal cost for a given damage. An optimal attack plan never tries to disable the same link twice.

The logic-program representation of any communications network can can be constructed in the same way. The structure of the network is described with the predicates `node`, `link` and `source`, and the lines 12-19 of Fig.3 are copied as such.

6 Stable models as attack scenarios

We will interpret the logic programs according to the *stable model semantics* of Gelfold and Lifschitz [9]. The stable models are defined for a *ground* logic program, i.e. one without variables. Therefore, we have to first remove variables from the program by substituting them with all possible constant value combinations. Our programs are *strongly range-restricted*: the possible values of all the variables are those listed in the `node` and `link` predicates. Such programs can be grounded efficiently. In the experiments, we used an implementation by Syrjänen [17]. After grounding, the program has grown in size but it is variable-free. The predicates with only constant arguments are called *atoms*. Atoms and negated atoms are called *literals*.

The different semantics for logic programs differ mostly in the way they interpret the negation. (For example, a negative literal in Prolog is true if Prolog's resolution strategy fails to prove that the positive literal is true.) A stable model is a set of atoms that passes the following test: (1) For each atom in the model, remove from the program all rules that

```
 1 node(s).
 2 node(n1).
 3 node(n2).
 4 node(n3).
 5 source(s).
 6 link(s,n1).
 7 link(s,n2).
 8 link(n1,n3).
 9 link(n2,n3).
10 link(n3,n2).
11
12 nodeOk(N) :- node(N), not nodeBroken(N).
13 nodeBroken(N) :- node(N), not nodeOk(N).
14 linkOk(N,M) :- link(N,M), not linkBroken(N,M).
15 linkBroken(N,M) :- link(N,M), not linkOk(N,M).
16
17 connected(M) :- link(N,M), connected(N), not nodeBroken(M),
18                 not linkBroken(N,M), not linkBroken(M,N).
19 connected(N) :- source(N), not nodeBroken(N).
```

Figure 3. Logic program representation of the network in Fig. 2

have a negative literal for that atom in their bodies (on the right side of the implication). The idea is that these rules do not apply to this model. (2) Remove all negative literals from the bodies of the remaining rules. The result is a program without any negative literals. Such a program has a unique minimal Herbrand model (a set of atoms that satisfies the implication in each rule of the program) that can be computed simply as a closure of an empty set of atoms with respect to the remaining rules. To pass the test, that unique model must be exactly the same set of atoms as the original stable model. Thus, a stable model is a fixpoint of this process of computing what is called the *reduct* of the program and its unique minimal Herbrand model. A program may have several stable models (e.g. `a :- not b. b :- not a.`) or none (e.g. `a :- not a`).

The stable models are a natural way of defining the meaning of a logic program. For most applications, they are the possible sets of conclusions that a rational agent might make from the program. The stable models of our programs correspond to all possible attacks against the protocol. The atoms in each stable model describe accurately the state of the network (disabled components, connected nodes) after the corresponding attack.

We will formulate this as a somewhat informal proposition. A more rigorous theorem and proof would require a formal definition of the transformation from communication networks to the logic-program representation. We find the example above to be more illustrative.

Proposition 8 Let $CN = \langle N, L, s, c, d\rangle$ be a communications network. There is a 1-1 mapping between the stable models of the logic-program representation of CN and the possible attacks $A \subseteq N \cup L$ against the network. The stable model corresponding to an attack A contains the atom `connected(`n`)` for each node n if and only if the node n is connected to the source in the remaining network CN^A.
□

Proof Let Π be the logic-program representation of CN and let $A \subseteq N \cup L$ be an attack. Define M_A to be a set containing exactly the following groups of atoms:

1. `node(n)` for each node $n \in N$,
   `link(n,m)` for each link $\langle n, m\rangle \in L$, and
   `source(s)` for the source node s

2. `nodeBroken(n)` for nodes $n \in A$,
   `nodeOk(n)` for node $n \in N \setminus A$,
   `linkBroken(m,n)` for links $\langle m, n\rangle \in A$, and
   `linkOk(m,n)` for links $\langle m, n\rangle \in L \setminus A$

3. `connected(n)` for nodes that are connected
   to the source in CN^A.

We claim that M_A is stable model for Π. The reduct of Π with respect to M_A has the following rules:

a. all the facts about the network structure
   (like lines 1-10 in Fig. 3).

b. `nodeOk(n) :- node(n).` for all $n \in N \setminus A$,
   `nodeBroken(n) :- node(n).` for all $n \in A$,
   `linkOk(n,m) :- link(n,m).`
   for all $\langle n, m\rangle \in L \setminus A$, and
   `linkBroken(n,m) :- link(m,n).`
   for all $\langle n, m\rangle \in A$.

c. `connected(m) :- link(n,m),connected(n).` for all links $\langle n,m\rangle$ such that $\langle n,m\rangle \in L \setminus A$, $\langle m,n\rangle \in L \setminus A$ and $n \in N \setminus A$, and
`connected(s) :- source(s).` for the source node s if $s \in N \setminus A$.

The next step is to find the unique minimal Herbrand model M of the reduct. The rules of Type a cause M to contain all the atoms of Group 1. These are `node(n)` for all nodes $n \in N$, `link(n,m)` for all directed links from a node n to m, and `source(s)` for the source s. No other atoms for these predicates can be in M.

Since `node(n)` $\in M$ for all nodes, the rules of Type b add to M atoms `nodeOk(n)` and `nodeBroken(n)` for all nondisabled and disabled nodes, respectively. Similarly, they add to M the correct atoms for nondisabled and disabled links. Hence, all the atoms of Group 2 are in M. No other atoms for these predicates can be in M because these are the only rules that have the predicates on the left side.

The third group of atoms requires a bit more thought. As Def. 2 says, CN^A is obtained from the original network CN by removing the disabled nodes, disabled links, and any links attached to the removed nodes. Thus, a node n is connected to s in CN^A iff there is a directed path from s to n in CN that does not include any of the links or nodes in A. It is not difficult to see that the rules of Type c induce `connected(n)` to be true for exactly these nodes n. This can be formally shown by induction on the length of the path and on the length of the proof with the rules of Type c. Hence, the model M contains all the atoms of Group 3 and no other atoms with predicate `connected`. This suffices to show that $M = M_A$ passes the test for a stable model.

We now know that Π has at least one stable model corresponding to each $A \subseteq N \cup L$. We still need to show that it has no other stable models. Every stable model of Π must have all the atoms of Group 1. Because of the lines 12–15 of Fig. 3, every stable model must also have either `nodeOk(n)` or `nodeBroken(n)` for every node $n \in N$, and either `linkOk(n,m)` or `linkBroken(n,m)` for every link $\langle n,m\rangle \in L$. This choice of broken nodes and links defines a unique attack A'. As above, it can be shown by induction on the length of the proof that the closure of these atoms includes `connected(n)` for all nodes that are connected to s in $CN^{A'}$. Therefore, any stable model of Π will have at least all the atoms of M_A for some attack A. But it is not possible for the stable model to be proper superset of M_A because stable models are always minimal Herbrand models [9]. This allows us to finally conclude that the models M_A are the only stable models of Π. □

7 Optimal attacks

The previous section showed that the stable models of the logic program representation of a communications network accurately describe the different attacks on the network. This makes it possible to use a general-purpose logic programming system to find optimal attacks and to answer Problems 4–6. We have used *smodels*, an efficient stable models implementation by Niemelä and Simons [13]. While the program of Fig. 3 is a standard logic program, the additions that we make to it in this section are specific to *smodels*.

The simplest way to find the optimal attack would be to enumerate the attacks, to compute the closure of the connected nodes in each remaining network, and to select one with cost below the budget B and highest damage. This approach will work well for small networks like that in Fig. 3. It will not work for even moderately large networks as the number of possible attacks $2^{|N|+|L|}$ grows exponentially with the size the the network. Branch-and-bound techniques and heuristics that find good (although not necessarily optimal) solutions quickly can improve the efficiency. However, it may be difficult to develop heuristics that are suitable for a particular problem domain. The idea in using a general-purpose logic programming system to find the solutions is that such systems already have efficient and well-tested implementations of the search.

The additional rules that *smodels* needs for the task are listed in Fig. 4. The rule on lines 20–24 tell which attacks are too expensive: if the total cost of the disabled components in a model is 151 or more, the atom `false` will also be included. (Here `false` is an atom, not a keyword with any special meaning.) This rule is simply a shorthand notation and could be expanded to a standard logic program by replacing it with all combinations of broken components whose weight exceeds the budget 150. The shorthand notation is not only more convenient for the programmer but also saves time and space in finding the solution. Line 25 simply creates a contradiction if the atom `false` is in the model. Therefore, the program cannot have any stable models with `false` in them.

Lines 27-29 are directives for *smodels*. They ask it to consider all stable models and to find one with the minimal sum of the weights of the `connected` atoms.

8 Implementation

We have implemented an experimental tool for analyzing the robustness of a network topology. The user inputs a network structure into a graph editor, the network is translated into a logic program representation, *smodels* is invoked to find the optimal attack for a given budget, and the result is shown on the graph.

```
false :- 151 [ linkBroken(s,n1)=100, linkBroken(s,n2)=60,
               linkBroken(n1,n3)=50, linkBroken(n2,n3)=100,
               linkBroken(n3,n2)=100,
               nodeBroken(s)=1000, nodeBroken(n1)=1000,
               nodeBroken(n2)=1000, nodeBroken(n3)=1000 ].
f :- not f, false.

compute 0 { }.
minimize { connected(s)=0, connected(n1)=1,
           connected(n2)=1, connected(n3)=3 }.
```

Figure 4. Finding optimal attacks with smodels

Some general guidelines can be given for creating the network model:

- A subnetwork behind a single gateway router should be represented by a single node. The weight (damage value) of the node can be used to encode the size of the subnetwork. This will greatly reduce the model size.

- A LAN with a broadcast architecture (e.g. Ethernet) should be modelled as an auxiliary node that may break itself but is connected to the stations on the LAN by indestructible links. That is, the broadcast LAN either functions or fails for all stations.

- It is important to distinguish between a single bidirectional link and two unidirectional links. The former is represented by two link predicates and the latter by two auxiliary nodes. (See Sec. 3.)

- A group of replicate servers can be either combined into a single source node or declared separately as sources in the logic program. This can be used to compare different choices for the placement of replicate servers.

The analysis method was tested with random sparse graphs and artificial network models. Results from random graphs of up to 100 nodes are encouraging although the performance depends heavily on the density of the graph. Tests should still be done with actual networks. We expect the results for real communications networks to be at least as good as they are for the random graphs as real nets are usually sparsely connected and tend to have mostly local dependencies. (It is difficult to get hold of sufficiently large maps of an actual networks for the tests.)

In our implementation, the logic program representation was a slightly optimized version of the one in Figs. 3 and 4. There is still room for further optimization, in particular, in preprocessing of the graph before the translation to a logic program.

9 Quantitative analysis of denial of service

In the view of the previous sections, we will try to form general guidelines for denial-of-service models. The main idea is to express the damage caused by an attack as a function of the resources required for its execution. When determining the seriousness of a threat or comparing architectures, one should inspect the whole range of the function. By considering a range of optimal attacks, we avoid setting any awkward thresholds for when an attack is successful and when not. Meadows [11] presents a formal method for evaluating resistance of cryptographic protocols against denial of service and suggests a similar measure of robustness where cost-damage points of the attacks are compared to an application-specific tolerance level.

Formally, let *ATTACKS* be the set of possible attacks. The cost function $\mathit{Cost} : \mathit{ATTACKS} \rightarrow \mathit{COSTS}$ is a mapping from the attacks to the costs of the attacks and the damage function $\mathit{Damage} : \mathit{ATTACKS} \rightarrow \mathit{DAMAGES}$ is a mapping from the attacks to the damages caused by the attacks.

The *COSTS* and *DAMAGES* are sets of scalar or vector values. In the simplest case, they may be dollar values. The cost is often a vector of the various resources needed for the attack: communications bandwidth, computational power, number and type of conspiring entities, required access rights, etc. Damages are more likely to be scalars because of the need to put the threatened services in an order by their importance. However, it may sometimes be necessary to express the damage in components, such as lost money and time. Possible vector values are compared by components. Thus, scalar values have a natural linear order and vector values a partial order.

The main question that we want to answer is

Problem 9 For given $C \in \mathit{COSTS}$ and $D \in \mathit{DAMAGES}$, decide whether there is an attack $A \in \mathit{ATTACKS}$ such that $\mathit{Cost}(A) \leq C$ and $\mathit{Damage}(A) \geq D$? □

If the damages are linearly ordered (scalars), the same question can be formulated as an optimization problem.

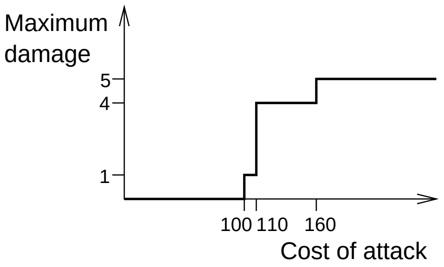

Figure 5. Optimal cost-damage curve

Problem 10 For a given $C \in$ *COSTS*, find an attack A such that *Damage*(A) is maximal over all $\{A \in$ *ATTACKS* $|$ *Cost*$(A) \leq C\}$. □

The best picture of the seriousness of the attacks is obtained by plotting the damage caused by the optimal attacks as a function of the allowed attack cost. For example, Fig. 5 shows how the damage varies as a function of the attack cost in the network of Fig. 2. The plots for different system architectures can be compared to evaluate their relative robustness under attack. When the cost is a vector-valued quantity, a single two-dimensional plot cannot satisfactorily depict the function. In that case, comparisons of the functions take more time and effort.

Similarly, the seriousness of two types of attacks could be compared by plotting the maximal damage caused by attacks of either type. When the damage is measured in more-than-one-dimensional vector values, each component of the damage must be plotted separately.

An equally interesting problem is to minimize the cost of an attack for a desired level of damage, although this only makes sense if the cost is measured in scalar values.

Problem 11 For a given $D \in$ *DAMAGES*, find an attack A such that *Cost*(A) is minimal over all $\{A \in$ *ATTACKS* $|$ *Damage*$(A) \geq D\}$. □

It should be mentioned that it is often impossible or unnecessary to calculate the accurate damage function values. Nevertheless, these concepts can be used to argue about the relative greatness of the damages or costs and about the effects of changes in the system architecture.

The comparison approach has been successfully applied in some areas completely different from network inhibition. Aura and Nikander [2] compare the robustness of stateless and stateful servers and protocols. Dwork and Naor [6] suggest increasing the cost of sending junk mail and Hirose and Matsuura [10] and Aura, Nikander and Leiwo [3] design key-agreement protocols where the attacker is always the first to commit to expensive computations. The general idea is that a system is considered robust if *Cost*$(A) >$ *Damage*(A) for all attacks.

To make the analysis easier in practice, attacks and damages can be evaluated separately from each interested party's point of view. In open systems where the participants have few common interests, it does not make sense to model the whole system. It is often much easier to consider the threats only from a single entity's perspective. For example, it is different to consider the availability of a service with the server's or a client's interest in mind. For the server, the goal is to allow as many clients as possible to efficiently use the service. For the single client, it is important to obtain the desired services from any of possibly many alternative servers. Rarely is it necessary for every client to be able to access every service, and even if it is, there may be no authority that would want to invest in improving the availability for all these parties. This could be summarized by saying that doing the analysis with the paying client in mind rather than for the public good makes it easier to get useful results.

10 Conclusion

We defined the single-server network inhibition problem and showed it to be NP complete like many related problems. Logic programs with stable-model semantics were used to represent the network. Optimal attacks were found with a tool that implements this semantics. The techniques of this paper can be used to analyze the robustness of network architectures against denial-of-service by link and node destruction.

Possible future work includes optimizing the search for attacks by reducing parts of the networks at a preprocessing stage and evaluating other techniques such as integer linear programming and simulated annealing to solve the same problem. The ultimate goal should be to develop vulnerability detection techniques that can be incorporated as standard components into network engineering tools.

We also suggested that the analysis of denial-of-service attacks should, in general, aim to give the damage caused by optimal attacks as a function of the cost to the attacker rather than setting sharp thresholds for availability and denial. This makes it possible to compare the robustness of systems under attack even when it is impossible to guarantee the availability.

Acknowledgments

This research was done mostly while Tuomas Aura was visiting UC Davis Computer Security Laboratory. His work was funded by Academy of Finland projects #44806 and #47754. Matt Bishop Dean Sniegowski were were supported by grant NAG21251 from the National Aeronautics and Space Administration (NASA), and by a gift from Microsoft Corporation. The work benefited greatly from discussions with Ilkka Niemelä.

References

[1] Edward Amoroso. A policy model for denial of service. In *Proc. Computer Security Foundations Workshop III*, pages 110–114, Franconia, NH USA, June 1990. IEEE Computer Society Press.

[2] Tuomas Aura and Pekka Nikander. Stateless connections. In *Proc. International Conference on Information and Communications Security (ICICS'97)*, volume 1334 of *LNCS*, pages 87–97, Beijing, China, November 1997. Springer Verlag.

[3] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant authentication with client puzzles. In *Proc. Security Protocols Workshop 2000*, Cambridge, UK, 2000. Springer. To appear.

[4] William H. Cunningham. Optimal attack and reinforcement of a network. *Journal of the ACM*, 32(3):549–561, July 1985.

[5] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). In *Proc. 24th Annual ACM Symposium on Theory of Computing (STOC'92)*, pages 241–251, Victoria, Canada, May 1992. ACM Press.

[6] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - Proc. CRYPTO '98*, volume 740 of *LNCS*, pages 139–147, Santa Barbara, CA USA, August 1992. Springer-Verlag.

[7] Michael R. Garey and David S. Johnson. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.

[9] M. Gelfold and V. Lifschitz. The stable model semantics for logic programming. In *Proc. 5th International Conference on Logic Programming*, pages 1070–1080, Seattle, WA USA, August 1988. The MIT Press.

[10] Shouichi Hirose and Kanta Matsuura. Enhancing the resistance of a provably secure key agreement protocol to a denial-of-service attack. In *Proc. 2nd International Conference on Information and Communication Security (ICICS'99)*, pages 169–182, Sydney, Australia, November 1999. Springer.

[11] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 4–13, Mordano, Italy, June 1999. IEEE Computer Society.

[12] Jonathan K. Millen. A resource allocation model for denial of service. In *Proc. 1992 IEEE Computer Society Symposium on Security and Privacy*, pages 137–147, Oakland, CA USA, May 1992. IEEE Computer Society Press.

[13] Ilkka Niemelä and Patrik Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proc. 4th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 1265 of *LNCS*, pages 420–429, Dagstuhl, Germany, July 1997. Springer.

[14] Christos M. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[15] Cynthia A. Phillips. The network inhibition problem. In *Proc. 25th Annual ACM Symposium on the Theory of Computing*, pages 776–785. ACM Press, May 1993.

[16] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, July 1997.

[17] Tommi Syrjänen. Implementation of local grounding for logic programs with stable model semantics. Technical Report B18, Digital Systems Laboratory, Helsinki University of Technology, October 1998.

[18] Che-Fn Yu and Virgil D. Gligor. A formal specification and verification method for the prevention of denial of service. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 187–202, Oakland, CA USA, April 1988. IEEE Computer Society Press.