Enforcing Secure and Robust Routing with Declarative Policies

Palanivel Kodeswaran, Wenjia Li, Anupam Joshi and Tim Finin
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250
{palanik1, wenjia1, joshi, finin]@cs.umbc.edu

Filip Perich
Shared Spectrum Company
Vienna, Virginia 22182
fperich@sharedspectrum.com

***Abstract*—Internet routers must adhere to many polices governing the selection of paths that meet potentially complex constraints on length, security, symmetry and organizational preferences. Many routing problems are caused by their misconfiguration, usually due to a combination of human errors and the lack of a high-level formal language for specifying routing policies that can be used to generate router configurations. We describe an approach that obviates many problems by using a declarative language for specifying network-wide routing policies to automatically configure routers and also inform software agents that can diagnose and correct networking problems. Our policy language is grounded in ontologies encoded in the Semantic Web language OWL, supporting machine understanding and interoperability. Polices expressed in it can be automatically compiled into low-level router configurations and intelligent agents can reason with them to diagnose and correct routing problems. We have prototyped the approach and evaluated the results both in a simulator and on a small physical network. Our results show that the framework performs well on a number of use cases, including checking for policy coherence, preventing asymmetric routing patterns, applying organizational preferences, and diagnosing and correcting failures.**

Keywords- declarative network management. policy, agent based systems

I. INTRODUCTION

BGP is the de-facto routing protocol used in the Internet today for advertising network reachability. Unlike other routing protocols, BGP is a policy based routing protocol that allows operators to control which routes are chosen in the routing protocol. This flexibility provides operators with the ability to tweak BGP to enforce the high level goals of their organizations. However, configuring BGP routers correctly to enforce organizational goals is a formidable task.

The lack of a high level language for modeling and enforcing network wide routing policies forces operators to manually configure BGP routers at the lowest level of detail. The resulting configurations have no usable semantics associated with them and consequently cannot be verified for correctness. Furthermore, the configurations do not always reflect the organization's high-level goals. Most configuration files run into hundreds of lines, further making debugging harder. In fact, recent studies [2] have shown that human error is a major cause for a number of BGP related routing failures. This problem becomes much more severe in the case of military networks where there is a lack of skilled network operators on the field. Furthermore, given the dynamic nature of the environment, an organization's routing goals can change quickly, requiring a corresponding rapid change in the BGP configurations. Expecting operators on the field to manually configure the routers quickly and correctly is unrealistic for medium to large sized networks. Even away from the field, military networks such as NIPRNet sometimes make policy decisions due to lack of skilled manpower and configuration issues that lead to vulnerabilities, as we discuss later [3].

There is a growing need for a high-level language to model and configure BGP routing policies. The goal of such a high-level language is to allow operators focus on the policy decisions rather than on the low-level implementation details. For example, operators would like to automatically configure routers to implement existing trust relationships between autonomous systems (ASs) by merely stating the relationship type such as customer-provider between them without having to manually construct the associated export and import filters. These high-level languages are particularly important in dynamic environments where operators neither have the time nor the skills to manually configure routers.

In this paper we describe an ontology-based declarative language for modeling and configuring BGP routing policies. Our language is based on Semantic Web languages such as the Web Ontology Language (OWL) [1] that have well defined syntax and logical semantics. Consequently, policies expressed in our language can be formally reasoned over, conflicts and vulnerabilities discovered and corrections made. Policies expressed in our language are automatically compiled into appropriate BGP configurations by our framework. We build on this declarative framework and construct an argumentation protocol in which neighboring routers can argue with each other to diagnose and recover from router misconfigurations.

We differ from previous tomography based "black-box" approaches such as NetDiagnoser [7] by employing a "white-box" approach that provides visibility into operator policies expressed in our language. In addition to locating the cause of failures, we also try to recover from the failure by router reconfigurations if so allowed by policy. We fathom scenarios where operator policies may prevent router reconfigurations

due to privacy and security concerns. In these cases, we alert the network operator providing them with the location and cause of the failure, thereby saving precious time in diagnosing network failures.

We see two main contributions in this work: (1) defining an ontology-based, declarative language for expressing BGP routing policies and (2) using this to solve some key problems in the military networking context. The rest of the paper is organized as follows. In section II, we review related work. We describe our declarative language and argumentation protocol in sections III and IV, respectively. In section V, we present our system architecture, and describe example scenarios in section VI. In section VII, we discuss issues related to the practical deployment of our approach and finally conclude in section VIII.

II. Related Work

There has been a great deal of work recently in the development of declarative languages for network management which space does not permit us to exhaustive survey. In [4], Hinrichs et al. propose a generic declarative network management framework that can be used for configuring many pieces of network management such as ACLs, NATs, QoS etc. at a single place. Their policy engine applies policy decisions at the granularity of individual unidirectional flows. Further the language also has support for conflict detection and policy prioritization for conflict resolution. Performance results from actual deployments showed that the policy based flow management framework typically performed well in these environments and promised good scalability.

Voellmy et al. in [5] propose Nettle, a domain specific embedded language in Haskell for BGP configuration. They use the type safety checks of the language to ensure that common configuration errors are avoided. RPSL [6] is used by ISPs to express their routing policies that are stored in Internet Routing Registries. However, none of these languages have implicit support for reasoning. Wang et al. propose a unified framework for the specification, design, implementation and verification of networking protocols in a logic based framework in [7]. The authors describe a framework in which logical statements describing the high level properties of network protocols can be specified and verified for correctness. Further, these logical statements can be automatically translated into NDlog programs for distributed execution through a property preserving transformation. The framework also supports inferring logical statements directly from NDlog programs. While the above framework deals with the verification of network protocol behavior in general, our work focuses on automatically creating BGP router configurations from high level policies.

III. Ontological Framework for Routing Policies

In this section, we present our ontology based declarative language for expressing network-wide routing policies. There are several advantages to developing an ontology-based language for routing policies for a distributed enterprise setting such as the military. The language is generic and can be used to express the policies of different organizations and sub-organizations. The ontological approach enables reusing existing ontologies such as those developed in [19] as well as naturally supporting evolution. New policies can be modeled as they become relevant to the organization by updating the appropriate ontologies. Furthermore, the logical basis of the language automatically supports reasoning and conflict resolution among policies. This becomes critical when policies are created at different sites and hierarchical levels of the organization and could potentially conflict with each other.

As part of this work, we have developed an ontology using OWL [1] and RDF [11] for representing routing policies. Our ontology models BGP routing concepts such as neighbors, autonomous systems, and network prefix. We also model the different types of policies and policy rules as one of permissive, obligatory and prohibitive. These policies typically influence whether route updates are accepted, shared or denied. We support policies that represent preferences, e.g., prefering routes from one AS over another depending on their relationship. The relationship itself could be based on multiple factors such as economical and political [12].

Our framework also supports prioritization of policies which becomes useful in the context of resolving conflict among multiple policies. The policies expressed in our language are automatically translated into appropriate BGP level router configurations. Our framework handles typical BGP scenarios seen in practice that include specifying neighbors, preferring one AS over the other as well as configuring export and import filters.

For the rest of the paper, we focus on aspects of the framework that are used for configuring import and export filters. In this discussion, we define a policy as a rule that specifies how to handle a route update. Since we limit our discussion to import/export policies, the set of allowable actions include accepting/rejecting a route update from a neighbor and sharing/denying a route advertisement to a neighbor. For example, the valley free policy [20] that requires a provider to announce its customer's prefixes to its upstream provider and peers can be expressed as follows using the familiar Prolog rule syntax.

```
shareUpdate(Update, Node, Neighbor) :-
  origin(Update, Source),
  customer(Source, Node),
  provider_or_peer(Neighbor, Node).
```

The above rule specifies that *Node* shares *Update* with its *Neighbor* if the update originated at a Customer and *Neighbor* is a peer or provider with respect to *Node*. Our framework translates the above policy into the following BGP configuration fragment

```
Router bgp ASN(Node)
Neighbor IPAddrOfRouter(Customer) filter-list 1 in
ip as-path access-list 1 permit ^ASN(Customer)$
```

where ASN(X) represents the Autonomous System Number of AS X. Any route that is not explicitly shared is not advertised to a neighbor.

A. *Creating a Routing Knowledge Base*

The knowledge base contains facts, rules and any piece of information that is useful in deciding the routing policy of the

organization such as the operating policy, traffic matrix, time of the day etc. Minimally, the knowledge base contains a high level representation of the network wide routing policies of the organization. The knowledge base is initially loaded with the following minimal set of base rules that specify the operating policy of the node, conditions for sharing/denying route updates as well as for retracting policies.

```
(?Node follow ?Policy) :-
  (?Node trust ?Authority),
  (?Authority issues ?Policy)                (Rule 1)
```

Rule 1 states that a node follows a policy issued by a trusted entity. Nodes assert that they trust their parent organization. Additionally, nodes could be configured to trust external organizations as well.

```
(?Node shareRouteAdvt ?Address) :-
  (?Node follow ?Policy),
  (?Policy sharesRouteAdvt ?Address)         (Rule 2)
```

Rule 2 asserts that a node shares a route advertisement for an address block with a neighbor as long as it follows a policy that permits sharing the advertisement.

```
(?Node denyRouteAdvt ?Address) :-
  (?Node follow ?Policy),
  (?Policy deniesRouteAdvt ?Address)         (Rule 3)
```

Rule 3 is similar to Rule 2 for denying route advertisements to neighbors.

```
(?Authority retracts ?OldPolicy) :-
  (?Authority issues ?NewPolicy),
  (?Authority issues ?OldPolicy),
  (?NewPolicy replaces ?OldPolicy)           (Rule 4)
```

Rule 4 specifies that an authority can retract a policy by issuing a replacement policy. Retractions typically occur when a higher priority policy replaces a lower priority one as well as when a later version of a policy replaces the older one. When a new policy is created, the rules representing the policy are asserted into the knowledge base along with the asserted and inferred facts. For example, the valley free policy that denies sharing non-originating prefixes to a provider makes the following assertion into the knowledge base of AS C

```
"ValleyFreePolicy deniesRouteAdvt 12.1/16"
```

where 12.1/16 corresponds to a prefix not originating in AS "C". We can thus automatically create the knowledge base from the policies expressed in our language. The knowledge-base thus created is used in the argumentation protocol described in the next section.

IV. ARGUMENTATION FRAMEWORK

We provide a high level overview of the argumentation protocol [21] used for diagnosing and recovering from router misconfigurations. The core idea is for the router agents to exploit the reasoning capability of our declarative policies to collaboratively reason with neighbors to diagnose routing failures and recover from them by reconfiguring routers. Our argumentation protocol is based on the FATIO [14] protocol and consists of the six message types.

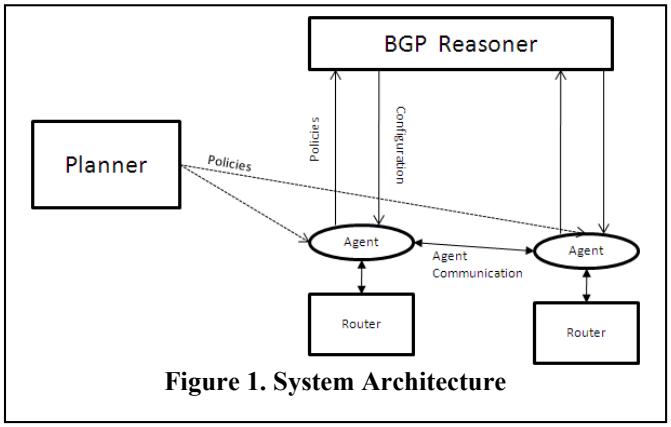

Figure 1. System Architecture

The *Ask* message is used by the initiating agent to query its neighbor for a route to a destination prefix. The recipient of Ask responds with either a *Confirm* or *Deny,* depending on whether the query evaluates to true or false. On receiving a *Confirm* or *Deny,* the initiating agent can challenge the neighbor's response with a *Challenge* message if it does not agree with the neighbor. Following this, the neighbor responds with a *Justify* message containing a partial proof tree [15] of the neighbor's evaluation of the Ask query against its knowledge base. If the initiating agent does not agree with the neighbor's query evaluation, it responds with an *Assert* message. This message contains assertions that the initiating agent believes in that invalidate the neighbor's evaluation. Such situations arise for example when the neighbor is following an older version of a policy or when one policy replaces another. The neighbor now evaluates the assertions and if they are acceptable, reconfigures the router according to the new policy and responds with a *Confirm* message. Note that the argumentation may not always converge, in which case the network operator is alerted along with a log of the argumentation executed so far.

V. SYSTEM ARCHITECTURE

In this section, we describe our system architecture, wh.ich includes routers and their agents and a common reasoner and planner, as shown in Figure 1.

Planner: The planner is the front end used by the network administrator to create network wide routing policies. Typically, this is a graphical user interface with support to view network status such as topology and traffic load on links. The planner created policies are then distributed to router agents in the network. The GUI based planner typically allows the operator on the field to use a policy wizard to create high level routing policies rather than dealing with low level configuration details. Similarly, when the routing goals of the organization change over time, the operator can use the planner to view the current operating policies and edit them to reflect changing needs. In addition to generating a RDF representation of the policies, our planner also generates a human understandable representation of the policies to aid in understanding and debugging of policies

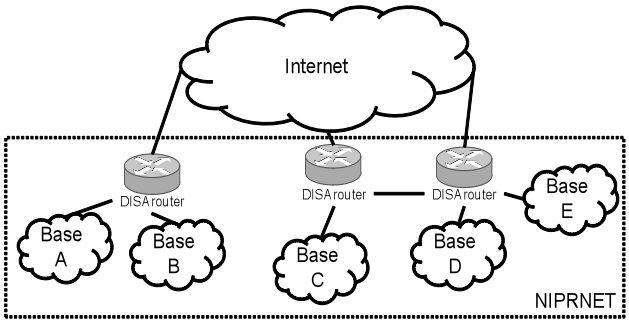

Figure 2. The DoD uses DISA maintained routers to provide connectivity within the Non-secure Internet Protocol Router Network (NIPRNET) and to the external Internet

Agent Framework: Each router is bound to an agent that is responsible for controlling and configuring the underlying physical router through a query and control interface. Policies created by the network operator are sent to the agents which are responsible for configuring the routers in line with the policies. Typically, the agents have an onboard reasoner or invoke a centralized reasoner to transform the policies into appropriate configurations. Agents communicate with each other during argumentation using an out of band communication channel such as an overlay network.

Reasoner: The Reasoner generates BGP configurations from the specified policies. Typically, each agent has its own reasoner, although a centralized reasoner for all agents seems possible for small to medium sized networks. The configurations generated by the reasoner are sent back to the calling agent, which then applies it to the underlying router.

Router: The physical router which performs packet processing and is controlled by the router agent.

We have implemented a hierarchical policy structure in our system in which operators can write policies for different sites at various organizational levels. Each agent can have its own local set of policies as well. Additionally, an agent can make local assertions about its neighbors including configuration data like IP addresses, AS Number and relationship type. The network wide policies are then merged with local policies at the node to form a unified policy which is used for generating the configurations. The ontological basis of our framework enables merging of policies as well as detecting conflicts. In our framework, we assume that network wide policies have higher priority compared to local policies during conflict resolution.

VI. USE CASES

In this section, we demonstrate the utility of our framework through a set of use cases. These use cases show how our framework eases the configuration of routers for a variety of military scenarios. The first two use cases deal with automatically choosing routes and configuring export/import filters based on the political relationship between a pair of ASs. The third use case deals with asymmetric routing that makes deploying firewalls for security in the enterprise infeasible. We propose a route injection coupled with selective advertising based solution to alleviate this problem and show how our framework can be used to automatically inject the routes as well as configure export filters. The fourth use case demonstrates our argumentation protocol for diagnosing packet dropping.

We have implemented our framework on a small router testbed of Cisco 1811 Series [17] routers. Our agent infrastructure is written in Java and uses a directory-based service for agent communication. During set up, each agent is bound to a router and communication is established with the router through telnet. We use the open source Jess [18] rule engine for generating BGP configurations from policies specified in our language.

A. *Routing Preferences*

ASs typically prefer routes from certain neighbors over others. In the military scenario, this could rise from political relations such as the US preferring routes from friendly nations such as NATO partners compared to non-NATO ASs. This preference is generally achieved in BGP by appropriately setting the local preference of routes obtained from neighbors. The higher the value, the more likely the route is to be chosen. In our framework, operators can write an obligatory policy using the abstraction of a trust relationship to prefer routes from one AS over the other. Furthermore, the local knowledge base contains the parent organizations of the neighbors and can thereby infer the trust associated with a neighbor. For example, in the case of US routers preferring routes from a neighboring NATO AS (AS 200), our framework generates the following configuration (assuming a default local preference of 100)

```
routerbgp 100
neighbor 120.120.10.2 remote-as 200
neighbor 120.120.10.2 route-map inlocal in
ip as-path access-list 1 permit ^200.*
route-map inlocal permit 10
match as-path 1
set local preference 200
```

B. *Export/Import Filter Configuration*

Similar to route preferences, ASs might want to enforce security by selectively sharing route advertisements with their neighbors. For example, US routers might want to share private networks with only other US routers. Similarly, they may not want to share certain other routes with non-NATO routers. Using our framework, operators can write a prohibitive policy that configures an export filter to deny sharing route advertisements to a neighbor based on the relationship type. As in the earlier case, the local knowledge base contains enough information about each neighbor to infer the relationship type. For example, using our framework, an operator can write a policy that denies sharing a private network 12.1/16 with non-US routers. This results in the following configuration for the neighbor 120.120.10.2 belonging to a non US organization

```
Router bgp 100
neighbor 120.120.10.2 remote-as 200
neighbor 120.120.10.2 distribute-list 1 out
access-list 1 deny 12.1.0.0 0.0.255.255
```

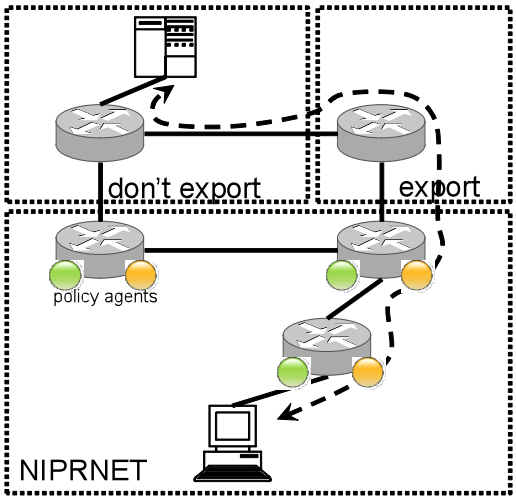

Figure 3. Policy automated framework for controlling ingress flow

C. *Asymmetric Routing*

A common characteristic observed in the Internet is asymmetric routing. As a result, packets between the same end hosts traverse different paths along the forward and backward directions. In particular for the military, this poses significant security vulnerability as follows. The U.S. Defense Information Systems Agency (DISA) is responsible for maintaining the Non-secure Internet Protocol Router Network (NIPRNET) for exchanging information freely between departments, services, bases, posts, and ships throughout the entire world.

To accomplish the task, DISA has established a set of geographically dispersed border routers providing connectivity within the large number of small military networks throughout the world. Additionally, these border routers connect the military networks to the external Internet as shown in Figure 2., making it vulnerable to security attacks. In order to limit and respond to security attacks, the military would like to employ firewalls for controlling ingress and egress traffic. The deployment of firewalls, however, is limited due to the asymmetric nature of the Internet which makes validating that incoming responses are actually requested since requests and responses could use different egress and ingress routers.

A typical routing based solution to alleviate this enterprise security problem involves selective prefix advertisements with the goal of ensuring that all ingress and egress traffic for a end host pass through the same router. Consider the scenario shown in Figure 3. Assume the client belongs to 10.1/16 and the following BGP configurations for the left and right routers.

Left Router

```
Router bgp 100
neighbor 120.120.10.2 remote-as 200
neighbor 120.120.10.2 distribute-list 1 out
access-list 1 deny 10.1.0.0 0.0.255.255
```

Right Router

```
Router bgp 100
Neighbor 120.120.10.4 remote-as 200
Network 10.1.0.0
```

These configurations ensure that only the right router shares advertisements for 10.1/16 with its neighbors, and therefore all traffic destined for 10.1/16 enters the network only through the right router. Similarly, the network is configured to route all egress traffic from 10.1/16 through the right router. This enables deploying a firewall to verify that all inbound traffic is in fact requested. Manually configuring each router to enforce this solution for large networks such as NIRPNET seems impractical. On the other hand, our policy based framework can
be used to automate the configuration process for large scale networks through a single DoD policy that states

- "A router shall route incoming Internet originated traffic only for prefixes that use it for outgoing Internet-destined traffic "

This policy is distributed to all routers in NIRPNET. Additionally, each border router asserts the IP address block that it manages in its local knowledge base. With these two pieces of information, our policy framework is able to create appropriate BGP configurations to enforce the above described security solution. In this automated solution, an agent is responsible for periodically evaluating the state of the BGP and internal routing protocols. Whenever the agent discovers that a path has been added for an internal prefix, the agent can query the managing agent for the internal prefix to find out whether or not this router is on its preferred path to the Internet. If the router is on the path, then the agent allows the prefix to be exported to Internet peers. On the other hand, if the router is not on the path, then the agent configures a BGP export filter to block the export of that prefix to its peers.

D. *Traffic Dropping Diagnosis*

We consider the following scenario where each router is loaded with one of the following two policies

- "DropPolicy" drops traffic destined for 12.1/16
- "AllowPolicy" allows traffic destined for 12.1/16

When a router detects that its traffic is being dropped by an intermediate router, the router initiates an argumentation. The argumentation proceeds in two steps. In the first step, the router queries each upstream router to see if they are configured to drop traffic to 12.1/16. At the end of the first step, the router locates the upstream router responsible for dropping traffic. In the second step, the router argues with the dropping router following the argumentation protocol described earlier. At the end of argumentation, the dropping router either reconfigures its policy to allow traffic or alerts a human operator if a conflict rises.

VII. Deployment Issues

In this section we discuss three issues related to the practical deployment of our approach.

1) Changes to Existing Routers: We note that our approach does not require any changes to the underlying routers. The agent communication framework can be built as an overlay on top of the existing ip networks. The only requirement is for the routers to support a query and control

interface to the agent for example through telnet. Almost all routers support configuration through telnet which can then be used as the agent interface.

2) Security and Privacy: The argumentation protocol shares routing policies with neighboring domains as part of the failure diagnosis and some operartors may not prefer this sharing due to privacy concerns. Additionally, there is a reconfiguration process that also takes place as part of argumentation. We propose a two level solution to this privacy concern. At the first level, the operator could configure the argumentation to not share the policy information but merely respond to only Ask messages with a Confirm or Deny. For example, an agent could say that it doesn't export a route but not specify the policy that denies the sharing. In this case, the neighboring agent could either continue the argumentation with this limited information or alert its human operator. At the second level, the operator could configure the argumentation to continue as normal, but not perform the reconfigurations. All router reconfigurations need to either be from a list preapproved by the operator or perfromed manually .

3) Policy Conflicts: When the argumentation protocol does not converge due to policy conflicts, having a centralized policy store where the entire routing policy of the organization is stored could mitigate such scenarios. In these cases, the argumentation protocol could consult the centralized repository to resolve conflicts. Alternatively, the operator could be alerted with a log of the argumentation.

VIII. Conclusions

We have described an approach to managing network routers by using a declarative language for specifying network-wide routing policies to automatically configure routers and also inform software agents that can diagnose and correct networking problems. This can solve a variety of problems including the selection of paths that meet complex constraints on length, security, symmetry and organizational preferences as well as diagnosing and fixing routing problems caused by router misconfiguration. Our policy language is grounded in ontologies designed in the Semantic Web language OWL, supporting machine understanding and interoperability. Polices expressed in it can be automatically compiled into low-level router configurations and intelligent agents can reason with them to diagnose and correct routing problems. We have prototyped the approach and evaluated the results in both a simulator and on a small physical network. Our results show that the framework performs well on a number of use cases, including checking for policy coherence, preventing asymmetric routing patterns, applying organizational preferences, and diagnosing and correcting failures.

Acknowledgment

This work was supported by an STTR grant from the Defense Advanced Research Agency (W31P4Q-06-C-0395), the Air Force Office of Scientific Research under MURI award FA9550-08-1-0265 and the NSF under Grant Number 0910838.

References

[1] S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L.Stein. OWL Web Ontology Language Reference W3C Recommendation. Technical report, W3C, February 2004.

[2] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in Applications, Technologies, Architectures, and Protocols for Computer Communication, vol. 32, no. 4, 2002.

[3] Gibson. Personal Communication.

[4] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, "Practical declarative network management," in Proceedings of the 1st ACM workshop on Research on enterprise networking - WREN '09. New York, New York, USA: ACM Press, 2009.

[5] A. Voellmy and P. Hudak. Nettle: A language for configuring routing networks. In DSL '09: Proc. IFIP TC 2 Working Conference on Domain-Specific Languages, pages 211-235, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing policy specification language (RPSL), 1999.

[7] Formally Verifiable Networking. A. Wang, L. Jia, C. Liu, Boon T. Loo, O. Sokolsky and P. Basu. 8th Workshop on Hot Topics in Networks (ACM SIGCOMM HotNets-VIII), New York, Oct 2009

[8] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data. In Proc. 3rd ACM International Conference on emerging Networking EXperiments and Technologies, pages 1--12. ACM, 2007.

[9] Y. Huang, N. Feamster, and R. Teixeira, "Practical Issues with Using Network Tomography for Fault Diagnosis."

[10] N. Feamster, H. Balakrishnan, and J. Rexford, "Some foundational problems in Interdomain routing," in In HotNets, 2004. (Cited on, 2004, pp. 41–46.

[11] O. Lassila and R. Swick. Resource description framework (RDF) model and syntax. Technical report, W3C, February 1999.

[12] P. Kodeswaran, S. B. Kodeswaran, A. Joshi, and F. Perich, "Utilizing semantic policies for managing BGP route dissemination," in IEEE INFOCOM 2008 - IEEE Conference on Computer Communications Workshops. IEEE, 2008, pp. 1–4.

[13] P. Hunter. Pakistan YouTube block exposes fundamental Internet security weakness:: Concern that Pakistani action affected YouTube access elsewhere in world. Computer Fraud & Security, 2008(4):10--11, 2008.

[14] P. McBurney and S. Parsons, "Locutions for Argumentation in Agent Interaction Protocols," in Proc. International Conference on Autonomous Agents, 2004.

[15] A. Eriksson and A.-L. Johansson. Neat explanation of proof trees. In Proceedings of the 9th international joint conference on artificial intelligence, pages 379--381. Morgan Kaufmann Publishers Inc., 1985.

[16] B. Quoitin and S. Uhlig, "Modeling the Routing of an Autonomous System with C-BGP," 2005.

[17] "Cisco 1811 Series." , http://www.cisco.com/en/US/products/ps6187/

[18] E. Friedman-Hill. JESS in Action. Manning, 2003.

[19] Ontologies for Distributed Command and Control Messaging. Duc N. Nguyen, Joseph B. Kopena, Boon Thau Loo, and William C. Regli. 6th International Conference on Formal Ontology in Information Systems (FOIS), May 2010.

[20] L. Gao, "On inferring autonomous system relationships in the internet," IEEE/ACM Transactions on Networking (TON), vol. 9, no. 6, 2001.

[21] Palanivel Kodeswaran, Anupam Joshi, Tim Finin and Filip Perich. "A Declarative Approach for Secure and Robust Routing". In Proceedings of the 3rd ACM Workshop on Assurable and Usable Security, Chicago 2010.