Correcting Routing Failures using Declarative Policies and Argumentation

Palanivel Kodeswaran*, Anupam Joshi*, Tim Finin* and Filip Perich†

*Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore MD 21250

†Shared Spectrum Company
Vienna VA

Many Internet failures are caused by misconfigurations of the BGP routers that manage routing of traffic between domains. The problems are usually due to a combination of human errors and the lack of a high-level language for specifying routing policies that can be used to generate router configurations. We describe an implemented approach that uses a declarative language for specifying network-wide routing policies to automatically configure routers and show how it can also be used by software agents to diagnose and correct some networking problems. The language is grounded in an ontology defined in OWL and polices expressed in it are automatically compiled into low-level router configurations. A distributed collection of software agents use the high-level policies and a custom argumentation protocol to share and reason over information about routing failures, diagnose probable causes, and correct them by reconfiguring routers and/or recommending actions to human operators. We have evaluated the framework in both a simulator and on a small physical network. Our results show that the framework performs well in identifying failure causes and automatically correcting them by reconfiguring routers when permitted by the policies.

Our declarative language exploits the advantages of ontology based languages in that they naturally support evolution and can be used to model newer policies as they become relevant to the organization. Furthermore, the logical basis of the language automatically enables reasoning and conflict resolution among policies. The developed ontology models concepts such as neighbors, autonomous system, network prefix etc. We also model the different types of policies viz. permissive, obligatory and prohibitive. In this work, we focus on import/export filters and these policies typically influence whether route updates are accepted, shared or denied. Using concepts defined in our ontology, we can write policies that specify which routes are accepted from neighbors based on the relationship with the neighbor. The relationship itself could be based on a variety factors such as economical, political etc. Our framework also supports prioritization of policies that becomes useful in the context of resolving conflicts among multiple policies.

Building on the reasoning capability of our declarative policies, we propose architecture for the diagnosis and recovery of routing failures. In our architecture, router agents use the high level policies to collaboratively reason with neighbors to diagnose routing failures and recover from them by reconfiguring routers. Under cases where reconfiguration is not permitted by policy, our framework recommends appropriate actions to the human operator. Agent communication in our framework is achieved through an out of band communication channel such as an Overlay network.

Our developed argumentation protocol is based on the FATIO argumentation protocol. Our protocol consists of the following six messages that are used by router agents to argue with their neighbors for diagnosing and correcting import/export filter misconfigurations. *Ask* is used by the initiating agent to query its neighbor. In our use case, the Ask query is whether the neighbor has a route to a destination prefix. The recipient of Ask responds with either a *Confirm* or *Deny* depending on whether the query in the Ask message evaluates to true or false. In our use case, this corresponds to whether the neighbor has a route to the destination prefix or not. 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 proof tree of the neighbor's evaluation of the Ask query. 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. This ends a round of argumentation. We would like note that our protocol does not necessitate complete information disclosure for operation. The pieces of information that are exchanged in the argumentation as well as the set of allowable local reconfigurations can be dictated by the operator.

We have evaluated our framework in both a simulator and on a small physical network. Our results show that the framework performs well in diagnosing routing failures and automatically correcting them by reconfiguring routers where so permitted by operator policy.