Analyzing single-server network inhibition

Presented by **Maarit Hietalahti**

Tuomas Aura

Helsinki University of Technology

Laboratory for Theoretical Computer Science

`Tuomas.Aura@hut.fi`

Matt Bishop, Dean Sniegowski

University of California, Davis

Computer Security Laboratory

{bishop,sniegows}@cs.ucdavis.edu

CSFW, July 4th, 2000

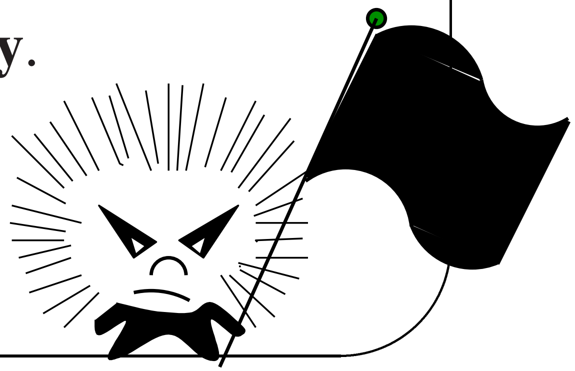
Contents

Denial-of-service (DOS) attacks by the destruction of communications links.

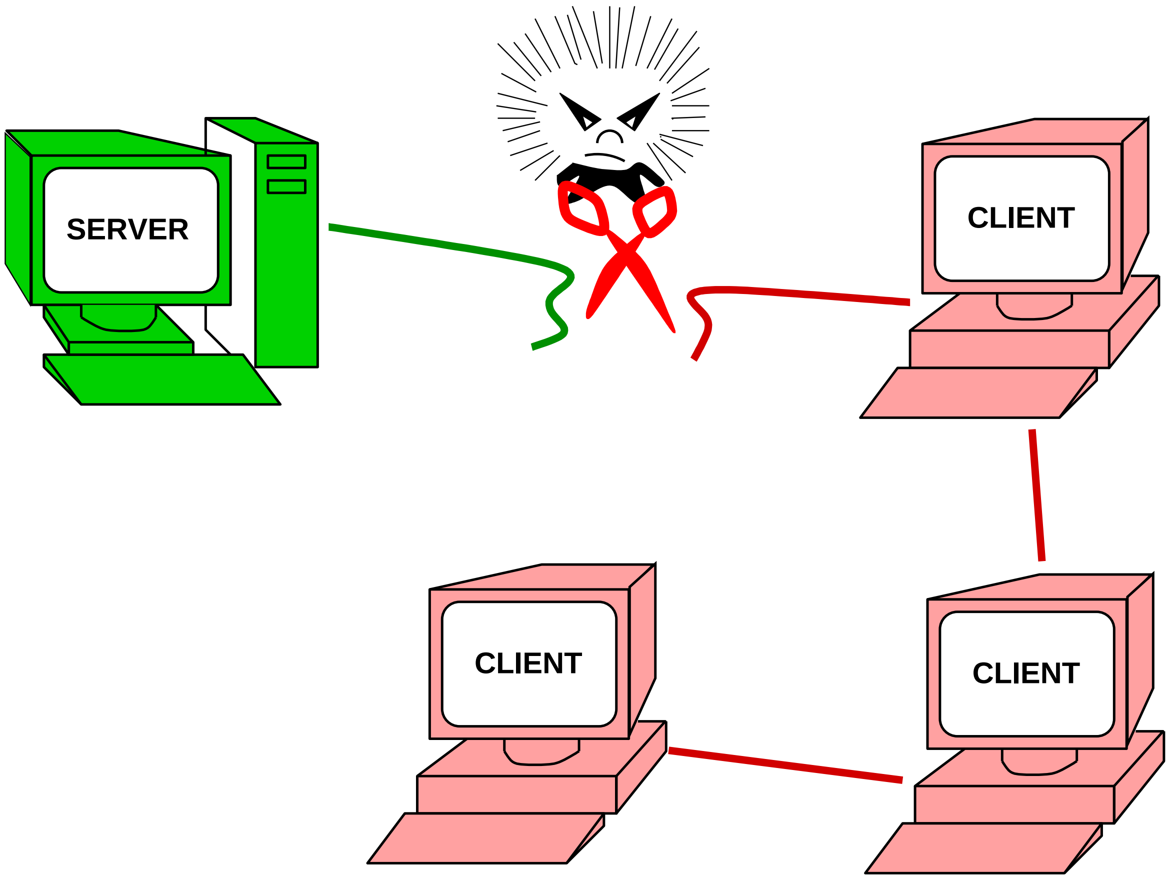
1. Introduction to network inhibition attacks
2. Formal definition and theoretical complexity of finding optimal attacks
3. Logic-program solution

Network inhibition

- Denial-of-service (DOS) attacks against a communications networks: **disconnecting nodes from each other by destroying links (and nodes)**.
- Attacker has a **limited budget** and tries to **maximize damage**, or a tries to minimize the cost of causing a given damage.
- In this paper:
  Adaptive routing assumed. No capacity limitations.
  No partially broken links.
  $\Rightarrow$ Sufficient to have one good path from source to destination.
- The success of an attack depends on the **network topology**.

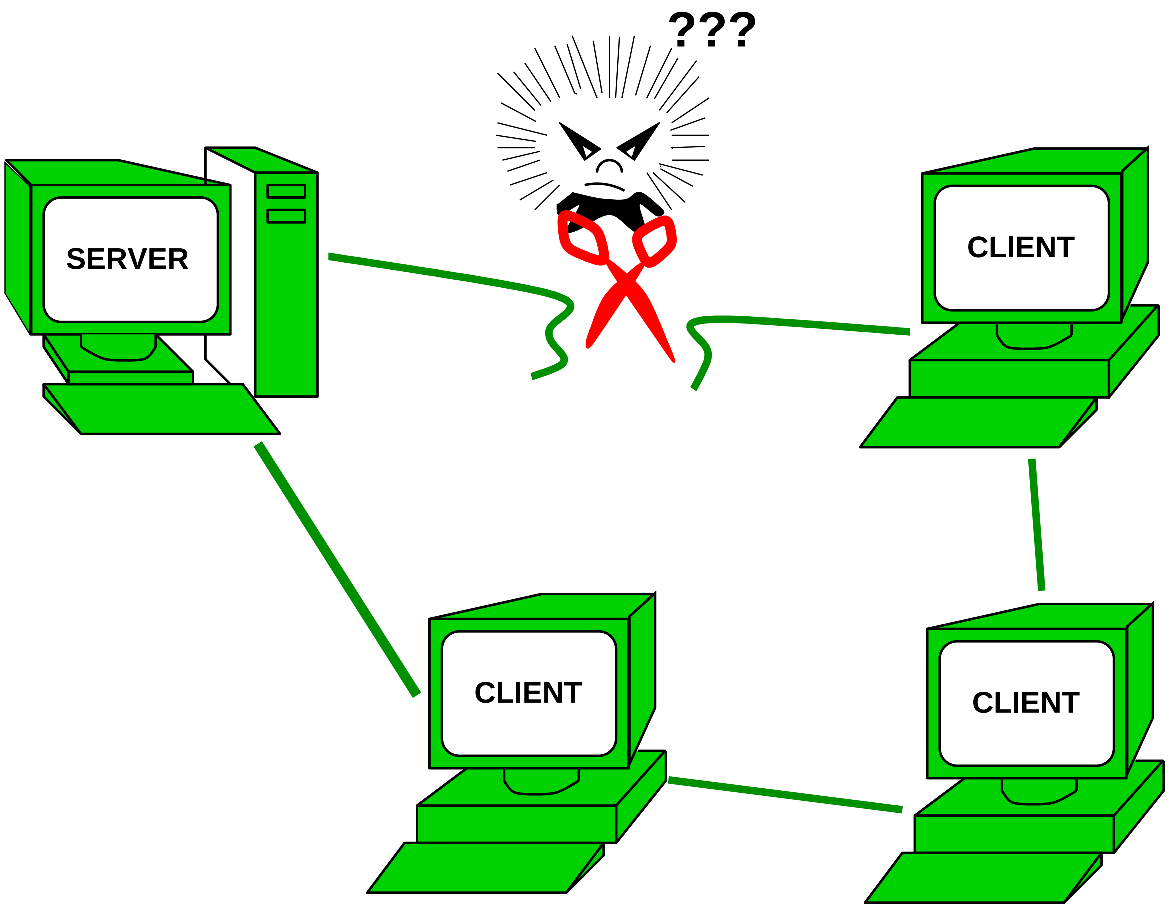


- A badly designed network:
  an attack with a small fixed cost can create a lot of damage.



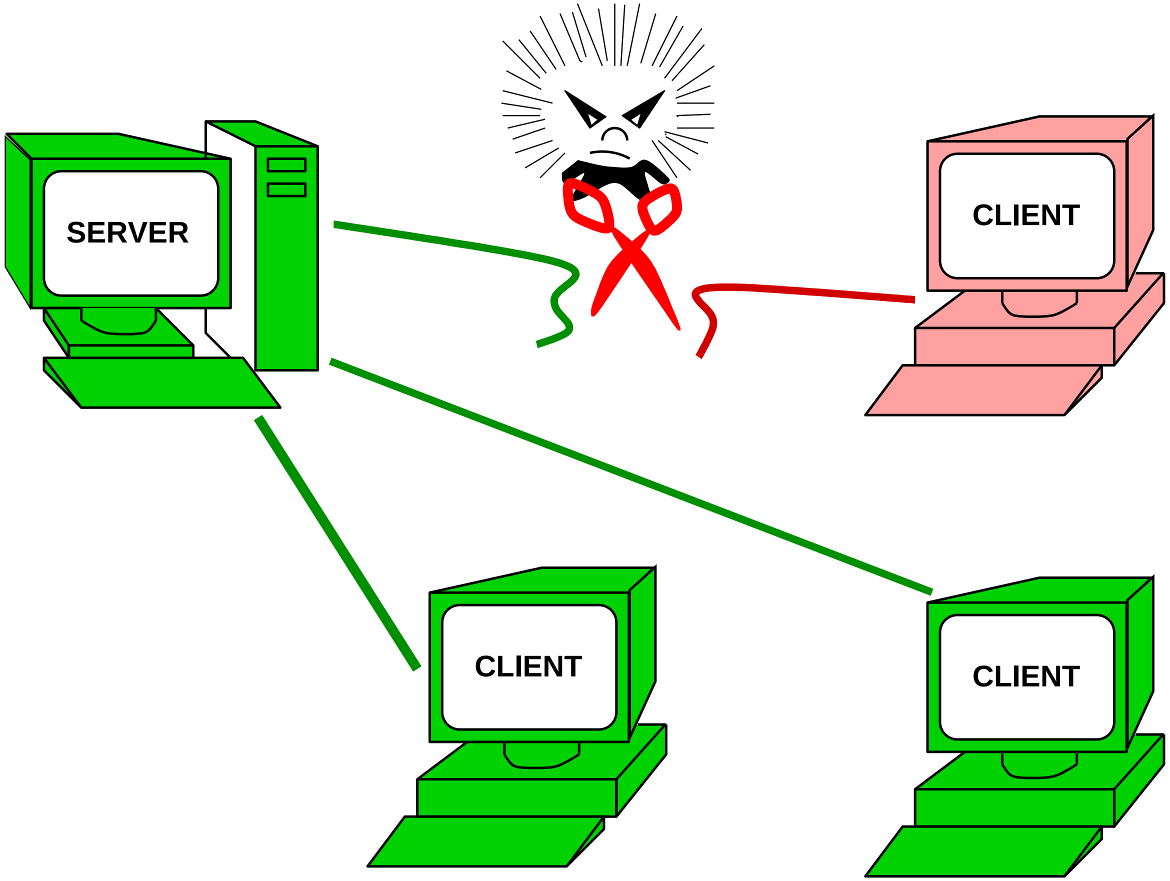
links cut	clients disconnected
1	3

- Added robustness with a redundant link:
  the cost of the attack is higher but still constant.



links cut	clients disconnected
1	0
2	3

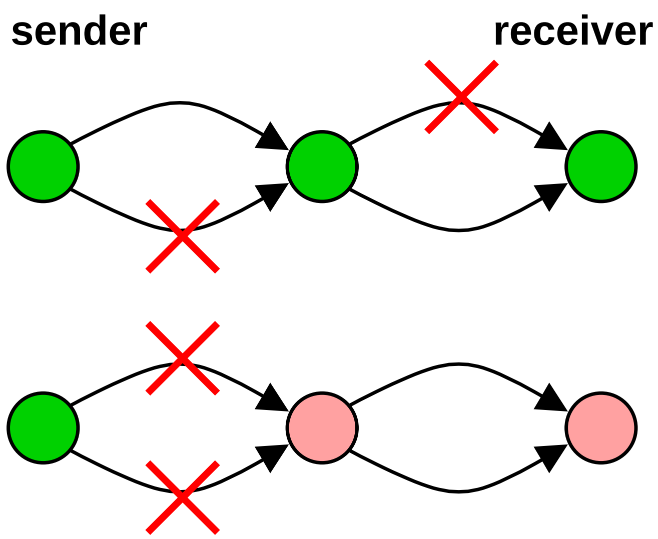
- Even more robust, with only 3 links:
  cost increases linearly with damage!



links cut	clients disconnected
1	1
2	2
3	3

Security vs. reliability

- **Reliability engineering** assumes components fail randomly and independently. Worst-case scenarios are usually unlikely to occur.
- **Security analysis** must expect worst-case failures.



Graph theoretical literature

Several variations of the inhibition problem have been studied in graph-theoretical literature:

- **Disconnect two given nodes from each other**
  $\longrightarrow$ **MIN CUT**, well-known polynomial time algorithms
- **Partition the network to many components,**
  maximize the number of components / the number of removed links
  $\longrightarrow$ a polynomial problem (Cunningham 85)
- **Disconnect 3 or more nodes from each other**,
  or disconnect 3 or more pairs of of nodes
  $\longrightarrow$ **Multi-way cut**, NP-complete (Dahlhaus & al. 92)
- **Minimize total capacity between two given nodes**
  by reducing link capacities at linear cost
  $\longrightarrow$ NP-complete (Phillips 93)

Single-server network inhibition

- We study a **new variation** of the inhibition attack:
  Attacker tries to disconnect many clients from a single server.
- Each link (and node) has a certain cost of destroying or disabling.
  The attacker has a limited budget and cannot break all links (or nodes).
- The connection to each node has a certain value for the server.
  This is the damage if the node becomes disconnected from the server.
- The attacker will use its budget optimally to break connections to clients with maximal value.

Why is this an interesting problem?

- Expensive or impossible:
  - Guaranteeing the availability of a service under all attacks.
  - Protecting the connection between one client and one server.
  - Protecting all connections in the network.
- More realistic goals:
  - Increase the cost of attack and reduce the damage.
  - Improve the availability of one service to a large number of users.
- Example:
  A determined attacker can always prevent me from connecting to Yahoo and to my bank. It is much more serious if the attacker can prevent a large number of clients from accessing the same services.
  However, the bank will not want to invest in improving the overall robustness of the Internet but may improve its own network.

Formal model

- **Network is a (directed) graph.** (Communications links are arcs.)
  One node is called the **source**.
  Each link l has a **cost** of disabling $c(l)$.
  Each node n has a **damage value** $d(n)$.
- **Attack is a set of links (and nodes)** that will be removed.
  Cost of attack is the total cost of removed elements.
- A node is **connected** if there is a path from the source to the node, otherwise **disconnected**.
- **Damage** is the total damage value of disconnected nodes after attack.

Typo in Def. 2: $CN^A = \langle N \setminus A_N,\ L \setminus (A_L \cup A_L^{-1} \cup\ A_N \times N\ \cup\ N \times A_N),\ s', c', d' \rangle.$

Optimization problem

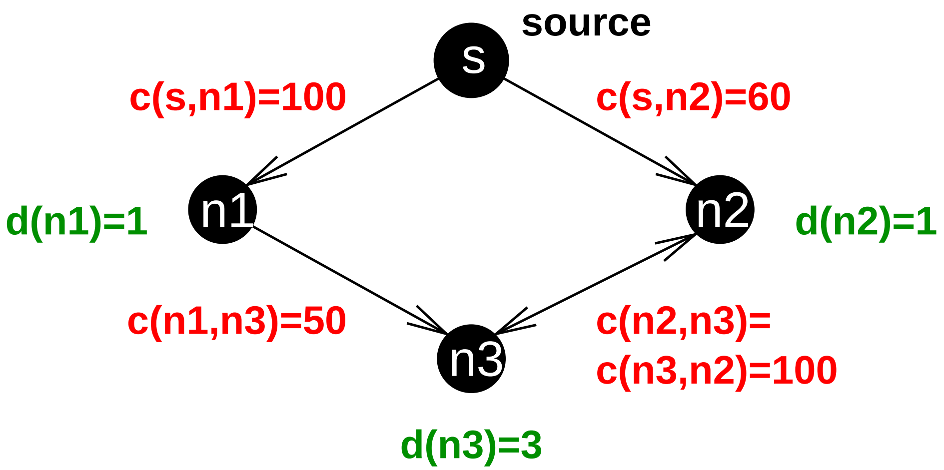
- Attackers and network designers want to know about **optimal attacks**:
  - Maximize the total damage over all attacks
    for which the cost of attack $\leq C$.
  - Minimize the cost over all attacks
    for which the total damage $\geq D$.

  Equivalent decision problem:
  - Does an attack exist such that cost $\leq C$ and the damage $\geq D$?

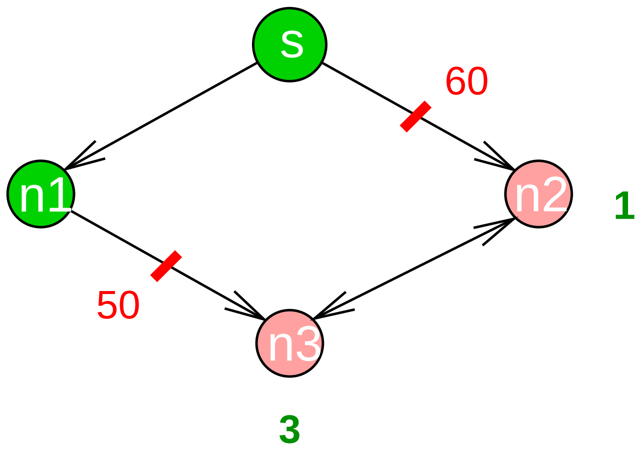
Typo in Problems 4-6: $A \subseteq N \cup L$.

A small example

(Nodes cannot break. Only one two-directional link.)

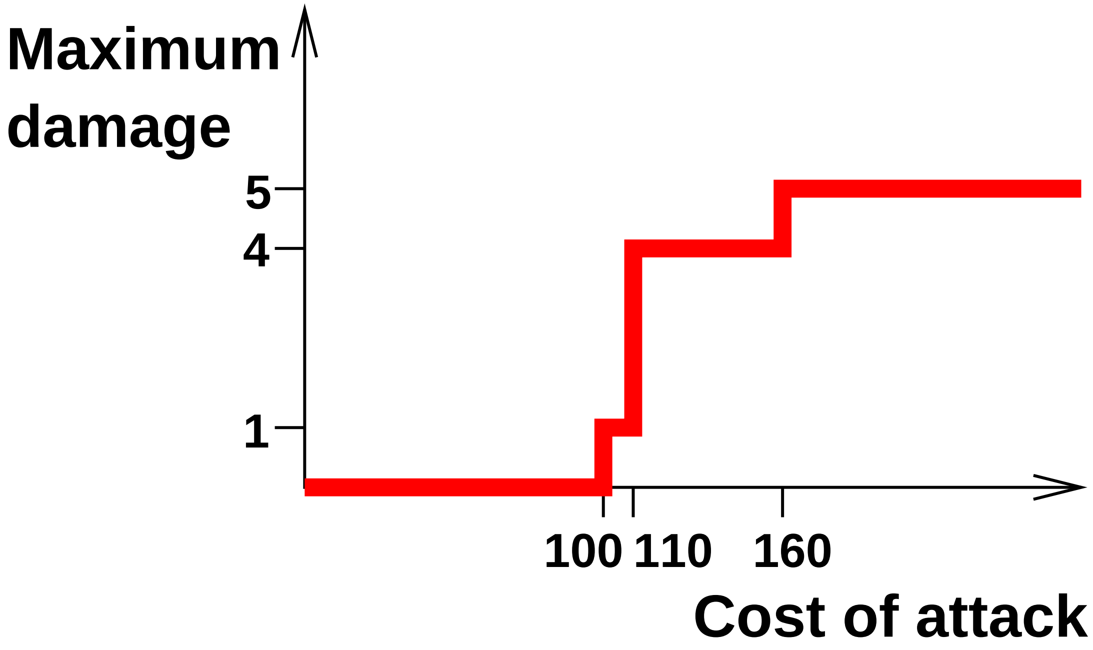


- An attack: $A = \{\langle s, n2\rangle, \langle n1, n3\rangle\}$. Cost = 110. Damage = 4.



Goal: cost-damage curve for optimal attacks

- A full picture of the seriousness of an attack is obtained by plotting the damage caused by optimal attacks against the attack budget.



(A monotonically increasing function.)

Graphs can be used to compare the robustness of network topologies.

Complexity of finding optimal attacks

- Finding an optimal attack for a given budget is **NP complete**.

  Proof in the paper by reduction from the **minimal graph bisection** problem.

- Open questions:

  Approximate solutions would be sufficient in practice.
  Are there efficient approximation algorithms?
  Are **planar or nearly planar graphs** easier to analyze?

Modelling networks as logic programs

- Networks can be represented as logic programs with **stable model semantics**.

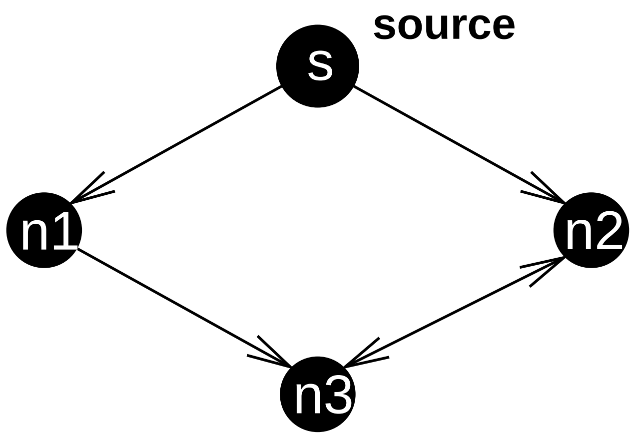
  The stable models of the program correspond to all attacks that are within the given budget.

- How to represent our problem as a declarative program:

  1. describe the network structure
  2. guess an attack and compute the connected nodes
  3. check that the attack is within the budget
  4. select the attack with the highest damage.

- Computing the connected nodes:
  If L^* is the transitive closure of the link relation,
  then a node n is connected to the source s iff sL^*n .

Network topology as a logic program




```
node(s).
node(n1).
node(n2).
node(n3).

source(s).

link(s,n1).
link(s,n2).
link(n1,n3).
link(n2,n3).
link(n3,n2).
```

Attacks

- **Every link is either broken or ok.**

```
linkOk(N,M) :- link(N,M), not linkBroken(N,M).
linkBroken(N,M) :- link(N,M), not linkOk(N,M).
```

- **Every node is either broken or ok.**

```
nodeOk(N) :- node(N), not nodeBroken(N).
nodeBroken(N) :- node(N), not nodeOk(N).
```

Connected nodes

- **The source node is always connected (unless broken).**

```
connected(N) :- source(N), not nodeBroken(N).
```

- **Connectedness spreads to ok nodes over ok links.**

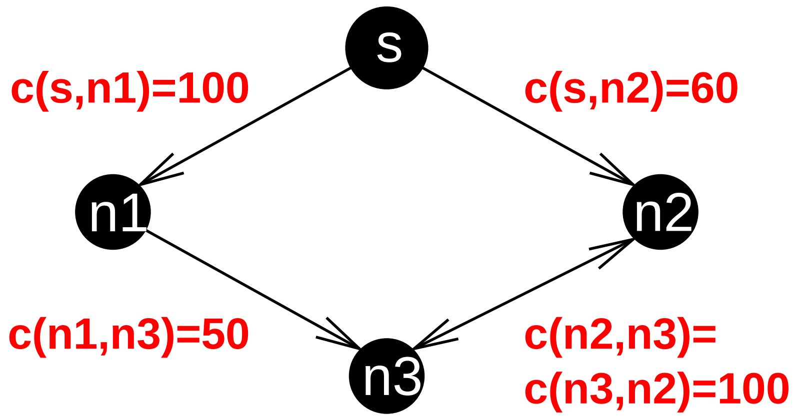
```
connected(M) :- link(N,M), connected(N),
                not nodeBroken(M),
                not linkBroken(N,M),
                not linkBroken(M,N).
```

Note: breaking a link in one direction breaks connections both ways.

Cost of breaking links

- **In this case,**
  nodes cannot break.

```
nodeOk(s).
nodeOk(n1).
nodeOk(n2).
nodeOk(n3).
```



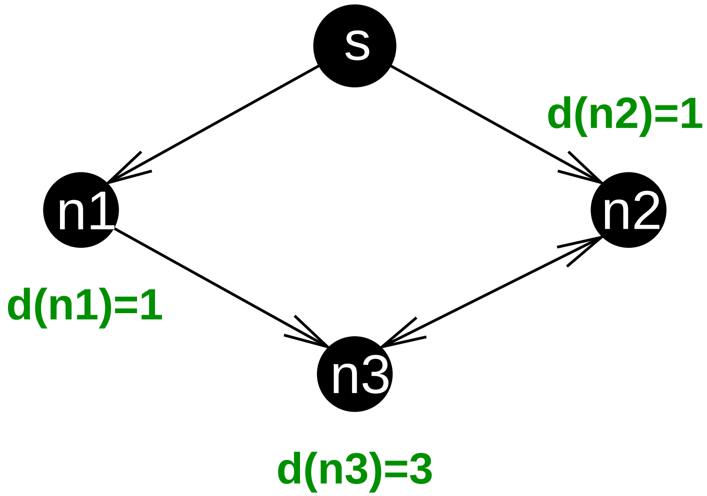
- **Attack budget is 150. Are we over the budget? (weight rule)**

```
:- 151 [ linkBroken(s,n1)  = 100,
         linkBroken(s,n2)  = 60,
         linkBroken(n1,n3) = 50,
         linkBroken(n2,n3) = 100,
         linkBroken(n3,n2) = 100 ].
```

Minimize damage

- **Find solutions with minimal total value of nodes that remain connected.**

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



- Note: The optimization and weight rules are not standard logic programs. They can be reduced to one but the resulting program would be much less efficient. The implementation must support these features.

Implementation and tests

- **Smodels**, a general-purpose logic-programming system, was used to find models for the logic programs.

- Graphs can be entered via a graphic editor and results can be viewed graphically. (student project in progress...)

- Random and fictional networks of up to 100 nodes were analyzed.
  Sparse networks with few redundant links are easier.
  (Nearly) planar and (nearly) tree-structured graphs may be easier because they have less global dependencies.

  Testing with real-world network topologies would be needed.

Results

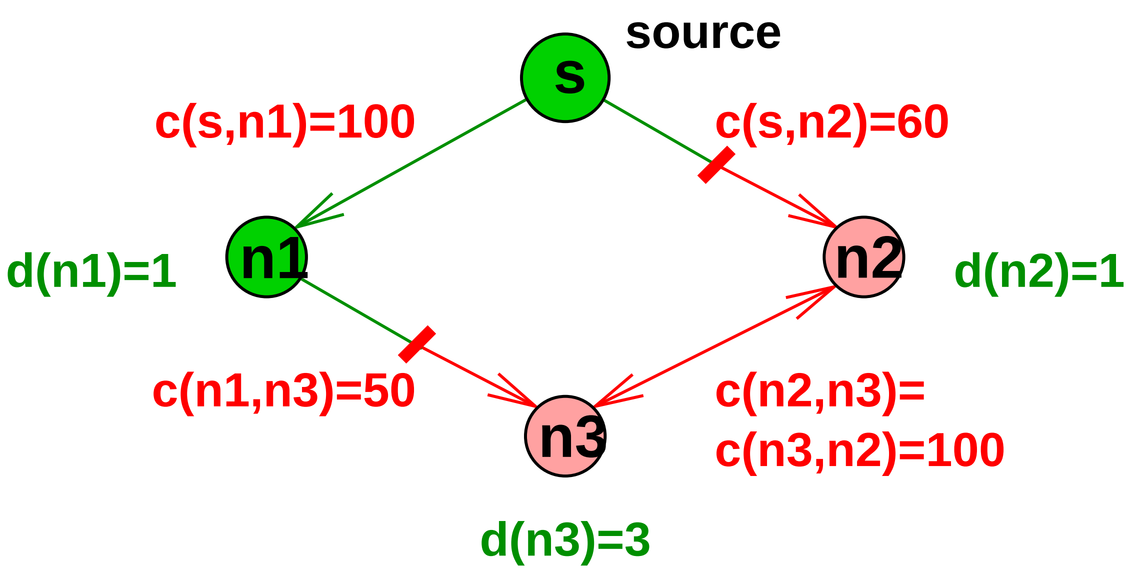
Smodels output:

```
Stable Model:  connected(s) connected(n1)
linkBroken(n1,n3) linkBroken(s,n2) linkOk(n3,n2)
linkOk(n2,n3) linkOk(s,n1) nodeOk(n3) nodeOk(n2)
nodeOk(n1) nodeOk(s) link(n3,n2) link(n2,n3)
link(n1,n3) link(s,n2) link(s,n1) source(s)
node(n3) node(n2) node(n1) node(s)
{ connected(s), connected(n1), connected(n2),
connected(n3) } min = 1
```

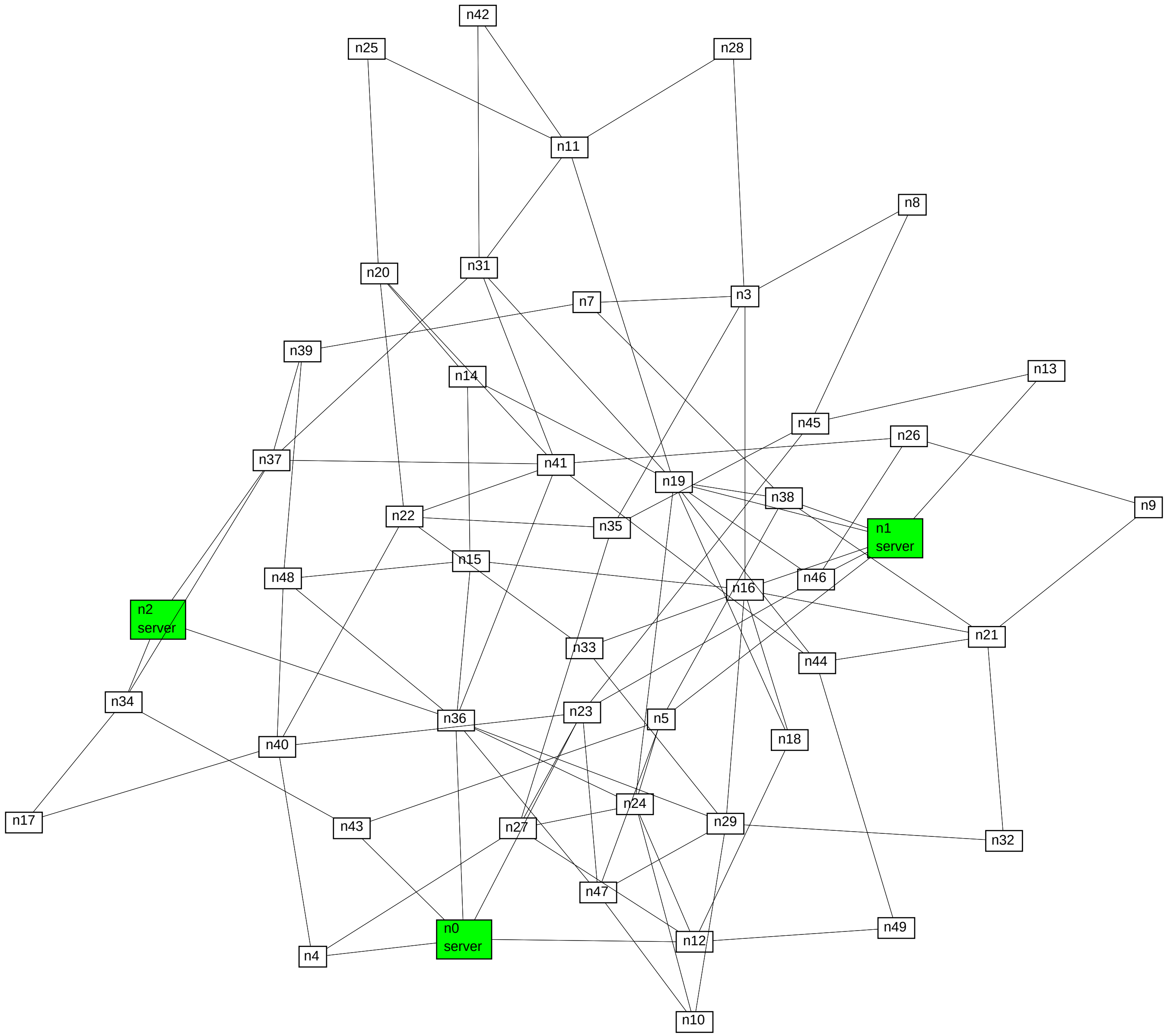
Total values of nodes that remain connected is 1.

Damage is 4.

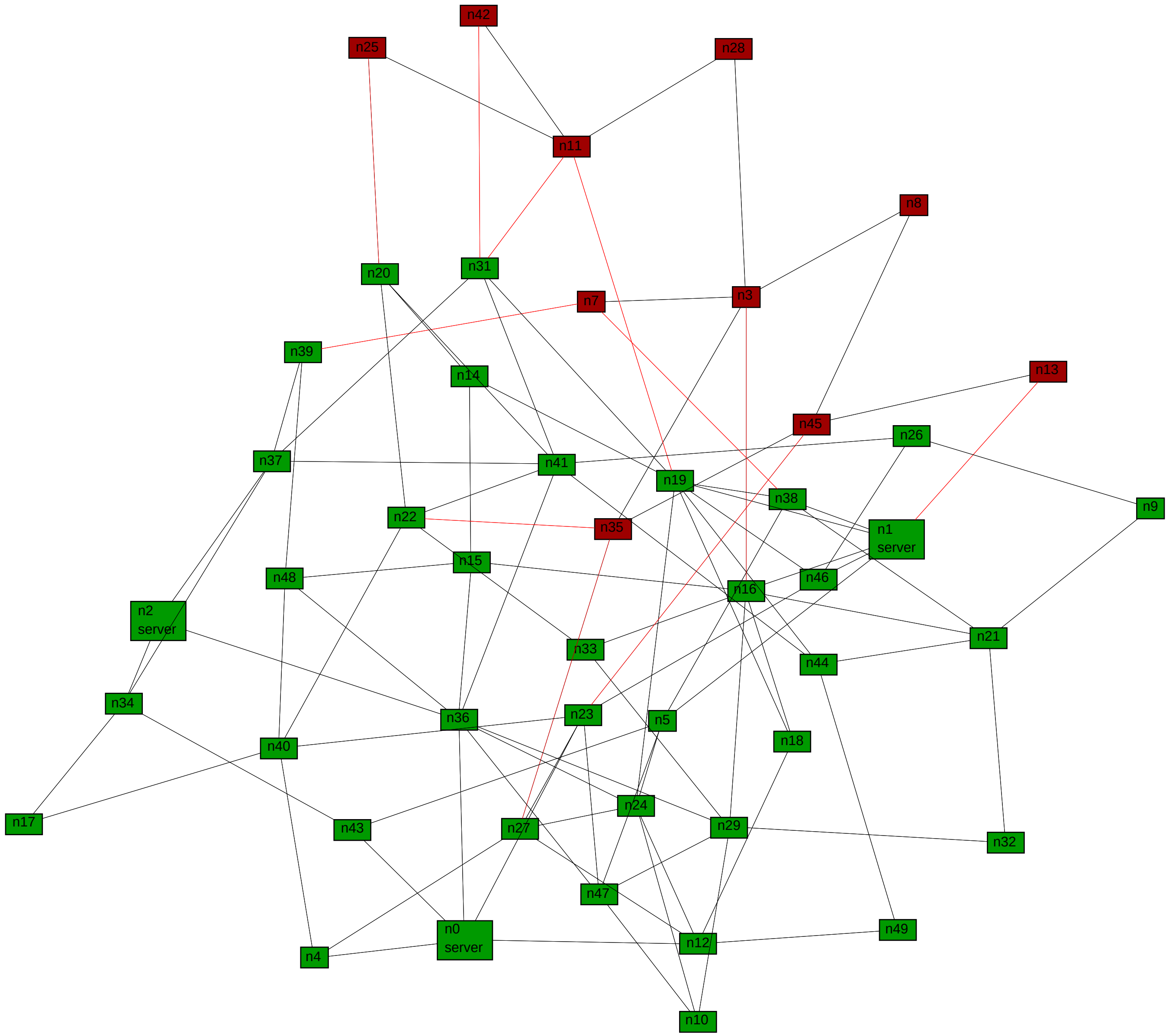
This is the attack with cost 110 seen earlier.



A random network with 48 nodes and 100 links. Budget = 11 links.



An optimal attack (budget = 11 links) disconnects 10 nodes.

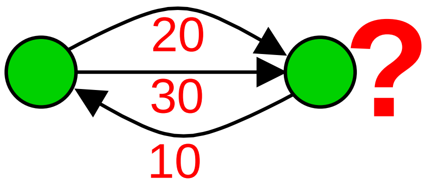


Smodels

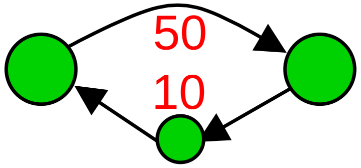
- **Smodels is an implementation of the stable model semantics for logic programs.**
- Currently one of the most efficient general-purpose solvers for constraint-satisfaction problems.
- Created by Patrik Simons and Ilkka Niemelä at HUT.
- Supports weight rules and optimize statements.
- See: `http://www.tcs.hut.fi/Software/smodels/`

Modelling: one and two-directional links

- We have chosen to represent two-directional links as two one-directional ones. It is enough to break a link in one direction. It will break in both directions without additional cost.



- Two links in the same direction: add costs together.



- Independent one-directional links: use auxiliary nodes.

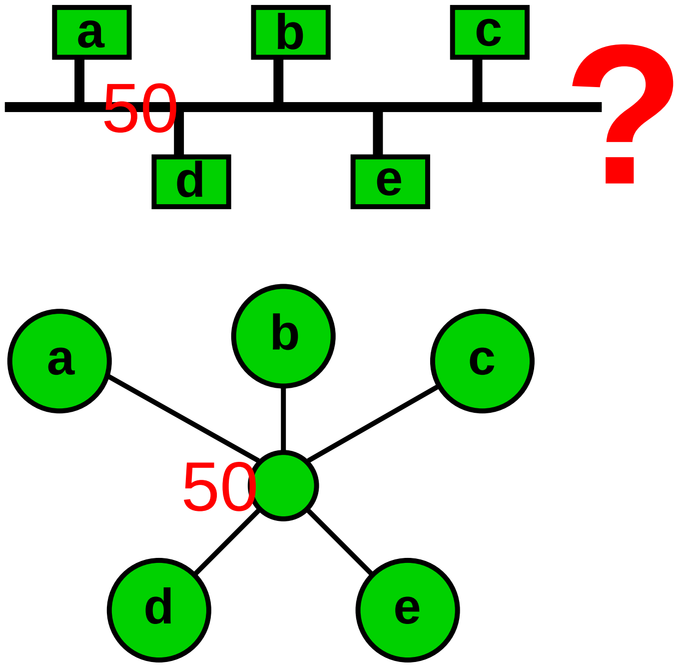
- **Power of general-purpose modelling tools and declarative programming**: Small adjustments to the model, like changes in the breaking of two-directional links, require only small changes in the logic program but may change a graph-theoretical algorithm completely.

Modelling: replicated servers

- Reliability of a service is typically achieved by maintaining several copies of the server.
  Easy to model: merge server nodes into one.
- Or, in the logic program, simply define multiple sources.

```
source(s1).
source(s2).
source(s3).
```

Modelling: broadcast LANs



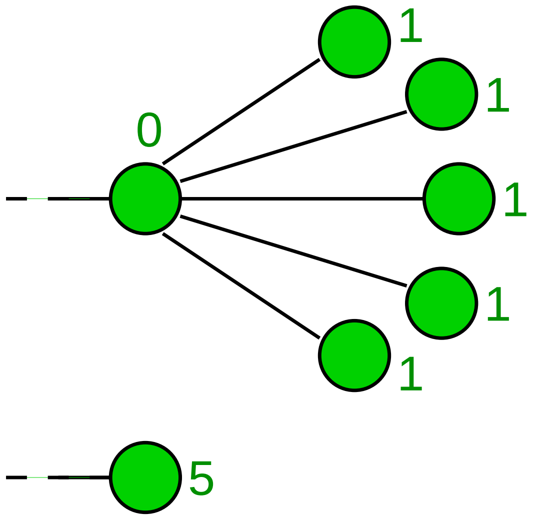
- Broadcast LANs (e.g. Ethernet) fail simultaneously for all attached nodes: use an auxiliary node for the LAN.

Modelling: optimizations

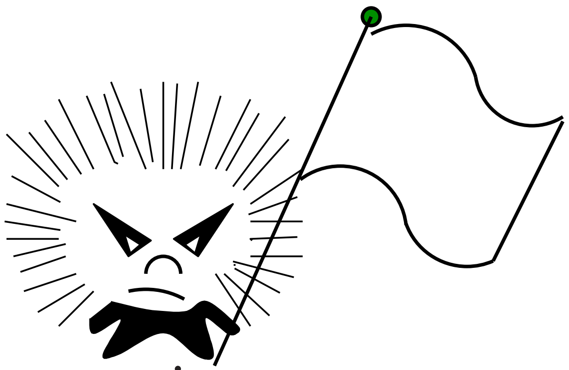
- Models can be optimized to reduce the number of nodes and links and to speed up the analysis.

  Challenge: automatic optimizations.

- Example: LANs at the edges of the network can be reduced to single nodes.



Conclusions



- We analyzed a new type of DOS attack where the enemy tries to disconnect as many clients from a single server as possible by destroying or disabling communications links.
- Finding optimal attacks is NP complete but reasonable size networks can be analyzed in practice.
- **A step towards tools for automatic analysis and design of robust network topologies.**
- **Quantitative analysis of the cost and damage of DOS attacks is needed when they cannot be completely prevented.**