Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2007-039 July 26, 2007

Agent Organization and Request Propagation in the Knowledge Plane

Ji Li

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

CSAIL

Agent Organization and Request Propagation in the Knowledge Plane

Ji Li
jli@mit.edu
MIT CSAIL
Cambridge, MA 02139

Abstract

In designing and building a network like the Internet, we continue to face the problems of scale and distribution. In particular, network management has become an increasingly difficult task, and network applications often need to maintain efficient connectivity graphs for various purposes. The knowledge plane was proposed as a new construct to improve network management and applications. In this proposal, I propose an application-independent mechanism to support the construction of application-specific connectivity graphs. Specifically, I propose to build a network knowledge plane and multiple sub-planes for different areas of network services. The network knowledge plane provides valuable knowledge about the Internet to the sub-planes, and each sub-plane constructs its own connectivity graph using network knowledge and knowledge in its own specific area. I focus on two key design issues: (1) a region-based architecture for agent organization; (2) knowledge dissemination and request propagation. Network management and applications benefit from the underlying network knowledge plane and sub-planes. To demonstrate the effectiveness of this mechanism, I conduct case studies in network management and security.

Index Terms

knowledge plane, connectivity graph, agent, region, knowledge dissemination, network topology

I. Introduction

In designing and building networks like the Internet, we continue to face the problems of scale and distribution. Traditionally, network analysis, diagnosis and management have been done manually by a small number of people. As the Internet continues to grow in reach and in density, we have seen increasing problems with understanding how it works, where it runs into problems and how to address those problems. With the increased scale, penetration, and distribution of the Internet, those traditional manual approaches requires an increasing number of people to be involved, and the management task itself becomes increasingly complex. Therefore, a central problem is to make the network self-knowledgeable, self-diagnosing, and perhaps in the future self-managing.

In network architecture, we recognize two architectural divisions: a data plane over which data is transported, and a control plane that manage the data plane. To make the network more intelligent, a new idea was proposed recently as *the knowledge plane* [1]. The knowledge plane (KP) was proposed as a new higher-level artifact to improve network management and network applications. At an abstract level, the knowledge plane gathers observations, constraints and assertions, and applies rules to these to generate observations and responses. At the physical level, it runs on hosts and servers within the network on which knowledge is stored. The KP is a loosely coupled distributed system of global scope. The KP brings up a number of challenging problems, such as knowledge representation and dissemination, incorporation of AI and cognitive techniques, conflict resolution, trust and security, how to design a distributed knowledge base for the networks, how to incorporate into the framework pre-existing sets of specialized data and tools for network management, etc.

A. Problem Statement

In light of the fact that the overall objective of the KP is to make the Internet or another network of that scale and scope more intelligent about itself, I focus on a particular problem in this proposal. In the Internet, network applications often need to maintain efficient connectivity graphs for various purposes. Examples include overlay networks, content distribution networks, end system multicast, peer-to-peer networks, publish/subscribe systems, etc [2], [3], [4], [5], [6]. For instance, routing overlays build their own routing graphs to route around congested paths with comparable or even better performance [2]; end-system multicast constructs the application-layer multicast tree to transmit video efficiently [3]; nodes in peer-to-peer networks probe each other to find nearby neighbors to improve lookup performance [4]. Currently each of those applications builds and maintains its own connectivity graph by probing latency, available bandwidth, loss rate between hosts actively, which often incurs significant cost in the network as redundant operations are performed by them individually. We believe that the introduction of the knowledge plane as a common infrastructure can help those applications construct more efficient connectivity graphs with greatly reduced maintenance cost. In particular, I propose to design *an application-independent mechanism at the network layer that can be used for network management and applications to build application-specific efficient connectivity graphs*. By doing this, I hope to not only reduce the maintenance cost for network management and applications, but also to make new services available.

Specifically, the core of the research in this proposal is to define, design and demonstrate a system and its supporting mechanisms that provide the KP with the ability to improve existing applications, and to organize new applications. The applications here have the following features:

1) They are widely distributed in the Internet, and formed as a set of cooperating agents, and follow the pattern of "many local agents, plus some global correlation, filtering, consistency, and/or analysis functions";
2) They involve a variety of expertise and reasoning in order to handle widely distributed, incomplete, possibly contradictory and inaccurate information;
3) There is a commonality in the structure and nature of the pattern that it is both useful and possible to abstract out.

Thus we arrive at our research objective, which is to derive a system that will provide for organizing a set of agents to achieve the functions of a KP application, adaptively to a set of constraints. To do this, we divide the subject of our research into three parts, the abstraction of the "KP applications" into organizing constraints, a set of orthogonal constraints that are external to the application design itself and imposed by the Internet, administrative organizations, physical or topological constraints, and context, as well as an underlying network knowledge plane in which information will be made available.

B. My Approach

For the purpose discussed above, I propose to build a network knowledge plane (NetKP) for the network layer and on top of it, sub-planes for network management (sub-KPs) each designed to provide one category for network management functionality. That is, the KP consists of the generic NetKP and multiple specialized sub-KPs. Both the NetKP and sub-KPs are composed of agents. The relationship of components in my work is demonstrated in Figure 1.

The NetKP provides valuable knowledge about the Internet to network management and applications in a scalable and efficient way. The knowledge in the NetKP includes network topology, and performance information (latency, bandwidth, etc). I will address problems including agent organization and network knowledge collection and dissemination.

On top of the network knowledge plane, we construct multiple sub-planes for network management. Each sub-KP focuses on one network service. For example, agents interested in the

DNS form a sub-KP about DNS that helps diagnose DNS failures. In this proposal sub-KPs are for the purpose of network management, but I expect that sub-KPs may be used for other purposes, for example, a sub-KP for music. The connectivity graphs of sub-KPs are constructed using the knowledge provided by the NetKP and knowledge in their specific areas.

Both network management and network applications can benefit from the NetKP and the sub-KPs. I will demonstrate the effectiveness of this approach by prototyping several sub-KPs and network management capabilities combining them. In the case of network applications, I improve the organization of local detectors and global detectors in intrusion detection by taking advantage of network knowledge.

As we propose that the knowledge plane is composed of agents, the key problems here are agent organization and request and knowledge propagation in the knowledge plane. By designing the NetKP and sub-KPs and conducting several case studies, I hope to improve our understanding on the knowledge plane structure in terms of agent organization and knowledge dissemination, and network management using KP, as well as to motivate future research in this area.

Note that in this proposal I adopt an incremental approach in designing the knowledge plane. In my design, agents run on dedicated servers or on end hosts, and some agents are able to access routers' status and receive BGP feeds, but agents do not control routers or run on routers. I choose this incremental approach instead of requiring upgrading routers or the whole network, so as to make it feasible to deploy the proposed knowledge plane gradually in today's Internet, and also because I believe that routers should be focus on routing instead of providing sophisticated functionality. The downside is that this design choice limits the ability of agents and thus the knowledge plane, as routers know and control network properties directly.

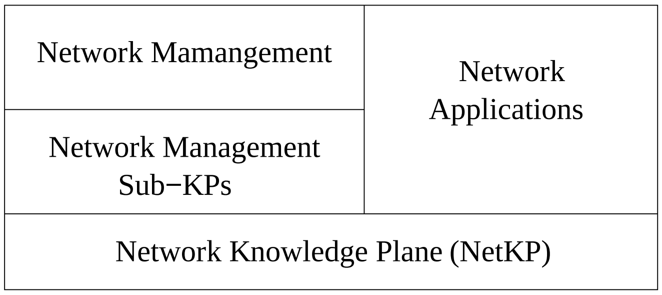

Fig. 1. The system architecture. The underlying is the NetKP that provides application-independent knowledge. On top of that, there are two parts. One is network management sub-KPs organized based on the area of interest, and network management applications are built on top of the sub-KPs. The other is network applications that take advantage of the NetKP directly.

C. Proposal Organization

The rest of the proposal is organized as follows: In Section II, I present an overview of the agent architecture that is common for both the NetKP and the sub-KPs. Then I describe the network knowledge plane in detail in Section III. Section IV discusses how to organize sub-KPs using the network knowledge and area-specific knowledge. Case studies on network management are described in Section V Section VI, and Section VII. Section VIII examines related work, including various overlay networks, publish/subscribe systems, etc. Research agenda is presented in Section IX. Section X summarizes the proposal.

II. General Design Overview

In this section, I discuss general design requirements and basic components of the knowledge plane. Note that issues addressed here apply to both the NetKP and management sub-KPs.

A. System Requirements

To build a distributed system of global scale as the knowledge plane, I believe the following features must be met:

1) *Scalability*. As the amount of knowledge about the Internet is huge and it is distributed throughout the Internet, the number of agents is large and agents are distributed all over the Internet. We need to organize the agents in a scalable fashion to better support request propagation, and knowledge collection and dissemination.
2) *Efficiency*. The KP is supposed to respond quickly to requests, and distribute knowledge to where it is needed efficiently. Furthermore, in many cases knowledge, such as the available bandwidth of a path, may become outdated very soon. Therefore, we need efficient knowledge collection mechanisms.
3) *Robustness*. The KP is proposed to facilitate network management, and agents are needed most in case of network behaving incorrectly. Therefore, the agent organization must be robust to network failures.
4) *Non-intrusiveness*. The KP needs to collect, share and distribute large amounts of knowledge, without adding too much burden to the network. Therefore, aggregation is necessary on both knowledge and requests to reduce network traffic overhead.
5) *Others*. There are other properties we should consider. We should design the KP for trustworthiness, longevity, and heterogeneity, as they are important issues. But they are not the focus of this proposal.

B. System Design

In both the NetKP and sub-KPs, there are three tasks in common: (1) knowledge plane organization; (2) request propagation and knowledge dissemination; (3) knowledge management. The first task, knowledge plane organization, refers to how agents in the KP discover each other and how agents organize themselves together. The second task, request propagation and knowledge dissemination, deals with how to propagate requests so that requests can be resolved quickly and efficiently and how to disseminate knowledge so that agents interested in that knowledge can receive it in a timely fashion. The first two tasks are tightly related to each other, because agent organization largely determines how requests and knowledge can be propagated. The third task, knowledge management, addresses the question of how agents manage their local knowledge and learned knowledge. For example, agents may maintain a distributed knowledge base. The first two tasks are the focus of this proposal.

I propose a region-based agent architecture to organize the knowledge plane. Region is the central organizing components in the KP and sub-KPs. Both the NetKP and sub-KPs consist of agents in their layers respectively. Agents are responsible for collecting, disseminating knowledge and resolving requests. Agents also manage a distributed knowledge base. The knowledge base is distributed among agents since the network knowledge is distributed and managed by different parties in the Internet.

Agents are organized into regions to achieve scalability and efficiency. The concept of the *region* is proposed as a new network feature to facilitate network organization [7]. In the NetKP, regions are often constructed following autonomous systems (ASes) or corporate networks, and agents in a large autonomous system may be divided into several regions. In sub-KPs, agents with similar interests form regions based on comprehensive criteria, as discussed later.

C. Agents

I define *agents* as entities in the Internet that participate actively in the KP including both the NetKP and sub-KPs. Agents are responsible for collecting, storing, sharing and distributing knowledge. The NetKP is composed of agents at the network layer, and sub-KPs are composed of agents in their specifi c areas of interest.

To facilitate the discussion, I also defi ne *request* and *offer*, following the defi nitions in [8]. A *request* is a message that looks for an answer to a problem, or asks for an action to be undertaken. An *offer* is a message that indicates the interest and ability to receive and solve certain requests. Note that I equalize knowledge and offers in many places below, although an offer usually only indicates the source of a piece of knowledge instead of the knowledge itself. Requests and offers exist in both the NetKP and sub-KPs, but they are of different types. For example, requests in the NetKP are for network knowledge, while requests in a sub-KP are for the specifi c knowledge in that sub-KP. A key problem in the KP is how to make requests and offers meet so as to solve a problem.

I believe that there are two base aspects of agents; each agent provides at least one of these. One is request/offer propagation and aggregation, and the other is request satisfaction which often involves reasoning and inference. We believe that there are two reasons to separate these two aspects. First, propagation and aggregation are common tasks in the KP, and they are similar in all the agents. Second, request satisfaction often requires reasoning and inference techniques, which are different in different agents. Depending on their capacity, willingness and availability, agents may have both or one of the components. These two separable aspects reflect the fact that different agents may have different roles and capabilities in the KP: some agents are responsible for receiving requests and responding to them, and some manage the knowledge, and some do both.

Agents are deployed by different parties, including end users, ISPs, application developers, etc, so they have access to different kinds of knowledge. Currently I assume static agents instead of mobile agents, and communication between agents is through message passing. In the future we may evaluate the need for mobile agents.

D. Region-based Organization

Due to their large number, agents need a scalable organization. To make the KP scalable and effi cient, I follow the divide-and-conquer strategy by dividing the KP into regions. The region is a new design element in the network architecture for large scale, wide distribution and heterogeneous networks [7]. A region is an entity that encapsulates and implements scoping, subdividing, and crossing boundaries of sets of entities. In this work, I use the region as the building block of the KP for the following reasons. First, due to the large scale of the KP, we need to introduce some organizing structure into the KP to achieve scalability and effi ciency, and region is a natural structure that fi ts our need. Second, as a general and flexible networking mechanism, a region or set of regions can be tuned to match the Internet structure which consists of tens of thousands of different administrative domains, or match the inherent structure of a sub-KP, as we discuss later.

Regions exist both in the NetKP and sub-KPs. Agents are grouped into different regions according to different criteria, such as network proximity, interest, etc, as we discuss in the following sections. Agents both inside and outside a region work together to monitor the region, and agents in the region can access more detailed network knowledge timely, while agents outside can provide a more comprehensive view of the region.

E. Knowledge

Knowledge as a term is more common in the AI community than in the networking. As a general term, knowledge in this proposal refers to any useful information in the Internet, including the information about individual objects in the network, and the relationships between objects, etc. There are various kinds of knowledge in the Internet. We categorize knowledge into two kinds: basic knowledge at the network layer, and specific knowledge in different areas of interest. Sources of knowledge include humans, measurements and monitoring, and inference, reasoning and learning. A specification of a knowledge domain is usually defined in language or ontology, such as XML, RDP or OWL [9], [10]. Because our emphasis is not on such a language, we will choose one (probably OWL) to meet our needs, but are unlikely to explore questions as the nature of a better or more suitable language.

Due to the distributed and heterogeneous nature of the knowledge in the networks and its large amount, we will need a distributed knowledge base. I will address issues including knowledge collection, storage and distribution in the following sections, but the knowledge base is not the focus of this proposal.

F. Trust Model

Trust is an important issue in the knowledge plane, as agents come from different sources. If agents are deployed by the authoritative local organizations, we call such agents "authoritative agents" within the local network (an AS or a corporate network), and assume that agents will not be malicious. But still, ISPs may be reluctant to admit failures within their own networks. As another example of a desire to hide the local state, consider agents residing on end hosts. An agent that represents an end user may want to hide the fact that worms start from his machine due to his fault. Therefore, we can see that agents must be prepared not to trust each other completely.

In this prototype knowledge plane, to make things simple, we assume that agents may provide incomplete knowledge, but they will not provide false knowledge. In the future, we will build a trust model that considered both authentication and reputation.

Even if all agents are honest, we still need to consider how to maintain consistency among knowledge from different agents. An agent must be able to resolve the differences when it receives different answers from different agents. The differences may be due to many reasons. For example, a multi-homed agent may find multiple AS paths between itself and a remote host, and it has to be able to tell which one is what it wants (maybe both). As another example, an agent is likely to receive different answers from different agents on the latency between two hosts, as agents are at different locations and may use different methods to obtain the latency.

III. Network Knowledge Plane

The NetKP is an application-independent distributed system that provides network knowledge to help construct efficient connectivity graphs for network management and applications. There are two types of uses of the NetKP. The first and obvious one is to provide information to KP applications. Initially, this information will be the basic information, measurements and other kinds of network specific information, both reasonably static and often quite dynamic. The NetKP may be seeded with basic information, but quickly it will also be extended with new, more accurate, more complete, or otherwise more extended information. In addition, sub-KPs also have need of information, both in terms of the underlying information about the functioning of the network and for finding and storing constraints and organizational structures of KP applications.

In this section, we discuss how to build the NetKP. I first describe the functions and interface that the NetKP provides, then agent organization that follows Internet topology, request propagation and resolution in the NetKP, and other issues.

A. Network Knowledge and Functions

As an initial classification, we divide the network knowledge into three categories: static, dynamic, and policy-related knowledge. More knowledge will be added later. The network knowledge is provided by agents in the NetKP to agents in sub-KPs, and various applications through the functions defined below. Accordingly, a beginning set of functions are defined in the following.

1) Static network topology. This topology knowledge refers to network topology at the AS level. Such knowledge is stable, and changes infrequently. The following functions are defined to return network topology knowledge. They correspond to the requests that an agent receives either from other agents in the NetKP or from agents in sub-KPs or from network applications.
   * *getASN(IP)*. Given an IP address *IP*, return the AS number where the IP address reside. This tells the topological location of a host.
   * *getASPath(AS1, AS2)*. Given two AS numbers, return the AS path from *AS1* to *AS2*. This gives a measure of the topological distance between two hosts.
   * *getLocation(IP, [metric])*. Given an IP address *IP*, return its geographic location in the form of *metric*. The metric can be longitude and latitude, zip code, etc.
2) Dynamic network performance. This knowledge includes latency, available bandwidth, loss rate, etc. Such knowledge usually changes frequently. Dynamic knowledge is very different from static knowledge. Dynamic knowledge usually changes frequently with time, and needs to be measured at runtime; in contrast, static knowledge is stable, and can be easily replicated and stored at multiple locations. The following functions are defined:
   * *getLatency(IP1, IP2, [time])*. Given two IP addresses, return the latency between *IP1* and *IP2*, and *time* is an optional parameter.
   * *getBandwidth(IP1, IP2, [time])*. Given two IP addresses, return the bandwidth from *IP1* to *IP2*.
   * *getLossRate(IP1, IP2, [time, accuracy])*. Given two IP addresses, return the packet loss rate from *IP1* to *IP2*. As loss rate is often small and hard to measure, *accuracy* specifies how accurate the returned answer should be.
3) Network policies. As the Internet consists of thousands of administration domains, and each domain define its own policy on route announcements, firewall rules, etc.
   * *isPortBlocked(port, AS)*. Given a port and an AS number, this function tells whether that port is blocked by that AS.
   * *getCost(IP1, IP2, metric)*. Return the cost of the path between IP1 and IP2 in term of the cost metric.

The function set above is just one set currently provided by the NetKP. Additional network knowledge will be provided, and more sophisticated functions can be built on top of the primitives. For example, given two pairs of IP addresses, we can use *getASPath* to determine if the paths between the two pairs of IP addresses intersect at some AS. As another example, agents can collaborate with each other to monitor the networks, and locate possible failures in the network [11]. Therefore, we expect that more functions will be added to the NetKP as required by new applications.

Every agent maintains at least two kinds of network topology knowledge at the autonomous system level: (1) the AS number of any IP address; (2) the AS path between the local AS and other ASes. From (2), we can also derive the degree of the local AS. Therefore, an agent only maintains its local view of the Internet. This is due to the following reasons: first, it is hard to get an accurate global view of network topology, while it is easier to obtain the local view of network topology; second, such local network knowledge should be able to satisfy local requests most of the time; third, remote network knowledge can be obtained from other agents through request resolution. Note that restrictions may be applied to the parameters of those functions. For example, an application may not be allowed to query the AS path between any two ASes, due to the privacy concerns of routing information of those ASes. Network topology knowledge is fairly stable, and is updated infrequently. Local BGP feeds provide an accurate view of the Internet from the point of the local AS, so BGP feeds from the local AS are the best source of topology information.

Geographic information is another kind of network knowledge we are interested in. Geographic information provides physical location information, which is often directly related to network performance, such as latency. It also enables a large set of location-aware applications. It is not trivial to obtain geography information today, but often approximate location is enough. We can leverage the existing techniques [12].

Performance knowledge is often complicated to obtain and maintain for two reasons. It is usually easy to obtain latency using measurement tools such as *ping*. Unless latency to a large number of hosts is needed, real-time measurement will work due to its simplicity and low overhead. Other information, such as bandwidth and loss rate, can be obtained through measurement with more overhead. Many tools have been developed to measure network status, and new tools are being developed. Agents can use those tools and share performance knowledge.

Note that the performance and geographic knowledge may be approximate instead of accurate. First, the performance knowledge changes frequently. Even if we obtain accurate measurement results, it may be outdated when it is returned to the requester. Second, in many situations it is enough to have approximate information. For example, a streaming video application only needs to know the class of bandwidth (high, medium, low) to determine the appropriate encoding method.

B. Agent Organization

1) Agent Discovery: Agent discovery deals with the problem of how a new agent finds agents already in the KP. In the NetKP we assume authoritative agents are configured manually to connect to nearby agents and agents in the neighboring regions. Non-authoritative agents join the NetKP by notifying a nearby authoritative agent, and that agent broadcasts this information to nearby authoritative agents. In my thesis I plan to explore more robust and automatic mechanisms for agent discovery.

2) Agent Organization: As the NetKP provides network topology knowledge about the Internet, a natural way to organize agents is to follow the network topology, including AS-level topology, corporate networks, etc. Here a region corresponds to an administrative domain in the Internet, which is either an autonomous system, or an institutional network such as the MIT network, or a corporate network such as HP. A large AS may be divided into several regions. Autonomous systems are very heterogeneous, and we may need to divide an autonomous system into several regions.

Agents are divided into three categories based on their owners and privileges. One is the authoritative agents delegated by ISPs, institutions and organizations in their own networks. They have (limited) access to the BGP information at border routers, and enforce certain policy when exposing this knowledge. The other is the agents residing on end hosts. These agents help the authoritative agents and provide more complete and accurate knowledge. A third category from the perspective of a region is the agents outside the region, both peers and other kinds of agents.

To facilitate the communication between authoritative agents, each agent maintains two lists. The first is a list of agents in the local region. Agents within a region periodically exchange network knowledge. The second list contains agents in neighboring regions. Typically, if a region matches an AS, the list includes agents in its providers, customers and peering ASes. This organization is shown in Figure 2.

Agents that are not authoritative connect to nearby authoritative agents. These agents provide not only backup for authoritative agents, but also valuable performance knowledge.

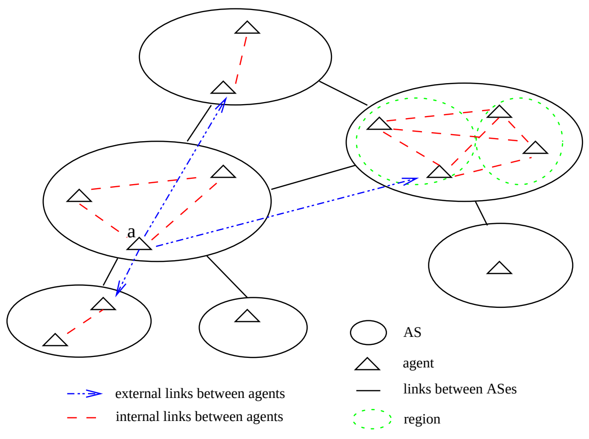

Fig. 2. Agent organization in the NetKP. Ellipses are ASes, small triangles are agents, solid lines are links between ASes, dashed lines are links between agents within a region, and dash-dot lines with arrows show links between agent *a* and agents in its neighboring regions. Dotted green circles show that a large AS is divided into two regions.

C. Request Propagation and Aggregation

A key function of the NetKP is to resolve requests for network knowledge. Requests for network knowledge can be issued by different entities. Agents in the NetKP may issue requests to obtain certain knowledge so as to organize themselves effectively. Requests for network knowledge may also come from agents in sub-KPs or network applications to construct their connectivity graphs. Requests can be about properties of different entities in the networks. A simple request example is on the latency between two hosts, while complicated requests can be *isPortBlocked* or on the network condition of an AS. In the latter case, sometimes it is not clear which agents are responsible for or can resolve the request. In that case, an agent that receives the request may initialze the operation and work together with other agents to resolve the request. Such an operation can be complicated and costly, and may require authorization. Here I start with simple requests, and will address more complicated issues in the near future.

A request for simple network knowledge is resolved by the local agent as follows. For some requests, the agent resolves them directly and returns the responses to the requesters, without consulting other agents. This occurs when the local knowledge base already contains the necessary knowledge. Examples of such requests are *getASN* and *getASPath* when the source IP is in the local AS, as the local BGP table contains the mapping between IP addresses and AS numbers and the AS paths originating from the local AS to all the other ASes.

For requests that require non-local knowledge, the agent forwards requests to the agents that have the knowledge to get the answers. Because the local agent has the knowledge of AS paths from the local AS to all other ASes, it can fi gure out which agents in which AS(es) should be able to resolve the request. Therefore, it forwards the request following the corresponding AS path to an agent in a neighboring AS, and that agent again forwards the request to its neighboring AS following the same AS path until the request reaches an agent in the destination AS. The agent returns the answer to the source agent following the same AS path reversely, and agents along the path may cache the answer. By following the reverse path instead of the default reverse AS path and agents caching along the path, we construct an implicit aggregation tree, which helps to resolve similar requests more quickly in the future, because agents along the AS path may resolve the request if they happen to have cached previous answers. Figure 3 shows the resolution procedure.

Request aggregation is crucial for scalability, as many similar requests may be issued from different places simultaneously but we cannot afford to propagate all of them individually. By matching the underlying autonomous systems, the NetKP provides a natural aggregation structure

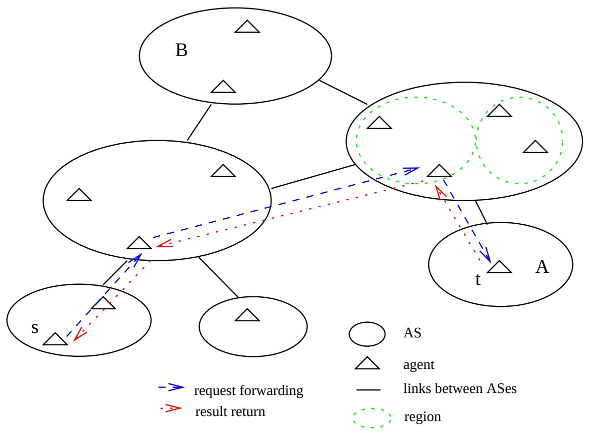

Fig. 3. Request resolution in the NetKP. Agent *s* issues a request for the AS path between AS *A* and *B*. Agent *s* forwards the request to agents along the AS path, until it reaches agent *t* in AS *A*, as shown in the dashed arrow lines. Agent *t* checks its local knowledge base, and returns the corresponding AS path, as shown in the dotted arrow lines.

along the AS path. Consider *getASPath* as an example. When requests from multiple locations are issued for the AS path between AS *A* and *B*, the requests will follow the AS path towards AS *A*, therefore they naturally converge along the AS path. If an agent on the path has cached the answer, all the following requests passing here are resolved at this agent. If not, the agent can aggregate the requests and forward only one request.

Currently I assume a request-oriented model in the NetKP, and there is no active knowledge dissemination. Note that responses to requests are cached along the returning AS path. This is one form of knowledge dissemination. In my thesis I will extend this model to incorporate knowledge dissemination. One possible method is a publish/subscribe-based mechanism.

D. Knowledge Collection

1) Topology Knowledge: Agents collect network topology knowledge from the local AS. To do so, agents need authorization to access BGP tables from local border routers. We believe that BGP information is usually not sensitive, so organizations may be willing to disclose this information. Furthermore, agents can employ policies when exposing such information to others.

2) Performance Knowledge: All the above discussions are based on network topology knowledge. Another category of knowledge is network performance, which is very different from network topology knowledge. Such knowledge is much more dynamic, and often requires real-time measurement, thus is more costly.

A request for performance knowledge between two hosts is resolved by agents near the hosts. For example, the latency between two hosts can be approximated by the latency between two agents plus the latency between each host and its nearby agent, similar to [13]. As another example, agents may infer the property of a new path by segmentation and composition using previous measurement results. This is similar to network tomography [14]. Consistency is an important issue here, as agents at different locations may return different answers to the same request. In my thesis, I will discuss this in more detail.

E. Discussion

This agent organization has the following advantages. First, such organization provides network knowledge to KP applications. By propagating requests to the correct agents, applications can obtain the correct knowledge.

Second, the mechanism is scalable. Network applications ask their local agents for network knowledge. Most requests can be resolved locally, as applications usually care about the knowledge related to or originating from the local network. Local agents only ask other agents when they cannot resolve the requests.

Third, request resolution is efficient. Most requests are resolved locally, and it is fast to retrieve local network knowledge. Furthermore, requests and results are naturally aggregated along the AS path, so similar requests can be aggregated and resolved without going all the way to the original agent.

Fourth, the operation overhead is low. The overhead of knowledge collection by agents is amortized among network applications. Even the performance knowledge can be obtained at a low cost, as agents are able to get approximate performance information from nearby routers or by segmenting and recombining previous results without measuring at real time.

The insights of the NetKP are two folds. First, valuable knowledge at the network level is exposed to network management and applications. Second, the overhead of this exposure is amortized among network applications.

IV. Network Management Sub-KPs

For the purpose of network management, we propose to build multiple sub-KPs, one for each network service. On top of the sub-KPs, we can build network management applications as syntheses of sub-KPs. In this section, we discuss the issues of utilizing the NetKP to construct efficient sub-KPs for various network services. Note that in the following discussion, without special specification, agents refer to those in the sub-KPs, not in the NetKP.

There are three significant differences between knowledge in the NetKP and in sub-KPs. The first is diversity of knowledge. The scope of knowledge is limited in the NetKP. In network management, there are various tasks and many different kinds of knowledge in different sub-KPs. For example, some agents have knowledge on the DNS system, while some others have knowledge on end-to-end connectivity. It is not very useful to connect together agents with these two different kinds of knowledge, and those agents are only interested in knowing other agents with similar interests and knowledge. Therefore, we first address how agents with similar knowledge and/or interests discover each other.

Second, agents in the same area of interest need an efficient organizing structure among them. However, unlike in the NetKP, sometimes there is no predefined structure to follow (like ASes in the NetKP), or the predefined structure cannot provide enough information. I discuss some comprehensive criteria to construct efficient connectivity graphs in sub-KPs using both network knowledge and area-specific knowledge.

Third, some agents in the sub-KPs do not reside on dedicated servers; they run on end hosts. Therefore, agents may join or leave at any time. We need to construct a robust organization among them.

A. *Constraints on Sub-KPs*

The set of orthogonal constraints for organizing sub-KPs fall into five categories. The most obvious one is the physical or topological information. When a sub-KP is to be instantiated, the authority on whose behalf it is happening may have an interest in constraining it to run only in some part of the network or in some other location based region, such as geographical. For example an enterprise network manager might want to run a particular sub-KP within the scope of the enterprise network, simply due to topological reasons. A simple version of this may be AS or IP address based. In terms of geography, such a specification may be as general as named geographic region or as specific as ranges of longitude and latitude.

A second kind of constraint is external policy constraints. They are security policies, pricing or economic policies, and other incentive-based policies. Security policies specify hard constraints on which information and functionality can and cannot be made available across specified boundaries and by whom. Pricing constraints allow for perhaps a sliding or degree based decision. Other forms of incentives may be designed specifically to encourage cooperation, in the face of proprietary and other security constraints.

The third type of constraint on the organization is efficiency. Applications may have to run with a set of efficiency criteria, which may determine the placements of agents. For example, if it is important to have low latency, then paths between agents should be selected on that basis; if the amount of network traffic should be low, then agents may be collocated on the same nodes as much as possible.

The fourth set of constraints is on shared resource usage. This requires that sub-KPs, in determining the organization of one KP application, know enough about others which may be sharing resources to make a possible compromise in order that the KP applications not interfere unnecessarily with each, as best possible. We will take a lead from previous work, beginning with a set of information similar to that of CoMon [15] from PlanetLab. CoMon provides a monitoring statistics for PlanetLab at both a node level and a slice level. It can be used to see what is affecting the performance of nodes, and to examine the resource profiles of individual experiments. We will extend and modify CoMon as needed to fit our purposes. In the longer run, it will be necessary for this sort of information to be distributed in the KP, unlike what is currently being built in PlanetLab.

Finally, there are constraints from the particular functionality or implementation. For example, if we design a diagnosis system for the DNS system, the naming hierarchy and the zone structure will inevitably play an important role in such a sub-KP. As another example, intrusion detection itself does not impose any constraints on the corresponding sub-KP.

B. Agent Discovery

An important issue in sub-KPs is how to discover agents with similar interests when an agent joins the knowledge plane. My discovery procedure takes advantage of the underlying application-independent NetKP. Each region in the NetKP maintains a list of local agents in sub-KPs and their interests. A request looking for agents with similar interests is propagated from the local region to neighboring regions. In most cases, there is already a sub-KP with this interest, and the new agent only needs to find one that is already in this network to join it. Furthermore, a high-level task may involve multiple agents belonging to different sub-KPs. For example, to diagnosis a web access failure, we need help from agents on DNS diagnosis, routing diagnosis, server status monitoring, etc. To find those agents, we need to first find local agents in the NetKP, and then search for agents in each sub-KPs.

To organize agents with similar interests together, another issue is to determine that agents with what degree of similarity in interests should be in the same sub-KP. The complexity comes from not only the granularity of knowledge or interests, but also the connections between knowledge/interests. As an example of granularity, network management can be divided into many tasks such as DNS management, traffic management, etc. Should we organize agents interested in network management into one knowledge plane, or into multiple sub-KPs, each of which focuses on one type of task? Furthermore, network management can also be classified according to functions: performance management, fault management, security management, etc. Should there be different networks of interest for each of them? How do we implement an efficient search on DNS fault diagnosis (intersection)? What if some one is interested in everything (union)?

Therefore, the following are the decisions we have to make to determine whether a sub-KP is needed:

1) Granularity. This depends on how many agents are interested in a certain kind of knowledge.
2) Intersection. Some quests may require collaboration among networks of interests. The networks should be structured so as to facilitate such operations.

C. Agent Organization in Sub-KPs

Similar to the NetKP, each sub-KPs consists of multiple regions. Agents are organized in a sub-KP into different regions. Unlike in the NetKP, sub-KPs can be very different in their inherent structure. Some sub-KPs have predefined structures related to their specific areas, while in some sub-KPs there is no existing structure. Therefore, we need to construct regions based on both common criteria and special feautures in that sub-KPs. A general organizing principle is as follows. Each agent computes the *distance* and connects with multiple nearby agents. Agents use distributed leader election mechanisms [16] to cluster themselves into regions based on distance. The proximity is defined as a comprehensive metric considering several factors as described below. Depending on each specific area of interest, agents may be organized differently.

1) Distance: To cluster agents into regions, we first define the criteria of clustering. Clustering is based on *distance*. We define four types of *distance*. First, *topological distance* is the distance in terms of network topology. There are different granularities of topological distance. The distance can be the number of AS hops between two agents, or the number of router-level hops. Due to the heterogeneous nature of ASes, we need to classify ASes into tiers [17], so as to better characterize the size of an AS.

Second, *performance distance* is the performance of a network property between two agents. The most common performance distance is latency. However, in some cases we may be more interested in the bandwidth between two hosts, and thus bandwidth becomes the metric of distance in that case. Therefore, performance distance may have different definitions in different situations.

Third, *geographic distance* is defined to be the physical distance between agents. In today's Internet, it is not easy to learn accurate geographic distance without GPS. But in our case we only require coarse geographic information, such as whether two agents are in the same city, which is feasible.

Fourth, even within a sub-KP, each agent may have different interests. I define *interest distance* as the measurement on the degree of the distance between agents.

To cluster agents together, multiple kinds of distances need to be considered together, because each kind of distance alone has its advantage and deficiency. For example, if AS distance is used as the only metric for clustering, two agents in a tier-1 AS may be clustered together while they are actually in Boston and Los Angeles respectively; if geographic distance is considered solely, two agents in the same city may be clustered together while they may belong to two small ISPs and have little to do with each other. Therefore, we need a comprehensive metric for clustering. A naive scheme is to have a weighted average of all the four distances. However, it may not be easy to measure interest distance as a number. The ultimate goal is to create a gradient within a sub-KP so that requests and offers can meet each other naturally and efficiently. In my thesis, I will further explore the distance criteria.

2) Clustering of Agents: Within a region, agents can be organized in different ways, depending on the size of the region and distribution of the distance metrics. When the size is small, a full mesh will suffice. That is, each one knows all the others. When the size is large, a structured peer-to-peer organization, such as distributed hash table, may be preferred. In general, we want to limit the size of a region so as to simplify its organization and recommend a full mesh structure. There is no single answer to the region size. It will depend on what they are doing under what conditions.

So far we have not talked about agent mobility. With the increasing demanding on mobile devices and the wide use of laptop and wireless networks, mobility is becoming more and more pervasive. If an agent moves, it should leave the current region and join the new local region. This will be the future work.

3) Organization of Regions in Sub-KPs: Agents in different regions need to collaborate with each other. We have at least two options for the organization between regions. The first is a hierarchical structure. Since agents are clustered based on distance, it is natural to build a hierarchy among regions, similar to the AS topology of the Internet. It can be a multi-layered structure. The advantage is that this structure is natural for aggregating information and suppressing redundant requests. However, a hierarchical structure is hard to maintain when agents join and leave the system frequently.

The second choice is peer-to-peer structure. There are two kinds of peer-to-peer organization: unstructured [4], [18] and structured [19], [20], [21]. Peer-to-peer organization is robust to churn, but its flat structure makes it hard to aggregate and suppress similar requests. We will address this problem late in this section.

D. Request and Offer Propagation and Aggregation

1) Request and Offer Propagation: The way that requests and offers propagate determines the efficiency of request resolution. Usually the request propagation is called “pull” and the offer propagation is called “push”. The key issue is a trade off between push and pull in different scenarios. In the KP, we need to balance push and pull. A similar technique, directed diffusion, is widely proposed in sensor networks [22]. However, unlike sensor networks, the Internet is larger in scale, and more complicated in structure. Both the scale of the Internet and its organization, which is generally not ad hoc or self-organizing, suggest that the current form of directed diffusion will be inadequate for the KP.

To facilitate request satisfaction, there are many possible request and offer propagation schemes. A simple scheme is as follows. When an agent issues a request, it is propagated to all agents in the same region, and also to one or a few agents in every region within a certain number of *region hops*, say *n*. When an offer is issued, it is also disseminated in the same way. Therefore, an offer within *n* region hops is guaranteed to be found by a request, and an offer out of *n* region hops may be missed. Figure 4 shows an example with *n*=2. This shows how regions help to improve scalability and efficiency. To further extend this, we can consider many factors, such as the position of an agent, the cost model, and even the diameter of the sub-planes, etc. This will be studied in my thesis.

2) Request and Offer Aggregation in Sub-KPs: Aggregation is a fundamental function in the KP. I have designed a peer-to-peer aggregation mechanism in [23], and plan to modify it for request and offer aggregation in sub-KPs. Peer-to-peer networks represent a robust way to organize information, since there are no central points of failure or bottleneck. However, the flip side to the distributive nature of the peer-to-peer structure is that it is not trivial to aggregate and broadcast information globally. I designed a novel algorithm for this purpose. Specifically, I build an aggregation tree in a bottom-up fashion by mapping nodes to their parent in the tree with a parent function. The particular parent function family I propose allows the efficient construction of multiple interior-node-disjoint trees, thus preventing single points of failure in tree structures.

However, the above mechanism is not enough for the KP, because it is not a general framework that can fit the needs of KP, and does not consider proximity and administrative isolation. We need to incorporate those factors into the new aggregation mechanism in the sub-KP and take advantage of the topology-aware agent clustering structure. A simple version will be a multi-layer aggregation: first, a local aggregation is performed within each region; then a global aggregation is performed between regions. This part also needs further exploration.

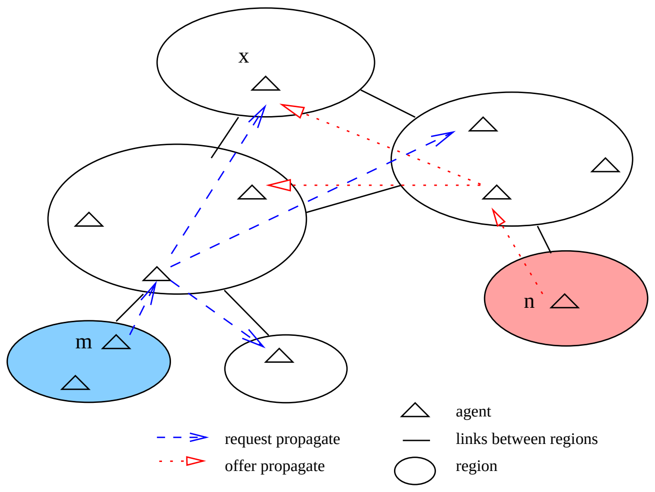

Fig. 4. Request and offer propagation in a sub-KP. This shows a request from agent *m* and an offer from agent *n* meet at agent *x*.

V. Case Study: Network Management

In the following sections I present several case studies in various areas, including testbed management, fault diagnosis, DNS diagnosis, intrusion detection, etc. By conducting the case studies, I want to: (1) demonstrate that the KP is a useful infrastructure to various applications; (2) learn from the case studies about different requirements and features of those applications so as to further improve the KP.

There are several key aspects to a KP application. The primary one is the functionality that defi nes it. We suppose that the underlying agent system provides a variety of communication patterns available to agents. Some agents may be designed to broadcast or multicast their fi ndings to a set of coordinating agents. We expect that a variety of such communication patterns must be available in the agent system, and a basic set include: broadcast, multicast, anycast, and one-to-one. Some conditions for communication will be important, such as timing intervals. Another characteristic is policy or economic requirements and incentives. Thus, we expect that a KP application will at least initially provide a specifi cation of its communications patterns and constraints as well as policy constraints and requirements.

Network management requires collaboration of multiple agents, and often multiple network management sub-KPs. In this section, we consider both network testbed management and fault diagnosis. However, this part is still sketchy, and more study is needed.

A. *Testbed Management*

We are working on improving the DETER [24] testbed management by designing “Distributed Diagnostic System” that monitors the status of experiments running on the testbed and the testbed itself. This will provide an interesting example not only of organization of agents, but also allows for defi ning ”regions” across which information flow must be stringently controlled. It will also provide a compelling argument for the need for extensibility, and perhaps the ability to consider the trade off of costs of KP activity.

The initial design is as follows. A KP runs on the testbed to monitor its status. Because currently DETER has two locations, ISI and Berkeley, we propose two regions, one for each location. KPs at different locations periodically exchange information. For each experiment to run on DETER, a region is created to monitor the experiment based on the user’s specifi cation. When an experiment runs at multiple locations, a region is created for each location, and these regions exchange information among them. Figure 5 shows its structure.

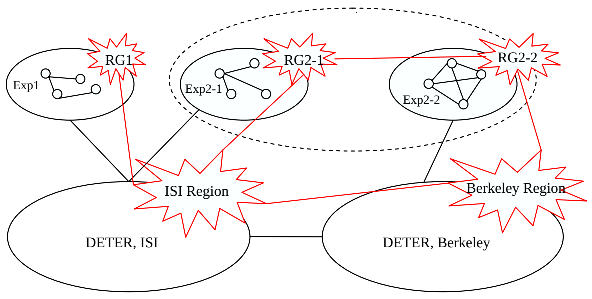

Fig. 5. DETER testbed management.

The DETER KP has two tasks. First, it monitors whether the static environment needed to run is set up correctly, including the generated virtual topology, bandwidth, and the OS and code loading, Second, it monitors if the experiment runs as specified. A user needs to specify the behavior of his experiment beforehand. This is very important to DETER, as experiments on DETER are security-related, and DETER has to guarantee not only no interference among experiments, but also the security of the testbed itself.

B. *Connectivity Failure Diagnosis*

Steinder et al. discussed various techniques and algorithms on Internet fault diagnosis in [25]. The diagnosis algorithms come from many different fields including decision trees, neural networks, graph theory, etc. They also propose a hierarchical partitioning of the problem space [26]. In that approach, a diagnose manager divide the diagnosis into multiple domains, and each domain performs local diagnosis separately. Then local diagnosis results are collected and analyzed by the manager to draw a global conclusion. In a similar spirit but more broadly, I propose to diagnosis network failures from end-to-end point of view, so our approach crosses multiple network layers and services, not just about routing.

There are many causes of connection failures. An incomplete list includes [27]: local misconfiguration; link failure; routing failure; congestion; DNS failure; destination failure, etc. Fault diagnosis usually involves the following steps: fault detection; fault localization (root cause analysis); fault source identification. Here I focus on how is to pinpoint the origin of a fault.

Assume there is a local agent running on each host. When a user cannot visit a website, he notifies the local agent on his computer. The local agent first checks if it is a configuration problem with the end host. If not, it tries to find nearby agents through the agent discovery service.

The diagnosis requires collaboration between several sub-KPs. First, the local agent asks the DNS sub-KP to see if the DNS resolution works correctly. If so, it then asks the connectivity sub-KP to see if the end-to-end path between the local host and the website is working. If so, the local agent finally needs to find out if the web server is running. Sometimes we need the help of agents. For example, we may need traceroute from multiple vantage points to determine the location and scope of the failure [11].

VI. Case Study: Intrusion Detection

Intrusion detection in large-scale networks requires a new approach to monitor and respond to security incidents. Host-based collaborative intrusion detection has been a promising direction. A key challenge in a collaborative intrusion detection system is that alerts from end hosts need to be propagated efficiently and quickly among participants so that an intrusion decision can be made before the worm infect most of the hosts. Many current mechanisms use simple gossiping protocols or peer-to-peer protocols [28], [29], [30]. We believe a good cooperative messaging protocol can be made more efficient by taking advantage of the knowledge from the network. In this case study, we propose a region-based message propagation protocol. We implement our mechanism on DETER testbed and compare it with a popular gossiping protocol. By doing this case study, I want to demonstrate that there can be both accuracy and efficiency improvements for intrusion detection to take advantage of the KP.

We propose to organize hosts into regions using the knowledge from the NetKP. Hosts are clustered into regions based on their distance. Within each region, hosts elect a leader using distributed leader election algorithm periodically [16]. Leaders of different regions form a complete graph if the number of leaders is small or other organizations such as multiple disjoint trees [23]. When hosts detect potential intrusion attempts, alerts are sent to its local leader directly. The local leader aggregates alerts from local hosts, and then exchanges this information with leaders of other regions. Therefore, each leader has an approximate global view of the intrusion situation. Whenever a leader has enough information to make a decision, it announces its decision to both its local hosts and all the other leaders. Leaders may or may not need to reach an agreement on the decision, depending on their policy and administration. Our mechanism is similar to [29] in the hierarchy of local detectors and leaders (global detectors in [29]). The difference is that in [29], each local detector sends its decision to a randomly selected subset of a large set of global detectors. The local detectors have to maintain the information of all or most global detectors, and the communication does not consider any network-layer information.

Besides a better organization of hosts, we also share more detailed information on intrusion attempts between hosts, to reduce the false positive rate. Currently our method is to have each host send basic intrusion information, including the port number(s) and source IP address(es). The leader can therefore either determine that the alerts are false as they are at different ports and from different sources, or there are multiple intrusions simultaneously, or there is an intrusion of the same worm from multiple sources.

I have conducted a preliminary experiment on DETER, in which our method and a gossiping protocol are compared in terms of the efficiency of distributed decision-making in collaborative intrusion detection systems.

The gossiping protocol is proposed in [30]. Its decision making is completely distributed where hosts exchange information using an epidemic spread protocol without any organizing structure or consideration on network topology. When a potential intrusion is detected by an end host, it forwards the alert to m randomly selected neighbors, and then its neighbors forward the alert to each of their m neighbors, together with their own observations (alert or not), and this continues. Usually m equals 1 or 2 for scalability. Each host computes the possibility of intrusion using all the alerts it has received. If a host determines that there is an intrusion, it broadcasts the decision to all hosts.

Both methods use distributed sequential hypothesis testing to detect global intrusion. Sequential hypothesis testing was first adopted to intrusion detection by Jung et al. in [31]. The algorithm in Jung's work is centralized, and it is extended to a distributed algorithm in [30] where each host performs the inference individually. In our method, inference is only carried out by leaders, not other hosts, as hosts only have their local observations. Our method is semi-centralized, and is robust to DDoS attacks, because of two reasons. First, there are multiple widely distributed leaders, and any of them can make the decision. It is hard to attack all the leaders simultaneously. Second, leaders are periodically re-elected distributively, so it is hard for attackers to predict the leaders when attacks happen.

I ran experiments on 42 nodes on DETER [1]. The experiment topology was similar to a

[1]That is the largest number of nodes I could get assigned by DETER. Although the number of free nodes is usually around 100, I never get all of them because of various limitations such as available connections, hidden nodes in the middle, etc.

dumbbell, except there was a LAN at each end, as shown in Figure 6. Intrusion was emulated using WormSim [32]. Intrusion was detected when a non-vulnerable host received an access attempt on an unserved port. The parameters of the experiment are in Table I, which were the same as those in the experiment in [30].

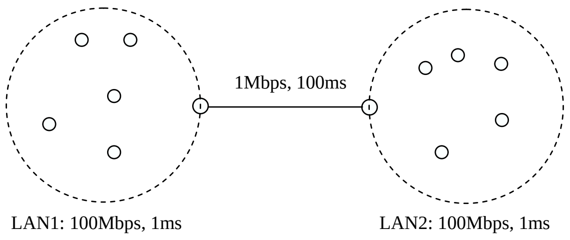

Fig. 6. Intrusion detection experiment topology.

Sequential hypothesis testing	
False positive	0.1
False negative	0.1
Desire false alarm rate	0.02
Desire detection rate	0.98
Decision number	8
Experiment settings	
Network size	42
Vulnerable nodes	25%
Non-vulnerable nodes	75%
Worm propagation rate	1 message/second
Method specifi c parameters	
Gossiping rate m	2
Aggregation rate	2

TABLE I
INTRUSION DETECTION EXPERIMENT PARAMETERS.

To evaluate the performance, I compared two aspects: detection speed and cost. Detection speed measures how fast hosts can reach a decision on intrusion detection. Detection cost consists of two metrics: (1) the number of messages that hosts propagate to reach a decision (network traffi c overhead); (2) the number of infected nodes by the time of decision.

One set of the experiment results is shown below in Figure 7. The results are the average of 5 tests. *Gossip* refers to the gossiping protocol in [30] and *Leader* is our region-based protocol. Our mechanism adaptively organizes hosts into two clusters, corresponding to the two LANs, based on network topology and properties (latency, bandwidth, etc). A host in each LAN is elected as the leader. In contrast, the gossiping protocol randomly chooses *m* hosts to forward messages. We can see that *Leader* outperforms *Gossip* in all the three metrics. *Leader* is faster in detection time because alerts are collected within each region fi rst before being exchanged among leaders, while in *Gossip* messages may cross the slow link (100ms) many times before reaching a decision. Therefore, the number of infected nodes is also smaller in *Leader* thin in *Gossip*. Mostly signifi cantly, the number of messages in *Leader* is greatly fewer than that in *Gossip*. The reason for this is because that the number of messages increases almost exponentially among hosts in *Gossip*, while in *Leader* each message is only sent to the local leader, and then exchanged among leaders with aggregation [2].

[2]Note that the messages do not include those for maintaining the cooperation among hosts in their method *Gossip* or messages for leader election in our method.

Those experimental results suggest that the network knowledge plane can improve intrusion detection by providing valuable network-layer knowledge to intrusion detection systems. In my thesis I will follow up with a series of further experiments:

1) Allow the gossip protocol to run locally within a cluster. This will allow us to improve our understanding of the tradeoff between network costs (traffic, delay, etc.) and both speed and correctness of detection.
2) Move to the model in which local and global detection are distinct. We plan to study two different questions here. One is that each local detector can be specialized, as suggested but not implemented in [29]. In addition, a cluster of global detectors might be specialized by region or enterprise, and clearly need not be comprised of as large a set as the local detectors, allowing for improved scaling.
3) Create a security policy that prohibits the exposure of most local information, but allows for the report of definitive attacks. For example an enterprise might not allow any information about the nature of its network or the degree of compromise it might have experienced to go beyond its security perimeter, but might allow for a report to the rest of the net that it has authoritatively identified an attack with a particular signature. In return for allowing that information out, it can expect to receive similar information from other enterprises or business entities. This is an example of understanding and utilizing incentives effectively.
4) Enhance the reporting scheme to reflect a signature of a worm, if detection has occurred. This will provide two improved capabilities. The first is that when composite detection is done, it can be partitioned by worm signatures. We expect this to significantly reduce the impact of false positive reports. The second is that it will improve the reporting of a worm, by allowing for reporting of a particular worm signature, thus enabling the disentanglement of simultaneous worm attacks.

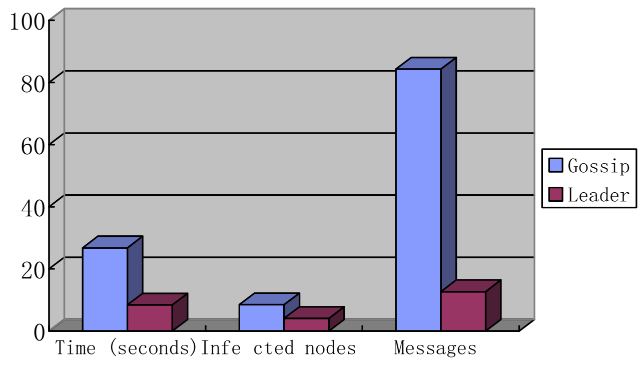

Fig. 7. Intrusion detection performance comparison.

VII. Case Study: DNS Diagnosis

In this section I present a case study on a DNS sub-KP whose task is to diagnose DNS failures. Unlike the case study on intrusion detection which has no inherent structure in itself, the DNS system has a well-defined naming hierarchy and zone structure, which will play an important role in organizing such a sub-KP for diagnosis.

There are several issues we need to consider first. One is the access to DNS servers. Many DNS problems are internal to DNS servers, such as delegation errors, synchronization errors, etc. Agents outside can observe some symptoms, but it is very hard to figure out the causes. In the following discussion I assume agents can access the internal state of DNS servers. I summarize the symptoms and causes of DNS failures in Table II. The agent organization should follow the DNS delegation, because that is how a domain name gets resolved, and regions are formed based on domain structure and other distances.

I am currently working on a DNS sub-KP as an example in network management. As we do not have access to name servers right now, the current experiment is limited to the end-to-end approach. An initial experiment setup for DNS fault diagnosis is illustrated in Figure 8. The experiment consists of multiple nodes in PlanetLab. The basic experiment consists of three steps, as shown in Figure 8. First, the programs running on multiple PlanetLab nodes maintain a list of various domain names. Second, periodically nodes issue DNS name resolution requests to the corresponding DNS name servers. Third, the resolution results are stored into a local database. Finally, the results are analyzed at the local machine. An improvement will be to enable interactive measurement. In the interactive measurement, the nodes will receive the analysis results from the local machine and change the requests accordingly, as shown as step 4.

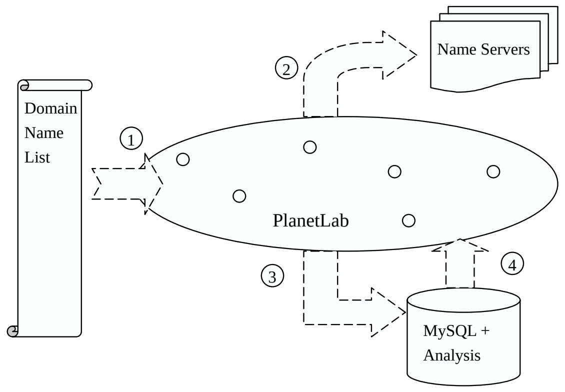

Fig. 8. DNS diagnosis experiment setup.

VIII. Related Work

There are many categories of related work to the knowledge plane, including publish/subscribe systems, interest-based overlay networks, content routing networks, etc, as many issues are involved in this work.

A. Overlay Networks

The closest related work to the NetKP is routing underlays [33], [34]. Routing underlays use network topology information to eliminate redundant virtual links in overlays to achieve scalability. Both routing underlays and the NetKP try to expose network topology knowledge to the applications. However, there are several differences. First, the NetKP is designed to be part of the KP under a general framework, and is to be extended to accommodate other kinds of knowledge. Second, it provides more knowledge than routing information. Third, it aims to help many applications in different ways, not only routing overlays.

Semantic overlay networks [35], [36], [37] are similar to the issues we have in organizing agents in the network applications in that both focus on the dissemination and sharing of complicated knowledge that requires semantic understanding.

Interest-based overlay networks such as [38] are similar to this work in that peers prefer to connect with others with similar interests. In that work, a node caches the information about other peers which provided useful results to recent queries, as those peers have shown similar interests, and are likely to provide good results for the future queries.

iPlane [39] is very similar to the NetKP in that both provide performance knowledge. Compared with many of its predecessors, iPlane provides a richer set of performance characteristics using several clustering techniques. Su et al. proposed a novel use of existing content distribution networks such as Akamai to improve overlay routing [40] by taking advantage of Akamai's open redirection mechanism. Both of them are similar to the knowledge plane in that they provide network knowledge to applications. One difference is that the knowledge plane is more general and provides not only performance knowledge but also topology and policy knowledge. iPlane only focuses on network information, while the knowledge plane will be extensible so that new kinds of knowledge can be made available.

B. Agent Organization

Another related work is large-scale measurement and monitoring infrastructures, which usually manage a large number of monitoring hosts. Many measurement and monitoring infrastructures have been proposed and built so far [41], [42], [11]. TAMI [41] is a measurement infrastructure that is both topology-aware and supports various scheduling mechanisms. But the topologies in TAMI are source and sink trees only, mainly for bandwidth measurement purposes. Those infrastructures only support basic operations between their members, mostly monitoring and probing. Our mechanism makes use of network topology such as autonomous systems information. Agents are organized based on their topological locations. In contrast with the simple behaviors in the previous measurement and monitoring infrastructures, agents in the knowledge plane can collaborate to perform intelligent reasoning. The prototype mechanism in this proposal is aimed to be a step forward from the simple infrastructure and distributed query processing toward the KP. KP will need to build on these and new measurement monitoring capabilities as underpinnings as well as possibly supporting the management and organization of such distributed tools.

C. Propagation and Aggregation

Publish/subscribe systems address problems similar to ours [5], [6]. There are three kinds of pub/sub systems: unicast, single-identifi er multicast and content-based multicast. Unicast systems transmit notifi cations directly to subscribers over the Internet's existing mechanisms. Single-identifi er multicast systems send messages to discrete message channels to which customers with identical interests subscribe. The most popular approach is content-based multicast systems, which forward messages based on the text content of the messages. The problem in the KP is more closely related to content-based multicast than the other two in that in our problem, requests and knowledge need to match each other based on content. However, our problem is more complicated because there are many heterogeneous knowledge sources with different interests and capacities.

Search has been an important component in many networked systems, especially in peer-to-peer systems. For example, Distributed Hash Tables use hashed IDs to fi nd specifi c fi les [19], [20], [43]. Gnutella [4] uses scoped broadcast of key words to search for contents, which is similar to our propagation problem. However, in the KP, more complete representations of requests and knowledge, and more sophisticated search functions are needed, such as range search, fuzzy pattern match, etc.

Content routing networks is a research effort to support name-based routing in the Internet [44]. To do so, routers need to support name-based routing which requires name-based aggregation. However, the current name aggregation mechanism is not scalable. The KP does not require the routing infrastructure changes, but we will need similar aggregation functionality to aggregate requests and offers among agents in both the network layer and the application layer.

Directed diffusion is an important data dissemination paradigm in sensor networks [22], [45]. We have a similar problem in the KP: how agents with similar interests fi nd each other, and how and where knowledge meets requests. However, sensor networks are different from the Internet in that it is smaller in their scale and simpler in their structure.

The semantic web is a task force aimed to make web pages understandable by computers, so that websites can be searched in a standardized way. The potential benefits are that computers can harness the enormous network of information and services on the Web. The semantic web uses the descriptive technologies Resource Description Framework (RDF) and Web Ontology Language (OWL), and the data-centric, customizable Extensible Markup Language (XML), to address the machine-readability problem. These existing techniques help us on knowledge representation problem, but they cannot solve the propagation problem.

D. Regions Work

In my previous research, two projects are related to this proposal. One is my Master's Thesis [46], which is part of the Region Projects [47], [7]. In that work, I designed and implemented a special region mechanism for peer-to-peer systems to improve lookup and replication performance using autonomous system information [48]. The *regions* in that work can be viewed as a simplified prototype of the KP, since regions provide the applications with the underlying network topology information, but lacking in the sophisticated support that the KP provides.

In another research effort [23], I designed an efficient and scalable information aggregation mechanism for peer-to-peer networks. Aggregation is an important function in the KP. For example, fault diagnosis often requires aggregated knowledge from multiple agents. This work can be a base for the aggregation mechanism in the KP. However, the aggregation in the KP is more general and complex.

IX. Research Agenda

A. Remaining questions

This proposal covers many challenging problems. Although I have addressed some of them, many questions still need further exploration. The following is an incomplete list of questions to study further in my thesis:

1) What other knowledge should and can be provided by the NetKP? Policy discovery is mentioned, but it needs to be better defined.
2) Related to the first question but more generally, besides knowledge, what other mechanisms/utilities should be provided by the NetKP for sub-KPs and applications to organize themselves? For example, the NetKP could provide mechanisms to help construct a minimum-cost tree or disjoint trees, or provide nearest neighbor selection.
3) In the KP organization, when and how should a large AS be divided into several regions in the NetKP?
4) How to measure and share performance knowledge in the NetKP considering the tradeoff between accuracy and overhead?
5) Sub-KP organization needs to be explored further, especially on request propagation and aggregation.
6) How do we manage all dimensions of distances? How do we define interest difference?

B. Schedule

I plan to finish the thesis by December 2007. The attempted schedule is below:

1) November 2006 - February 2007. Design algorithms and concrete case studies.
2) March - June, 2007. Implement a prototype NetKP on PlanetLab, and two prototype sub-KPs.
3) July - August, 2007. Improve and evaluate the NetKP and sub-KPs.

4) September - November, 2007. Thesis writing.
5) December 2007. Thesis defense.

C. NetKP and Sub-KP Implementation

I will implement and deploy the network knowledge plane in the Internet. To build the NetKP, we need to be able to access network topology information, especially AS information. However, such information is not public, and special arrangement is needed with the ISPs so as to set up BGP sessions with routers. Furthermore, it is not possible to set up authoritative agents in all ASes to receive BGP feeds at the same time, and we need gradual deployment. To do so, we can learn BGP information indirectly from other sources, such as Route Views [49], and inference techniques [17]. Those BGP tables cover most of the ASes, but many peering links are missing. PlanetLab [50] and DETER [24] are two places to do a large-scale deployment of this prototype mechanism.

I have also performed preliminary experiments on intrusion detection on DETER. The initial results show KP's potential to improve detection speed. I plan to build more sub-KPs for DNS, network management, etc. To evaluate the effectiveness of the knowledge plane, I will examine the efficiency of connectivity graphs and quantify the benefits under certain metrics.

X. Summary

The ultimate goal of the knowledge plane is to build a new generation of network that can drive its own deployment and configuration, that can diagnose its own problems, and make decisions to resolve them. There are many challenging issues to achieve this goal, such as knowledge representation and utilization, trust and security, economic incentives, etc. As a step towards the knowledge plane, in this proposal I design an application-independent mechanism at the network layer to help construct application-specific connectivity graphs. According to the end-to-end arguments [51], only common and essential functions should be put into the network layer, while in this research I propose to add more functions to the network layer. I believe that as the Internet becomes increasingly pervasive, more and more applications need to learn more about network conditions to work correctly and efficiently besides end-to-end connectivity. Therefore, we need a common infrastructure to provide such network knowledge and mechanisms, at low cost, for applications, and hence it does not contradict the end-to-end arguments.

Agent organization and request propagation are two key issues in the knowledge plane. In my thesis, I will make the following contributions. First, I design a network knowledge plane as the application-independent mechanism. The application-independent NetKP provides common functions on network topology and performance, and help agents in sub-KPs find each other (agent discovery). I also address the knowledge base issues, and knowledge collection, dissemination, aggregation, etc.

Second, I propose to construct multiple sub-KPs on top of the NetKP, each of which concentrates on one area of interest. The agent organization and request propagation are different from those in the NetKP, and depends on both network knowledge and knowledge of specific sub-KPs.

Third, I plan to implement the proposed the NetKP and multiple sub-KPs in network management, and conduct case studies to understand and improve the knowledge plane.

By designing and implementing the NetKP and sub-KPs, and conducting several case studies on network management and applications, I hope to address the two problems, improve our understanding of the knowledge plane and motivate future research in this area.

References

[1] D. Clark, C. Partridge, J. C. Ramming, and J. Wroclawski, "A knowledge plane for the internet," in *Proceedings of ACM SIGCOMM 2003*, August 2003.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Symposium on Operating Systems Principles*, 2001, pp. 131–145.

[3] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Measurement and Modeling of Computer Systems*, 2000, pp. 1–12.

[4] Gnutella, "http://gnutella.wego.com." [Online]. Available: http://gnutella.wego.com

[5] J. Kulik, "Network monitoring and diagnosis based on available bandwidth measurement," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[6] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: A high performance publish-subscribe system for the world wide web," in *Networked System Design and Implementation*, 2006.

[7] K. Sollins, "Designing for scale and differentiation," in *ACM SIGCOMM 2003 FDNA Workshop*, 2003.

[8] D. Clark, "Content-directed aggregation and propagation in the knowledge plane," *Unpublished*, 2005.

[9] W.-O. W. W. Group, "http://www.w3.org/2001/sw/webont/."

[10] M. Wawrzoniak, L. Peterson, and T. Roscoe, "Sophia: An information plane for networked systems," 2003.

[11] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "Planetseer: Internet path failure monitoring and characterization in wide-area services," in *Proc. Sixth Symposium on Operating Systems Design and Implementation*, December 2004.

[12] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," *Proceedings of SIGCOMM'2001*, p. 13, 2001.

[13] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.

[14] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, 2004.

[15] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006.

[16] N. Lynch, *Distributed Algorithms*, 1st ed. San Francisco, California: Morgan Kaufmann, 1997.

[17] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Proc. of IEEE INFOCOM 2002, New York, NY*, Jun 2002.

[18] KaZaA, "http://www.kazaa.com." [Online]. Available: http://www.kazaa.com

[19] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM 2001*, 2001.

[21] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001, pp. 329–350.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," 2003.

[23] J. Li, K. Sollins, and D.-Y. Lim, "Implementing aggregation and broadcast over distributed hash tables," *ACM SIGCOMM Computer Communication Review*, vol. Volume 35, Number 1, January 2005.

[24] DETER, "http://www.isi.edu/deter/."

[25] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks." *Sci. Comput. Program.*, vol. 53, no. 2, pp. 165–194, 2004.

[26] ——, "Multi-domain diagnosis of end-to-end service failures in hierarchically routed networks." in *NETWORKING*, 2004, pp. 1036–1046.

[27] G. J. Lee, "Caprid: A common architecture for probablistic distributed internet fault diagnosis," *Ph.D. Thesis Proposal*, April 2006.

[28] D. J. Malan and M. D. Smith, "Host-based detection of worms through peer-to-peer cooperation," in *WORM '05: Proceedings of the 2005 ACM workshop on Rapid malcode*. New York, NY, USA: ACM Press, 2005, pp. 72–80.

[29] D. Dash, B. Kveton, J. M. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman, "When gossip is good: Distributed probabilistic inference for detection of slow network intrusions," in *Proceedings of the 21st National Conference on Artifi cial Intelligence*, 2006, pp. 1115–1122.

[30] S. G. Cheetancheri, J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, and E. M. Schooler, "A distributed host-based worm detection system," in *LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*. New York, NY, USA: ACM Press, 2006, pp. 107–113.

[31] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," in *IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.

[32] J. McAlerney, "An internet worm propagation data model," Master's thesis, University of California, Davis, September 2004.

[33] A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," in *SIGCOMM*, 2003.

[34] ——, "Scalable routing overlay networks," *SIGOPS Oper. Syst. Rev.*, 2006.

[35] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *ACM SIGCOMM*, 2002.

[36] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for p2p systems," 2002.

[37] M. Cai, M. Frank, B. Yan, and R.MacGregor, "A subscribable peer-to-peer rdf repository for distributed metadata management," in *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2005.
[38] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proc. of IEEE INFOCOM 2003*, 2003.
[39] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iplane: An information plane for distributed services," in *OSDI 2006*, November 2006.
[40] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind akamai (travelocity-based detouring)," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 435–446, 2006.
[41] N. Hu, "Network monitoring and diagnosis based on available bandwidth measurement," Ph.D. dissertation, Carnegie Mellon University, 2006.
[42] A. Adams, J. Mahdavi, M. Mathis, and V. Paxson, "Creating a scalable architecture for internet measurement," in *Proceedings of INET*, 1998.
[43] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UC Berkeley, Tech. Rep. UCB/CSD-01-1141, April 2001.
[44] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," in *USENIX Symposium on Internet Technologies and Systems*, march 2001, pp. 37–48.
[45] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, haystacks: Balancing push and pull for discovery in large-scale sensor networks," 2004.
[46] J. Li, "Improving application-level network services with regions," Master's thesis, MIT, September 2003.
[47] T. R. Project, "http://krs.lcs.mit.edu/regions." [Online]. Available: http://krs.lcs.mit.edu/regions
[48] J. Li and K. Sollins, "Exploiting autonomous system information in structured peer-to-peer networks," in *Proceedings of The 13th IEEE International Conference on Computer Communications and Networks (ICCCN2004)*, October 2004.
[49] T. R. V. Project, "http://www.routeviews.org/." [Online]. Available: University of Oregon
[50] T. PlanetLab, "http://www.planet-lab.org/."
[51] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.

APPENDIX

A. *Tables*

Symptoms	Causes
1. Local name cannot be resolved	Primary master Local host confi guration Slave server Caching-only server
2. Remote name cannot be resolved	Local name server Network connectivity Remote name server
3. Wrong or inconsistent answer	Local name server Remote name server
4. Lookups take a long time	Network connectivity Incorrect delegation
5. rlogin and rsh to host fails access check	local host name server
6. Access to services denied	Name server: case-sensitive names Incorrect delegation
7. Cannot get rid of old data	Parent name server: old delegation information gTLD server: registration of a non-name server Child name server: old cached data

TABLE II
SYMPTOMS AND CAUSES IN DNS FAULT DIAGNOSIS.