NetSAT: Automated reasoning methods for verification and configuration of computer networks

Marco Gario

EMCL / NICTA

September 12, 2011

Content

Outline

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Rough idea of the problem

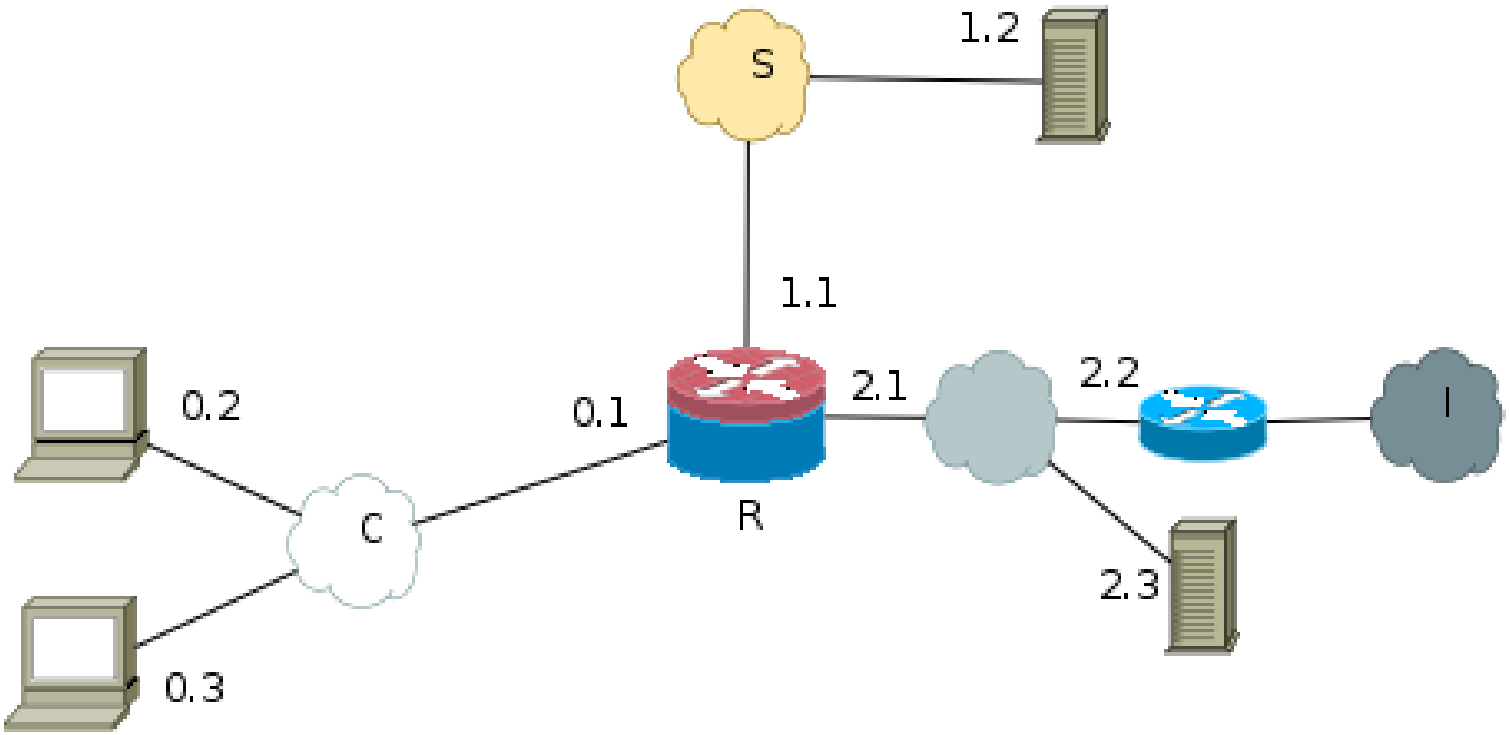
Given a computer network of which we know the configuration and the intended behaviour (*policy*), we want to check whether the current configuration "satisfies" the policy or not; if not we want to know what are the alternative configurations that can satisfy it (if any).

Eg.:

- Only the students inside the university network should be able to access the eBooks,
- Can the user A reach the service B?

Managing configurations in *complex* environments is not trivial.

Example Network



Related Work

- configAssure [1] (2008):
  - Alloy modelling language $\rightarrow$ KodKod: [2] a constraint solver for relational logic
  - Complexity in the specification of the requirements as Datalog
  - KodKod solves the problem by reducing to SAT
  - Commercial product *IPAssure* by Telecordia
- ConfigChecker [3] (2009):
  - More similar to this work
  - Extension of CTL to specify requirements
  - BDD based
  - Many modelling problem (as directionality) are not explicit in the reports

Outline

Formalization of the problem

- What level of detail do we want?
- How can we characterize network components?
- How does the policy looks like?

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router,
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host, that has an IP Address, can be Source or Destination of a connection
- Router,
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router, that is connected to multiple "networks" and can decide to which one to forward an incoming packet
- Firewall,
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router,
- Firewall, that can allow or deny the transit of a packet
- NAT

Detail Level

We need to decide how much into detail we want to go. We consider only the TCP and IP level of the TCP/IP suite. Therefore we talk mainly about *packets* of data.
We consider the following components:

- Host,
- Router,
- Firewall,
- NAT that can modify the content of a packet

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.

This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).

When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.

This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).

When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Rules

All network components are rule-based:

	Condition	Action
Routing	Destination IP	Next Hop
Firewall	Any TCP/IP field	Accept, Deny
NAT	Any TCP/IP field	Modify any TCP/IP field

Assumption: Rules are *deterministic* and are *independent* one from the other.

This structure (Condition,Action) can be used to describe the behaviour of many components in networking (eg. IPSec).

When dealing with reconfiguration we allow only the modification of the rules: we exclude, eg., topological modifications.

Network property

We define the following decision problem:

Basic Reachability Problem

Given:

- a network configuration $\mathcal{C}$
- an initial position Pos_0,
- a formula characterising a non-empty set of initial packets τ,
- a formula characterising the path *VALID*
- a final position Pos_G,
- and an integer n

Is it possible in the network $\mathcal{C}$ for all the packets p (s.t. $p \models \tau$) starting from Pos_0 to reach Pos_G in n steps (or less) satisfying the condition *VALID*?

Policy

We define also the Unreachability:

In the network $\mathcal{C}$ **no one** of the packets p (s.t. $p \models \tau$) starting from Pos_0 will reach Pos_G in n steps (or less) satisfying the condition *VALID*

A *Policy* is a collection of Network Properties (Reachability and Unreachability)

A *Policy* holds *iff* all the properties hold

Planning encoding

First solution as a Planning problem

- Intuitively we model each component as a set of actions performed on a single packet
- Is there a path/plan from A to B?
- We generated PDDL files and solved some toy examples.
- Not "suitable" if we want to verify that there's **no** path from A to B: we need a *complete* planner.

Planning encoding

First solution as a Planning problem

- Intuitively we model each component as a set of actions performed on a single packet
- Is there a path/plan from A to B?
- We generated PDDL files and solved some toy examples.
- Not "suitable" if we want to verify that there's **no** path from A to B: we need a *complete* planner.

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

SAT encoding

SAT reformulation of the verification problem:

- Planning as SAT
- SAT approach gives us a bit more flexibility
- It turns out to be very similar to Bounded Model Checking!
- We **do** have a bound!
- We have a complete method for verification

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Reconfiguration

We want to find a network $\mathcal{C}$ that satisfies the policy:

- Even preserving the topology, we still have an huge search space (eg. $\approx 2^{200}$ possible configurations for *each* network element)
- We simplify the problem assuming that we are given a set of *available* configurations.
- Encoding of the reconfiguration problem as 2QBF:
  - $\exists\forall$: $\exists$ configuration $\forall$ packets
  - Not many solvers are available. The ones tested couldn't even solve the problem with a single configuration.
- We manage to solve this problem "faster" by using an incremental SAT solver!

Complexity

We study two types of complexity:

- Packet
- Network

	Network	Packet
BRP	P (FPT)	coNP-complete
Reconf	NP-complete W[2]-hard	Σ_p^2-complete

Complexity

We study two types of complexity:

- Packet
- Network

	Network	Packet
BRP	P (FPT)	coNP-complete
Reconf	NP-complete W[2]-hard	Σ_p^2-complete

Complexity

We study two types of complexity:

- Packet
- Network

	Network	Packet
BRP	P (FPT)	coNP-complete
Reconf	NP-complete W[2]-hard	Σ_p^2-complete

Outline

Overview

We consider the information contained in the packet header:

- Src / Dest IP (32bit)
- Src / Dest Port (16bit)

Plus the Position of the packet in the network.

Overview (Cont.)

We describe the behaviour of a component with a set of formulas in the form conditions $\rightarrow$ action ($\psi \rightarrow e$):

Example: Firewall

$$\begin{aligned}
\delta_H &= (Pos_H \wedge \psi_1 \rightarrow e_*) \\
&\wedge (Pos_H \wedge \neg\psi_1 \wedge \psi_2 \rightarrow e_*) \\
&\wedge \ldots \\
&\wedge (Pos_H \textstyle\bigwedge_{i=1}^{k-1} \neg\psi_i \wedge \psi_k \rightarrow e_*) \\
&\wedge (Pos_H \textstyle\bigwedge_{i=1}^{k} \neg\psi_i \rightarrow e_{default})
\end{aligned}$$

with $e_* = \left\{ \begin{array}{ll} \neg Pos_H \wedge Pos'_{n^*} & \text{if it is an Accept rule} \\ \neg Pos_H & \text{otherwise} \end{array} \right\}$,

where Pos' indicates the value of the variable Pos at the next time-step.

Transition relation

- Build the transition relation $R(i, i+1)$ by putting together all the characteristic formulas
- $R(0, n)$ is obtained by making copies of $R(i, i+1)$
- How big should n be?
  - In IP networks we cannot have cross more than 256 hosts
  - But since the *same* packet will be processed only once by each host, we can do better than this:

    $$MP \propto (|Routers| * |NATRules|)$$

    between two hosts we will cross at most $|Routers|$. We can visit the same router twice only if the packet was modified, and there are $|NATRules|$ possible modifications.
- Note that the transition relation and the bound are independent from the properties that we want to check!

Properties

From the Basic Reachability Problem we define 5 properties that we are interested in solving: e.g.

	τ	$VALID$
P1	1 Value per field	$\top$
P5	Arbitrary over PKT_0	Arbitrary over $Visited_i$ and PKT_n

Verifying the properties means solving the following TAUT problem:

$$\phi_{BRP} = \forall.(\tau \wedge Pos_0 \wedge R(0,n) \rightarrow Pos_G \wedge VALID)$$

where the $\forall$ quantification is significant only for PKT_0 since all other variables are determined by Unit Propagation.

We can use a SAT solver to verify UNSAT of $\neg\phi_{BRP}$

Improving the SAT encoding

In a policy with k properties we need to solve k UNSAT problems.
We can use more efficiently the SAT solver by building one problem for the whole policy:

- The transition relation depends only on the configuration of the network, we can use it for testing multiple properties!
- We build a new UNSAT problem:

$$\exists Pos_0, PKT_0.R(0,n) \wedge (P_i \vee ... \vee P_k)$$

with $P_i = \tau \wedge Pos_0 \wedge \neg(Pos_n \wedge VALID)$

Reconfiguration

Can we modify the network components configuration in order to satisfy the Policy?

- Add a set of variables c_i describing which configuration are we considering,
- Each rule (condition,action) becomes

$$c_k \wedge condition \rightarrow action$$

- Rewrite the problem as:

$$\exists c_0, .., c_m. \forall Pos_0, PKT_0. R(0, n) \rightarrow \neg(P_1(c_0, ..., c_m) \vee ... \vee P_k(c_0, ..., c_m))$$

- Use incremental SAT solving to find the right configuration:

$$Policy$$
$$Policy \wedge \neg c_1$$
$$Policy \wedge \neg c_1 \wedge \neg c_2$$
$$...$$

where at each step we exclude a model obtained from the SAT solver.

Outline

Open Issues

Many interesting open issues:

- How to generate the configuration set $\mathcal{C}$?
- Scaling Tests
- Disjointness of rules
- Higher levels with more complex interactions (eg. TCP sessions)
- Rewriting as CTL problem

Questions?

References I

- Narain, S. and Levin, G. and Malik, S. and Kaul, V., Declarative infrastructure configuration synthesis and debugging, Journal of Network and Systems Management, 2008
- `http://alloy.mit.edu/kodkod/`
- Ehab Al-Shaer, Will Marrero, Adel El-Atawy and Khalid Elbadawi, Network Security Configuration in A Box: End-to-End Security
- Goultiaeva, Alexandra and Bacchus, Fahiem, Exploiting Circuit Representations in QBF Solving, LNCS-6175 Theory and Applications of Satisfiability Testing - SAT 2010, 2010.