A Knowledge Plane for the Internet

David D. Clark*, Craig Partridge♦, J. Christopher Ramming† and John T. Wroclawski*

*M.I.T Lab for Computer Science
200 Technology Square
Cambridge, MA 02139
{ddc,jtw}@lcs.mit.edu

♦BBN Technologies
10 Moulton St
Cambridge, MA 02138
craig@bbn.com

†SRI International
333 Ravenswood Avenue
Menlo Park, CA 94205 USA
chrisramming@yahoo.com

We propose a new objective for network research: to build a fundamentally different sort of network that can assemble itself given high level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so.

We further argue that to achieve this goal, it is not sufficient to improve incrementally on the techniques and algorithms we know today. Instead, we propose a new construct, the Knowledge Plane, a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network. The knowledge plane is novel in its reliance on the tools of AI and cognitive systems. We argue that cognitive techniques, rather than traditional algorithmic approaches, are best suited to meeting the uncertainties and complexity of our objective.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *network communications.* C.2.3 [**Computer-Communication Networks**]: Network Operations – *network management, network monitoring.* C.2.6 [**Computer-Communication Networks**]: Internetworking.

General Terms

Management, Measurement, Design, Experimentation.

Keywords

Cognition; network applications; network configuration; knowledge plane.

1. INTRODUCTION

The Internet of today is a wonderful success. But success should not blind us to the Internet's limitations. Its emphasis on generality and heterogeneity, the 'narrow-hourglass' combination of a simple, transparent network with rich end-system functionality, and the deeply embedded assumption of a decentralized, multi-administrative structure are critical strengths, but lead to frustrated users when something fails, and high management overhead with much manual configuration, diagnosis and design.

Both user and operator frustrations arise from the same fundamental design principle of the Internet—the simple and transparent core with intelligence at the edges [1,2]. The network carries data without knowing what that data is, or what its purpose is. If some combination of events is keeping data from getting through, the edge may recognize that there is a problem, but the core cannot tell that something is wrong, because the core has no idea what *should* be happening. The edge understands applications, and what their expected behavior is; the core only deals with packets. Similarly, a network operator interacts with the core in very low-level terms such as per-router configuration of routes and policies. There is no way for the operator to express, or the network to model, what the high level goal of the operator is, and how the low-level decisions relate to that high level goal.

As we design a new sort of network, we must not lose the features of the Internet that have made it a success—its openness to new applications, the adaptability of its protocols, and the essential plasticity basic to its nature. Yet we must devise a technique that marries these virtues to a new goal: the ability of the network to know what it is being asked to do, so that it can more and more take care of itself, rather than depending on people to attend to it. If the network had a high-level view of its design goals and the constraints on acceptable configurations, then it could make many low-level decisions on its own. It could communicate with the network designer in terms of how well it met the goals, rather than by displaying a mass of router configuration tables. And it could deal with changes in the high level requirements by reconfiguring itself.

We argue that traditional, algorithmic approaches to adaptivity are unlikely to provide the required sophistication of behavior. The approach we take must offer the ability to and isolate high level goals from low level actions, to integrate and act on imperfect and conflicting information, and to learn from past actions to improve future performance. These properties are precisely those required to function effectively in the Internet's environment of diverse and competing objectives, decentralized control, complexity, and dynamic change.

This paper proposes an approach to network design based on tools from AI and cognitive systems. Specifically, we propose a construct,

SIGCOMM'03, August 25–29, 2003, Karlsruhe, Germany.

A Knowledge Plane for the Internet

David D. Clark*, Craig Partridge♦, J. Christopher Ramming† and John T. Wroclawski*

*M.I.T Lab for Computer Science
200 Technology Square
Cambridge, MA 02139
{ddc,jtw}@lcs.mit.edu

♦BBN Technologies
10 Moulton St
Cambridge, MA 02138
craig@bbn.com

†SRI International
333 Ravenswood Avenue
Menlo Park, CA 94205 USA
chrisramming@yahoo.com

We propose a new objective for network research: to build a fundamentally different sort of network that can assemble itself given high level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so.

We further argue that to achieve this goal, it is not sufficient to improve incrementally on the techniques and algorithms we know today. Instead, we propose a new construct, the Knowledge Plane, a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network. The knowledge plane is novel in its reliance on the tools of AI and cognitive systems. We argue that cognitive techniques, rather than traditional algorithmic approaches, are best suited to meeting the uncertainties and complexity of our objective.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**] Network Architecture and Design – *network communications.* C.2.3 [**Computer-Communication Networks**]: Network Operations – *network management network monitoring.* C.2.6 [**Computer-Communication Networks**]: Internetworking.

General Terms

Management, Measurement, Design, Experimentation.

Keywords

Cognition; network applications network configuration; knowledge plane.

1. INTRODUCTION

The Internet of today is a wonderful success. But success should not blind us to the Internet"s limitations. Its emphasis on generality and heterogeneity, the 'narrow-hourglass' combination of a simple, transparent network with rich end-system functionality, and the deeply embedded assumption of a decentralized, multi-administrative structure are critical strengths, but lead to frustrated users when something fails, and high management overhead with much manual configuration, diagnosis and design.

Both user and operator frustrations arise from the same fundamental design principle of the Internet—the simple and transparent core with intelligence at the edges [1,2]. The network carries data without knowing what that data is, or what its purpose is. If some combination of events is keeping data from getting through, the edge may recognize that there is a problem, but the core cannot tell that something is wrong, because the core has no idea what *should* be happening. The edge understands applications, and what their expected behavior is; the core only deals with packets. Similarly, a network operator interacts with the core in very low-level terms such as per-router configuration of routes and policies. There is no way for the operator to express, or the network to model, what the high level goal of the operator is, and how the low-level decisions relate to that high level goal.

As we design a new sort of network, we must not lose the features of the Internet that have made it a success—its openness to new applications, the adaptability of its protocols, and the essential plasticity basic to its nature. Yet we must devise a technique that marries these virtues to a new goal: the ability of the network to know what it is being asked to do, so that it can more and more take care of itself, rather than depending on people to attend to it. If the network had a high-level view of its design goals and the constraints on acceptable configurations, then it could make many low-level decisions on its own. It could communicate with the network designer in terms of how well it met the goals, rather than by displaying a mass of router configuration tables. And it could deal with changes in the high level requirements by reconfiguring itself.

We argue that traditional, algorithmic approaches to adaptivity are unlikely to provide the required sophistication of behavior. The approach we take must offer the ability to and isolate high level goals from low level actions, to integrate and act on imperfect and conflicting information, and to learn from past actions to improve future performance. These properties are precisely those required to function effectively in the Internet"s environment of diverse and competing objectives, decentralized control, complexity, and dynamic change.

This paper proposes an approach to network design based on tools from AI and cognitive systems. Specifically, we propose a construct,

SIGCOMM'03, August 25–29, 2003, Karlsruhe, Germany.

a distributed cognitive system that permeates the network, that we call the **knowledge plane**.

The rest of this paper is organized as follows. Section 2 describes the concept of the knowledge plane. It contrasts this concept with alternatives, and argues for the cognitive approach. Section 3 is a discussion of what this construct might do for us—examples of how it can make networking better. Section 4 discusses some important design constraints and considerations for a knowledge plane architecture. Section 5 outlines the key challenges in our path.

2. A Proposal: the Knowledge Plane

The discussion above hints at a solution in which the network has a high-level view of what its purpose is—the goals of its designers, of the applications running on it, and of its users. In an application-specific network, one approach might be to utilize and embed such domain-specific knowledge in the core design of the network, as was done in the telephone network. But this defeats a fundamental objective of the Internet – its ability to host a broad and changing array of applications. Rather than pleasing no one by adding "just a little" application knowledge to the Internet's simple and transparent data transport plane, a better alternative is to devise a separate construct that creates, reconciles and maintains the many aspects of a high-level view, and then provides services and advice as needed to other elements of the network. This is the knowledge plane, or KP.

Understanding the precise best path towards this goal is a matter of significant research, and this paper neither can nor does propose a complete technical description of the knowledge plane. As a start, however, we sketch certain attributes potentially central to the success of a knowledge plane, and consider how this perspective differs from today's practice. These include:

Edge involvement: The end-to-end principle suggests that much valuable information about network performance originates not in the network, but in the devices and applications that use it. This is an inevitable and desirable consequence of the Internet's general-purpose data plane. It implies, however, that much of the "knowledge" in the knowledge plane may be produced, managed, and consumed at *or beyond* the "traditional" edge of the network. The reach of the knowledge plane is broader than that of traditional network management.

Global perspective: Most management systems are regional — each operator manages the part he owns. But truly useful problem identification may depend on correlation of observations from different parts of the network. Not only must data from the edges be combined with data from "inside" the network, but data from different parts of the network may be needed to fully comprehend a sequence of events. The knowledge plane would, ideally, be able to extend its perspective to the entire global network as required.

Compositional structure: If the reach of the KP is global, at the same time it must be designed to take account of what we may loosely call "compositional" considerations. A most basic example is that the KPs of two unconnected networks should be capable of merging their perspective and activities if the networks become connected.

A corollary of the composition problem is the need to operate in the presence of imperfect and conflicting information: some regions will desire to keep date private. Mutual distrust among some network operators and service providers, and indeed, among any parties that jockey for economic advantage, leads directly to today's need for highly skilled human reasoning to deduce and model network behavior. The KP faces a similar problem: it cannot assume a homogeneous network of shared objectives and shared information.

Unified approach: One might speculate that the various problems we aim to address could most easily be solved by distinct mechanisms, working bottom up, perhaps loosely tied together at the top. In contrast, the KP as we conceive it is a single, unified system, with common standards and a common framework for "knowledge". This unified approach is needed because real world knowledge is not strictly partitioned by task. We suggest that the knowledge plane should be structured similarly, based on the *knowledge*, not the task. We believe that while point solutions may be easier to develop, an integrated approach will be substantially more effective in the long run.

Cognitive framework: The knowledge plane needs to make judgments in the presence of partial or conflicting information; to recognize and mediate conflicts in policies and goals; to respond to problems and attacks in better-than-human time frames; to perform optimizations in high-dimensional environments that are too complicated to be addressed by humans or analytical solutions; and to automate functions that today must be carried out by rare and highly skilled network technicians. We therefore expect cognitive techniques to serve as the foundation of the knowledge plane: representation, learning, and reasoning that allow the knowledge plane to be "aware" of the network and its actions in the network.

We turn now to further discussion of three ideas key to our position: the necessity of a new construct, the desirability of a unified knowledge plane, and the value of cognitive tools.

2.1 Why a New Construct?

Most discussions of network architecture recognize two architectural divisions, or planes: a data plane, over which content is forwarded, and a control or management plane, which is used to direct, measure, and repair the data plane. By talking of a "knowledge plane" we are saying a fundamentally new construct is required, rather than fitting knowledge into an existing plane (presumably the management plane). Why do we believe a new construct is required?

If we look at the two existing planes, we find two radically different structures. The data plane (in almost any notable data transport architecture) uses some form of layering to hide complexity and to enable extensibility, interoperability and scalability. In contrast the control and management system is invariably designed to cut across the layering, giving visibility and access to all the aspects of the network, which must be monitored and managed. And, indeed, because the management plane is all-seeing, it tends to scale poorly and to be hard to change.

The knowledge plane clearly sits in a different place. Since it doesn't move data directly, it is not the data plane. And unlike the management plane it tends to break down boundaries to provide a unified view, rather than partition the world into managed enclaves. It is functionally unlike the management plane as well – it is hard to envision the KP managing accounting records (reading them occasionally, perhaps, but collecting, storing and processing them, no).

2.2 Why a Unified Approach?

Consider the example of a user trying to install a new application and discovering that it does not work. One reason might be that the ISP of the user has blocked that class of traffic. For the KP to give

a distributed cognitive system that permeates the network, that we call the **knowledge plane**.

The rest of this paper is organized as follows. Section 2 describes the concept of the knowledge plane. It contrasts this concept with alternatives, and argues for the cognitive approach. Section 3 is a discussion of what this construct might do for us—examples of how it can make networking better. Section 4 discusses some important design constraints and considerations for a knowledge plane architecture. Section 5 outlines the key challenges in our path.

2. A Proposal: the Knowledge Plane

The discussion above hints at a solution in which the network has a high-level view of what its purpose is—the goals of its designers, of the applications running on it, and of its users. In an application-specific network, one approach might be to utilize and embed such domain-specific knowledge in the core design of the network, as was done in the telephone network. But this defeats a fundamental objective of the Internet – its ability to host a broad and changing array of applications. Rather than pleasing no one by adding "just a little" application knowledge to the Internet's simple and transparent data transport plane, a better alternative is to devise a separate construct that creates, reconciles and maintains the many aspects of a high-level view, and then provides services and advice as needed to other elements of the network. This is the knowledge plane, or KP.

Understanding the precise best path towards this goal is a matter of significant research, and this paper neither can nor does propose a complete technical description of the knowledge plane. As a start, however, we sketch certain attributes potentially central to the success of a knowledge plane, and consider how this perspective differs from today's practice. These include:

Edge involvement: The end-to-end principle suggests that much valuable information about network performance originates not in the network, but in the devices and applications that use it. This is an inevitable and desirable consequence of the Internet's general-purpose data plane. It implies, however, that much of the "knowledge" in the knowledge plane may be produced, managed, and consumed at *or beyond* the "traditional" edge of the network. The reach of the knowledge plane is broader than that of traditional network management.

Global perspective: Most management systems are regional — each operator manages the part he owns. But truly useful problem identification may depend on correlation of observations from different parts of the network. Not only must data from the edges be combined with data from "inside" the network, but data from different parts of the network may be needed to fully comprehend a sequence of events. The knowledge plane would, ideally, be able to extend its perspective to the entire global network as required.

Compositional structure: If the reach of the KP is global, at the same time it must be designed to take account of what we may loosely call "compositional" considerations. A most basic example is that the KPs of two unconnected networks should be capable of merging their perspective and activities if the networks become connected.

A corollary of the composition problem is the need to operate in the presence of imperfect and conflicting information: some regions will desire to keep date private. Mutual distrust among some network operators and service providers, and indeed, among any parties that jockey for economic advantage, leads directly to today's need for highly skilled human reasoning to deduce and model network behavior. The KP faces a similar problem: it cannot assume a homogeneous network of shared objectives and shared information.

Unified approach: One might speculate that the various problems we aim to address could most easily be solved by distinct mechanisms, working bottom up, perhaps loosely tied together at the top. In contrast, the KP as we conceive it is a single, unified system, with common standards and a common framework for "knowledge". This unified approach is needed because real world knowledge is not strictly partitioned by task. We suggest that the knowledge plane should be structured similarly, based on the *knowledge*, not the task. We believe that while point solutions may be easier to develop, an integrated approach will be substantially more effective in the long run.

Cognitive framework: The knowledge plane needs to make judgments in the presence of partial or conflicting information; to recognize and mediate conflicts in policies and goals; to respond to problems and attacks in better-than-human time frames; to perform optimizations in high-dimensional environments that are too complicated to be addressed by humans or analytical solutions; and to automate functions that today must be carried out by rare and highly skilled network technicians. We therefore expect cognitive techniques to serve as the foundation of the knowledge plane: representation, learning, and reasoning that allow the knowledge plane to be "aware" of the network and its actions in the network.

We turn now to further discussion of three ideas key to our position: the necessity of a new construct, the desirability of a unified knowledge plane, and the value of cognitive tools.

2.1 Why a New Construct?

Most discussions of network architecture recognize two architectural divisions, or planes: a data plane, over which content is forwarded, and a control or management plane, which is used to direct, measure, and repair the data plane. By talking of a "knowledge plane" we are saying a fundamentally new construct is required, rather than fitting knowledge into an existing plane (presumably the management plane). Why do we believe a new construct is required?

If we look at the two existing planes, we find two radically different structures. The data plane (in almost any notable data transport architecture) uses some form of layering to hide complexity and to enable extensibility, interoperability and scalability. In contrast the control and management system is invariably designed to cut across the layering, giving visibility and access to all the aspects of the network, which must be monitored and managed. And, indeed, because the management plane is all-seeing, it tends to scale poorly and to be hard to change.

The knowledge plane clearly sits in a different place. Since it doesn't move data directly, it is not the data plane. And unlike the management plane it tends to break down boundaries to provide a unified view, rather than partition the world into managed enclaves. It is functionally unlike the management plane as well – it is hard to envision the KP managing accounting records (reading them occasionally, perhaps, but collecting, storing and processing them, no).

2.2 Why a Unified Approach?

Consider the example of a user trying to install a new application and discovering that it does not work. One reason might be that the ISP of the user has blocked that class of traffic. For the KP to give

the most effective feedback to each party, it needs access to the configuration constraints set by the ISP, so it can determine the rules behind the blocking and tell the user what this implies. So it is necessary that the information about network configuration and about user-observed problems be in the same framework.

A related example concerns overlay network such as CDNs. It is easy to imagine that one component of the KP is topology and performance information that a CDN could use to position its delivery nodes "close" to users. This information could come from a diversity of sources such as "network weather" services, user-reported experience, and ISPs, and would include not just traffic measurements, but information about administrative traffic restrictions and local firewall restrictions (perhaps the "users" can't receive certain types of content). The interested parties (users, CDNs) benefit from having this information integrated and presented in a consolidated form.

There are some cases where the KP may be able to resolve a problem on its own. If it discovers that the reason for a problem is a low-level decision that is not material to the high-level goals of the operators, it might change the decision. But to determine if a change is appropriate, KP needs access to the reasoning behind the setting. So the knowledge about planning needs to be in the same context as the repair problem.

When a component of the network makes a low-level observation about a possible anomaly, it has no idea what the relevance actually is. This observation might trigger a repair, a reconfiguration, a notification to a network operator in a distant part of the network, a security alert, or something else quite different. So observations on network conditions cannot be thought of as being a part of one problem space, but instead as being a part of the KP.

We recognize that point solutions to specific problems may get part of the way more rapidly that the general solution postulated here. But the core of our hypothesis is that to get to the final goal: a network that can configure itself, that can explain itself, that can repair itself, and does not confound the user with mysteries, the approach based on the combination of point solutions will not succeed.

2.3 Why a Cognitive System?

Our objectives for the Knowledge Plane require it to meet a number of significant challenges:

- It must function usefully in the presence of incomplete, inconsistent, and possibly misleading or malicious information. System failures, information filtering for privacy or competitive reasons, and finite network resources are just some of the forces conspiring to create this requirement.
- It must perform appropriately in the presence of conflicting or inconsistent higher-level goals among the Internet's different stakeholders. This is a manifestation of the *tussle* dilemma discussed in [12].
- It must operate effectively in the face of generality, including the introduction of new technologies and applications not conceived of at the time of its design, and in the face of a highly dynamic environment, including both short-term and long-term changes in the structure and complexity of the underlying network.

We hypothesize that these challenges cannot be met by analytical solutions, because analytical solutions generally require complete information, precise problem formulations, and a relatively static operating environment. Instead, we suggest that "cognitive" techniques will be needed. The key benefit of these techniques is their potential to perform effectively, and to evaluate and improve their own performance, in the presence of complex, inconsistent, dynamic, and evolving environments. We discuss two defining characteristics of a cognitive knowledge plane.

First, the KP must eventually "close the loop" on the network as does an ordinary control system. As we gain experience and trust, the knowledge plane will first enable a *recognize-explain* cycle, then a *recognize-explain-suggest* cycle, and ultimately a *recognize-act* cycle for many management tasks. Because the knowledge plane must be more general and flexible than standard control systems, we look elsewhere for additional inspiration. Architectures inspired by theories of human cognition [18] have achieved some successes and hint at one approach. In the knowledge plane context, a cognitive architecture would of course be distributed and decentralized, and the partitioning would be effected in part to support divergent interests of network stakeholders.

Second, the KP must be able to learn and reason. Learning is the principled accumulation of knowledge, and can take place through many means: by trial and error, by instruction, by generalization, by analogy, through problem solving and mental search, and more. Some learning approaches require human involvement, and some do not. In a static problem environment, one simple enough to admit of analytic solution, learning is irrelevant. But IP networks, by design and intent, are constantly evolving in many dimensions, and are infinite in potential configurations. To the extent possible, when new situations are recognized or new actions performed and evaluated, the knowledge plane should improve: its knowledge base should grow in useful ways. The first and most immediate challenge of learning is to model the behavior, dependencies, and requirements of applications through the obscuring veil of our existing transparent data plane.

Reasoning involves the composition of existing knowledge to draw new inferences and beliefs. Reasoning processes can translate declarative knowledge (whether handcrafted or learned) into interpretations of observations and decisions about actions. If we wish the network of the future to support high-level goals and constraints, we will need reasoning methods that can operate on these ions.

In the long run, an interesting and important function of reasoning in the knowledge plane will be to support mediation between users and operators whose goals may conflict with each other and/or with fixed design constraints. The inevitability of such conflicts suggests that we must develop new techniques for representing and reasoning about constraints and policies. Initially, these representations will need to be inferred from low-level configurations and actions, but the ultimate goal is to express goals and policies at a high level and use those to generate low-level configurations.

Even in the short run we can bring to bear a great deal of existing research on the design and construction of a knowledge plane. Experience with cognitive architectures [18], recent work in multi-agent systems [22], and the emerging field of algorithmic game theory may prove directly useful. However, the networking context also raises many challenges that will stretch the current state of cognitive systems and redirect research in new and intriguing ways [19,20].

the most effective feedback to each party, it needs access to the configuration constraints set by the ISP, so it can determine the rules behind the blocking and tell the user what this implies. So it is necessary that the information about network configuration and about user-observed problems be in the same framework.

A related example concerns overlay network such as CDNs. It is easy to imagine that one component of the KP is topology and performance information that a CDN could use to position its delivery nodes "close" to users. This information could come from a diversity of sources such as "network weather" services, user-reported experience, and ISPs, and would include not just traffic measurements, but information about administrative traffic restrictions and local firewall restrictions (perhaps the "users" can't receive certain types of content). The interested parties (users, CDNs) benefit from having this information integrated and presented in a consolidated form.

There are some cases where the KP may be able to resolve a problem on its own. If it discovers that the reason for a problem is a low-level decision that is not material to the high-level goals of the operators, it might change the decision. But to determine if a change is appropriate, KP needs access to the reasoning behind the setting. So the knowledge about planning needs to be in the same context as the repair problem.

When a component of the network makes a low-level observation about a possible anomaly, it has no idea what the relevance actually is. This observation might trigger a repair, a reconfiguration, a notification to a network operator in a distant part of the network, a security alert, or something else quite different. So observations on network conditions cannot be thought of as being a part of one problem space, but instead as being a part of the KP.

We recognize that point solutions to specific problems may get part of the way more rapidly that the general solution postulated here. But the core of our hypothesis is that to get to the final goal: a network that can configure itself, that can explain itself, that can repair itself, and does not confound the user with mysteries, the approach based on the combination of point solutions will not succeed.

2.3 Why a Cognitive System?

Our objectives for the Knowledge Plane require it to meet a number of significant challenges:

- It must function usefully in the presence of incomplete, inconsistent, and possibly misleading or malicious information. System failures, information filtering for privacy or competitive reasons, and finite network resources are just some of the forces conspiring to create this requirement.
- It must perform appropriately in the presence of conflicting or inconsistent higher-level goals among the Internet's different stakeholders. This is a manifestation of the *tussle* dilemma discussed in [12].
- It must operate effectively in the face of generality, including the introduction of new technologies and applications not conceived of at the time of its design, and in the face of a highly dynamic environment, including both short-term and long-term changes in the structure and complexity of the underlying network.

We hypothesize that these challenges cannot be met by analytical solutions, because analytical solutions generally require complete information, precise problem formulations, and a relatively static operating environment. Instead, we suggest that "cognitive" techniques will be needed. The key benefit of these techniques is their potential to perform effectively, and to evaluate and improve their own performance, in the presence of complex, inconsistent, dynamic, and evolving environments. We discuss two defining characteristics of a cognitive knowledge plane.

First, the KP must eventually "close the loop" on the network as does an ordinary control system. As we gain experience and trust, the knowledge plane will first enable a *recognize-explain* cycle, then a *recognize-explain-suggest* cycle, and ultimately a *recognize-act* cycle for many management tasks. Because the knowledge plane must be more general and flexible than standard control systems, we look elsewhere for additional inspiration. Architectures inspired by theories of human cognition [18] have achieved some successes and hint at one approach. In the knowledge plane context, a cognitive architecture would of course be distributed and decentralized, and the partitioning would be effected in part to support divergent interests of network stakeholders.

Second, the KP must be able to learn and reason. Learning is the principled accumulation of knowledge, and can take place through many means: by trial and error, by instruction, by generalization, by analogy, through problem solving and mental search, and more. Some learning approaches require human involvement, and some do not. In a static problem environment, one simple enough to admit of analytic solution, learning is irrelevant. But IP networks, by design and intent, are constantly evolving in many dimensions, and are infinite in potential configurations. To the extent possible, when new situations are recognized or new actions performed and evaluated, the knowledge plane should improve: its knowledge base should grow in useful ways. The first and most immediate challenge of learning is to model the behavior, dependencies, and requirements of applications through the obscuring veil of our existing transparent data plane.

Reasoning involves the composition of existing knowledge to draw new inferences and beliefs. Reasoning processes can translate declarative knowledge (whether handcrafted or learned) into interpretations of observations and decisions about actions. If we wish the network of the future to support high-level goals and constraints, we will need reasoning methods that can operate on these ions.

In the long run, an interesting and important function of reasoning in the knowledge plane will be to support mediation between users and operators whose goals may conflict with each other and/or with fixed design constraints. The inevitability of such conflicts suggests that we must develop new techniques for representing and reasoning about constraints and policies. Initially, these representations will need to be inferred from low-level configurations and actions, but the ultimate goal is to express goals and policies at a high level and use those to generate low-level configurations.

Even in the short run we can bring to bear a great deal of existing research on the design and construction of a knowledge plane. Experience with cognitive architectures [18], recent work in multi-agent systems [22], and the emerging field of algorithmic game theory may prove directly useful. However, the networking context also raises many challenges that will stretch the current state of cognitive systems and redirect research in new and intriguing ways [19,20].

3. What is the Knowledge Plane Good For?

At a high level, we proposed a unified goal for the KP: build a new generation of network by allowing it to have a view of what it supposed to be, and what it is supposed to be doing. To achieve this goal, there are more specific problem domains to be supported. Here we discuss in more detail some of them.

Fault diagnosis and mitigation: Today, when some part of the Internet fails, it is almost impossible for the end user to tell what has happened, to figure out who should be notified, or what to do to correct the fault. If we take the Internet of today as the starting point, it is appealing to imagine a command that a user can run to demand an explanation when something seems to be broken. This is the WHY(problem-x) command: why is x broken? So, for instance, the user might ask, "Why can't I get to www.acm.org?"

However, the WHY formulation is not bold enough. An over-bold alternative would be that if the KP is smart enough, the network should never fail. In this case, there is no need for WHY. But this ambition is fundamentally flawed. In some cases, only a human knows enough to determine if what is happening is actually a fault. When Dave unplugs his laptop and puts it in his briefcase, there may be some applications that suddenly stop working, but this is not a fault. It is what Dave intended, and if some semi-smart KP wakes up each time he disconnects his laptop and asks if he wants to reconnect it, this is a nightmare, not a success. So there will be times when only a person can give the KP guidance. Instead of WHY(problem-x), this is FIX(problem-x). The user is saying that something is broken, and make it right.

Is this enough guidance that the KP can correct what is wrong? In fact, the interesting examples are when the "problem" is caused by conflicting specifications or constraints that come from different people. One may say FIX(this game I just installed that does not work), and the reason it is failing is that the ISP has blocked that game. One may say FIX(lousy bandwidth) but the problem is that one has exceeded one's usage quota and the ISP is rate-limiting. These are cases where the KP may not be able to resolve the matter. What we might strive for, however, is a KP that can either resolve a problem or say why not. So one answer to FIX(problem-x) may be CANNOT(reason-y). And if the system does fix something, it may want to tell a person that this happened, in case there is a further action that only a person can take.

This example suggests that the interaction between the user and the KP is bi-directional and expressive. And of course, the KP may communicate with many entities about a problem. The demand from a user FIX(broken-game) might trigger a message back to the user that the game is blocked, but might also trigger a message to the ISP that it has an unhappy user.

A further extension of this story is that the KP can provide an assistant for user and managers, an agent that watches what the people do, and learns over time what is normal and what is not. So a KP agent on Dave's laptop might learn that Dave unplugs it all the time, while an agent on Dave's desktop machine might realize that he never disconnects it, and risk bothering Dave to ask if he meant to do that. In this way, the problem of fault diagnosis and mitigation has a learning component.

Once the FIX(problem-x) function has been implemented in the KP, programs as well as people can use it. As the user's agent learns, it should more and more often give this signal on its own. And other programs, such as application code, may detect and signal that something is wrong. The KP will have to decide how much credence to give these signals, depending on where they come from.

Behind the scenes, the FIX command will trigger a range of activities in the KP. The FIX command would start with a local component that runs on the user's machine, and then exchanges information with the KP to figure out what is wrong. The diagnostics can check out functions at all levels, from packet forwarding to application function. There are several current research projects that this application could build on [13,14].

Once the end node has performed what diagnosis it can, the next stage is for the tool to add assertions to the shared knowledge plane about what it has discovered, and ask the KP for relevant information. This contribution to the knowledge plane allows all the users on the network collectively to build a global view of network and service status. This data can be combined with information derived from measurement efforts now going on across the Internet that attempt to build an overall model of network status [9,15]. Such aggregation is important if the failure is one that affects lot of users.

Automatic (re)configuration: The dynamic routing of the original Internet did not take into account administrative and policy constraints, so routing today is more and more defined by static policy tables. This means that devices such as routers are increasingly manually configured and managed. Static tables and manual configuration make the network brittle to failure, hard to change, and even harder to reason about globally. Imagine, as part of the KP, a configuration manager for a region of the Internet, which would accept high-level assertions about how the components of a network are supposed to arrange themselves, and guide the actual detailed configuration accordingly. Examples include controlling the deployment of a consumer network in the home, an ad hoc network in support of a rapid deployment force, or a network for a small business. Successful accomplishment of this project could lead to substantial reductions in manpower needed to configure and operate networks.

The KP configuration manager should have enough understanding of low-level structure to detect if the network is properly configured according to the high-level constraints, to detect if a better configuration alternative is available, and to detect if the system appears to be corrupted. The reasoning must go in both directions. That is, the manager must be able to derive low-level settings given high-level goals, priorities and constraints, and it must be able to look at existing low-level settings and describe the resulting behavior in the high-level terms.

Again, the interesting problem (once we get the basic idea to work) is when the system encounters conflicting assertions made by different parties. The network manager might say ROUTING_PREFERENCE(low-cost links), and an end-user's machine might say FIX(low bandwidth). Again, the KP may be able to resolve some of these problems, and might learn over time when it is safe for it to act on its own, and when it must kick the problem back to the relevant humans in meaningful terms. (This example, by the way, illustrates why the KP must be seen as a unified system, not as separate systems for fault management and for configuration.)

The configuration task is not something that happens once at the turn-up of the network. It should be something that is happening constantly, looking at changing network conditions, application demands, and changing constraints. It is also a task that can run "recursively". A global network is not built top down. It is built bottom up, region by region. Each region will first configure itself

3. What is the Knowledge Plane Good For?

At a high level, we proposed a unified goal for the KP: build a new generation of network by allowing it to have a view of what it supposed to be, and what it is supposed to be doing. To achieve this goal, there are more specific problem domains to be supported. Here we discuss in more detail some of them.

Fault diagnosis and mitigation: Today, when some part of the Internet fails, it is almost impossible for the end user to tell what has happened, to figure out who should be notified, or what to do to correct the fault. If we take the Internet of today as the starting point, it is appealing to imagine a command that a user can run to demand an explanation when something seems to be broken. This is the WHY(problem-x) command: why is x broken? So, for instance, the user might ask, "Why can't I get to www.acm.org?"

However, the WHY formulation is not bold enough. An over-bold alternative would be that if the KP is smart enough, the network should never fail. In this case, there is no need for WHY. But this ambition is fundamentally flawed. In some cases, only a human knows enough to determine if what is happening is actually a fault. When Dave unplugs his laptop and puts it in his briefcase, there may be some applications that suddenly stop working, but this is not a fault. It is what Dave intended, and if some semi-smart KP wakes up each time he disconnects his laptop and asks if he wants to reconnect it, this is a nightmare, not a success. So there will be times when only a person can give the KP guidance. Instead of WHY(problem-x), this is FIX(problem-x). The user is saying that something is broken, and make it right.

Is this enough guidance that the KP can correct what is wrong? In fact, the interesting examples are when the "problem" is caused by conflicting specifications or constraints that come from different people. One may say FIX(this game I just installed that does not work), and the reason it is failing is that the ISP has blocked that game. One may say FIX(lousy bandwidth) but the problem is that one has exceeded one's usage quota and the ISP is rate-limiting. These are cases where the KP may not be able to resolve the matter. What we might strive for, however, is a KP that can either resolve a problem or say why not. So one answer to FIX(problem-x) may be CANNOT(reason-y). And if the system does fix something, it may want to tell a person that this happened, in case there is a further action that only a person can take.

This example suggests that the interaction between the user and the KP is bi-directional and expressive. And of course, the KP may communicate with many entities about a problem. The demand from a user FIX(broken-game) might trigger a message back to the user that the game is blocked, but might also trigger a message to the ISP that it has an unhappy user.

A further extension of this story is that the KP can provide an assistant for user and managers, an agent that watches what the people do, and learns over time what is normal and what is not. So a KP agent on Dave's laptop might learn that Dave unplugs it all the time, while an agent on Dave's desktop machine might realize that he never disconnects it, and risk bothering Dave to ask if he meant to do that. In this way, the problem of fault diagnosis and mitigation has a learning component.

Once the FIX(problem-x) function has been implemented in the KP, programs as well as people can use it. As the user's agent learns, it should more and more often give this signal on its own. And other programs, such as application code, may detect and signal that something is wrong. The KP will have to decide how much credence to give these signals, depending on where they come from.

Behind the scenes, the FIX command will trigger a range of activities in the KP. The FIX command would start with a local component that runs on the user's machine, and then exchanges information with the KP to figure out what is wrong. The diagnostics can check out functions at all levels, from packet forwarding to application function. There are several current research projects that this application could build on [13,14].

Once the end node has performed what diagnosis it can, the next stage is for the tool to add assertions to the shared knowledge plane about what it has discovered, and ask the KP for relevant information. This contribution to the knowledge plane allows all the users on the network collectively to build a global view of network and service status. This data can be combined with information derived from measurement efforts now going on across the Internet that attempt to build an overall model of network status [9,15]. Such aggregation is important if the failure is one that affects lot of users.

Automatic (re)configuration: The dynamic routing of the original Internet did not take into account administrative and policy constraints, so routing today is more and more defined by static policy tables. This means that devices such as routers are increasingly manually configured and managed. Static tables and manual configuration make the network brittle to failure, hard to change, and even harder to reason about globally. Imagine, as part of the KP, a configuration manager for a region of the Internet, which would accept high-level assertions about how the components of a network are supposed to arrange themselves, and guide the actual detailed configuration accordingly. Examples include controlling the deployment of a consumer network in the home, an ad hoc network in support of a rapid deployment force, or a network for a small business. Successful accomplishment of this project could lead to substantial reductions in manpower needed to configure and operate networks.

The KP configuration manager should have enough understanding of low-level structure to detect if the network is properly configured according to the high-level constraints, to detect if a better configuration alternative is available, and to detect if the system appears to be corrupted. The reasoning must go in both directions. That is, the manager must be able to derive low-level settings given high-level goals, priorities and constraints, and it must be able to look at existing low-level settings and describe the resulting behavior in the high-level terms.

Again, the interesting problem (once we get the basic idea to work) is when the system encounters conflicting assertions made by different parties. The network manager might say ROUTING_PREFERENCE(low-cost links), and an end-user's machine might say FIX(low bandwidth). Again, the KP may be able to resolve some of these problems, and might learn over time when it is safe for it to act on its own, and when it must kick the problem back to the relevant humans in meaningful terms. (This example, by the way, illustrates why the KP must be seen as a unified system, not as separate systems for fault management and for configuration.)

The configuration task is not something that happens once at the turn-up of the network. It should be something that is happening constantly, looking at changing network conditions, application demands, and changing constraints. It is also a task that can run "recursively". A global network is not built top down. It is built bottom up, region by region. Each region will first configure itself

using its locally specified goals and constraints. But when two regions then connect, there may be further constraints that one imposes on the other. So a provider network might say to a subscriber network: NO_MULTICAST. This might cause the subscriber network to change some of its internal organization, disable some end-user applications, and so on.

Support for overlay networks: If the KP has enough information to configure the network itself, that information can also be useful to applications that are configuring themselves. For example, we are increasingly seeing the development of application-specific overlay networks on the Internet. Each overlay network uses edge-based mechanisms to evaluate the performance of different possible paths through the Internet, and seeks to build a set of transport paths that effectively route application packets through what appears to be the part of the Internet best suited to the application's needs. Currently, application networks must probe the Internet, because there is no mechanism for them to learn about the capabilities of the network core. The KP would be in a position to aggregate application- and network-derived knowledge about network performance, offer applications better information about the network than they could learn by probing, and access to control points whose behavior could be modified to help better meet the applications' needs. The KP thus enables per-application control over traffic without the need to build per-application infrastructure throughout the network.

Knowledge-enhanced intrusion detection: There are a number of projects (and a number of products) that perform some sort of analysis to detect network intrusions. In general they look for patterns in data observed somewhere in the network. The current generation of these tools trigger both false positives and false negatives. It has been hypothesized that a next generation of tools for intrusion detection will require that observations from several points in the network will have to be correlated, in order to get a more robust and useful signal. The development of the knowledge plane provides a basis to implement this data gathering and correlation.

4. Knowledge Plane Architecture

Previous sections of this paper have outlined the goals we set for a knowledge plane. In this section, we consider aspects of its system structure. Our discussion is speculative: any successful KP architecture will be shaped by a number of requirements and constraints, not all of which are apparent today. At its highest level the architecture of the KP will be shaped heavily by two broad forces: its distributed, compositional structure, and its multi-scale, potentially global knowledge perspective.

Our ultimate objective is that networked systems should organize themselves, under the constraints and guidance of external inputs, to meet the goals of their stakeholders. Even in the near term, the KP must respect and build on the fact that networks have internal structure and dynamics -- large networks are composed by interconnecting smaller ones, participants come and go, and relationships between the owners, operators and users of different networks may change even when the physical structure does not.

This implies that the knowledge plane serving a network is not a globally engineered entity, but is instead an autonomously created structure that is recursively, dynamically, and continuously composing and decomposing itself from smaller sub-planes. This requirement argues that the KP:

- Is distributed - KP functionality for different regions of the network is physically and logically decentralized.
- Is bottom up - simple entities (e.g. web servers) can compose into larger, more complex entities (e.g. a web farm) as needed, and decompose themselves from the system as appropriate. This is a recursive process, that may proceed on many levels.
- Is constraint driven - the basic principle is that the system can, and may, adopt (or not) any behavior that is not specifically constrained.
- Moves from simple to complex. Speaking generally, the act of composing a set of networks to form a larger one places more requirements or constraints on the behavior of each network. A trivial example would be that a standalone IP network can use a wide range of addresses, but connecting it to a larger network constrains the range of options in this regard.

Our first objective for the KP system architecture is that it support this distributed, compositional perspective, providing the necessary enabling ions and capabilities.

In contrast with the distributed *organization* of the KP, we have argued in previous sections of this paper that KP may often benefit from taking a global *perspective* - integrating observations and conclusions from many points in the network. Key implications of this are that:

- Data and knowledge integration is a central function of the KP. The KP must be able to collect, filter, reduce, and route observations, assertions, and conclusions from different parts of the network to points where they are useful.
- The KP must operate successfully in the presence of imperfect information. Because this global perspective is both physically large and spans multiple administrative entities, the cognitive algorithms of the KP must be prepared to operate in the presence of limited and uncertain inputs.
- The KP must reason about information tradeoffs. Sometimes, a global perspective may be critical. Other times, it may be unimportant, or merely somewhat useful. The KP must be prepared to reason about the tradeoffs involved in using data of differing scope. For instance, diagnosing a web server failure may, or more likely, may not require polling for user experience from locations far away. A KP may need to employ introspective meta-reasoning to act most effectively in these circumstances.

Our second objective for the KP system architecture is that to the extent possible it develop, utilize, and reason about information at whatever scope is appropriate for the problem it is addressing.

4.1 Functional and Structural Requirements

The above objectives, together with the core goals of the knowledge plane, lead us to several top-level functional and structural architectural requirements. We discuss four of these below.

4.1.1 Core Foundation

The heart of the knowledge plane is its ability to integrate behavioral models and reasoning processes into a distributed, networked environment. The first component of this ability is support for the creation, storage, propagation and discovery of a variety of information, likely including *observations*, which describe current conditions; *assertions*, which capture high-level goals, intentions and constraints on network operations; and *explanations*, which are

using its locally specified goals and constraints. But when two regions then connect, there may be further constraints that one imposes on the other. So a provider network might say to a subscriber network: NO_MULTICAST. This might cause the subscriber network to change some of its internal organization, disable some end-user applications, and so on.

Support for overlay networks: If the KP has enough information to configure the network itself, that information can also be useful to applications that are configuring themselves. For example, we are increasingly seeing the development of application-specific overlay networks on the Internet. Each overlay network uses edge-based mechanisms to evaluate the performance of different possible paths through the Internet, and seeks to build a set of transport paths that effectively route application packets through what appears to be the part of the Internet best suited to the application's needs. Currently, application networks must probe the Internet, because there is no mechanism for them to learn about the capabilities of the network core. The KP would be in a position to aggregate application- and network-derived knowledge about network performance, offer applications better information about the network than they could learn by probing, and access to control points whose behavior could be modified to help better meet the applications' needs. The KP thus enables per-application control over traffic without the need to build per-application infrastructure throughout the network.

Knowledge-enhanced intrusion detection: There are a number of projects (and a number of products) that perform some sort of analysis to detect network intrusions. In general they look for patterns in data observed somewhere in the network. The current generation of these tools trigger both false positives and false negatives. It has been hypothesized that a next generation of tools for intrusion detection will require that observations from several points in the network will have to be correlated, in order to get a more robust and useful signal. The development of the knowledge plane provides a basis to implement this data gathering and correlation.

4. Knowledge Plane Architecture

Previous sections of this paper have outlined the goals we set for a knowledge plane. In this section, we consider aspects of its system structure. Our discussion is speculative: any successful KP architecture will be shaped by a number of requirements and constraints, not all of which are apparent today. At its highest level the architecture of the KP will be shaped heavily by two broad forces: its distributed, compositional structure, and its multi-scale, potentially global knowledge perspective.

Our ultimate objective is that networked systems should organize themselves, under the constraints and guidance of external inputs, to meet the goals of their stakeholders. Even in the near term, the KP must respect and build on the fact that networks have internal structure and dynamics -- large networks are composed by interconnecting smaller ones, participants come and go, and relationships between the owners, operators and users of different networks may change even when the physical structure does not.

This implies that the knowledge plane serving a network is not a globally engineered entity, but is instead an autonomously created structure that is recursively, dynamically, and continuously composing and decomposing itself from smaller sub-planes. This requirement argues that the KP:

- Is distributed - KP functionality for different regions of the network is physically and logically decentralized.
- Is bottom up - simple entities (e.g. web servers) can compose into larger, more complex entities (e.g. a web farm) as needed, and decompose themselves from the system as appropriate. This is a recursive process, that may proceed on many levels.
- Is constraint driven - the basic principle is that the system can, and may, adopt (or not) any behavior that is not specifically constrained.
- Moves from simple to complex. Speaking generally, the act of composing a set of networks to form a larger one places more requirements or constraints on the behavior of each network. A trivial example would be that a standalone IP network can use a wide range of addresses, but connecting it to a larger network constrains the range of options in this regard.

Our first objective for the KP system architecture is that it support this distributed, compositional perspective, providing the necessary enabling ions and capabilities.

In contrast with the distributed *organization* of the KP, we have argued in previous sections of this paper that KP may often benefit from taking a global *perspective* - integrating observations and conclusions from many points in the network. Key implications of this are that:

- Data and knowledge integration is a central function of the KP. The KP must be able to collect, filter, reduce, and route observations, assertions, and conclusions from different parts of the network to points where they are useful.
- The KP must operate successfully in the presence of imperfect information. Because this global perspective is both physically large and spans multiple administrative entities, the cognitive algorithms of the KP must be prepared to operate in the presence of limited and uncertain inputs.
- The KP must reason about information tradeoffs. Sometimes, a global perspective may be critical. Other times, it may be unimportant, or merely somewhat useful. The KP must be prepared to reason about the tradeoffs involved in using data of differing scope. For instance, diagnosing a web server failure may, or more likely, may not require polling for user experience from locations far away. A KP may need to employ introspective meta-reasoning to act most effectively in these circumstances.

Our second objective for the KP system architecture is that to the extent possible it develop, utilize, and reason about information at whatever scope is appropriate for the problem it is addressing.

4.1 Functional and Structural Requirements

The above objectives, together with the core goals of the knowledge plane, lead us to several top-level functional and structural architectural requirements. We discuss four of these below.

4.1.1 Core Foundation

The heart of the knowledge plane is its ability to integrate behavioral models and reasoning processes into a distributed networked environment. The first component of this ability is support for the creation, storage, propagation and discovery of a variety of information, likely including *observations*, which describe current conditions; *assertions*, which capture high-level goals, intentions and constraints on network operations; and *explanations*, which are

an example of how knowledge itself is embodied—explanations take observations and assertions and map them to conclusions.

To learn about and alter its environment, the knowledge plane must access, and manage, what the cognitive community calls *sensors* and *actuators*. Sensors are entities that produce observations. Actuators are entities that change behavior (e.g., change routing tables or bring links up or down). So, for instance, a knowledge application that sought to operate a network according to certain policies might use sensors to collect observations on the network, use assertions to determine if the network's behavior complies with policy, and, if necessary, use actuators to change the network's behavior.

The most central aspect of the knowledge plane is its support for cognitive computations. This is a challenging problem because the dynamic and distributed KP environment is not well matched to the classical knowledge level algorithms and agent architectures that underpin much of current AI. Most AI algorithms are not designed to work in a highly distributed context, and direct experience in building a large distributed data management system with embedded cognitive abilities is limited.[1] What is needed are robust, tractable and on-line algorithms for environments that are highly dynamic, partially observable, stochastic and error prone. The field of Multi-Agent Systems [22] has had some initial success in solving these problems, although those addressed to date typically lack the dynamicity required for the knowledge plane. Thus refinement of this portion of the knowledge plane architecture, its infrastructure support for a range of appropriate cognitive algorithms, is likely to progress in conjunction with further research in cognitive systems themselves.

4.1.2 Cross-Domain and Multi-Domain Reasoning

Where does the KP "run"? The composed, regional structure of the KP might suggest that a specific server would support the part of the KP that "reasons about" a region, for example an Internet AS. One possibility is that the administrator of the AS would run the KP that oversaw that AS. At a more level, one might state this structuring strategy as "each region is responsible for reasoning about itself."

This is a bad idea, for several reasons. If the AS is down, this could render the relevant KP information unreachable at exactly the wrong time. The administrator of an AS might wish to limit the conclusions that the KP reached about it, perhaps to remove knowledge that seems unflattering, while others may choose to reach those conclusions anyway. These examples show that reasoning about an AS occurs independently of the AS; a fact that should be reflected in the system structure. Different parts of the KP might independently reason about an AS, and compare answers, to detect that part of the KP is corrupted. This shows that there should be no specific physical relationship between a region of the network and the KPs reasoning engines related to that region.

A more radical possibility is that multiple entities compete to provide information about a given AS. Each entity collects its own data and sells its observations. The KP could seek information from whichever entity or entities it believes provides the most accurate and timely (or most cost effective) information. This ``knowledge marketplace" creates a host of architectural challenges, ranging from how to reason about information from multiple providers (even if three different companies tell you the same thing about an AS, it may turn out that they're all reselling data from one Internet weather service: if you really want a second opinion, how do you find the second weather service?) to how to design KP protocols to discourage different knowledge companies from subtly "enhancing" the KP protocols or data in ways that make it harder for users to concurrently use the servers of other knowledge providers?

This discussion demonstrates the potential richness of information flow in the KP. Messages need to flow to more than one location so that redundant reasoning can occur – and how a message flows may depend on who asks it. Different parts of the KP may reach different conclusions, and reconciling these is as important as is dealing with incomplete input data.

4.1.3 Data and Knowledge Routing

We have argued that the KP will benefit from gaining a global perspective on the network it serves. It is useful to consider how this perspective might come about. In a very small network, it might theoretically be possible to collect all relevant information, and flood that information to each node in the network (more precisely, in the distributed KP).

This idea is clearly impractical in larger networks. First, the sheer volume of information is technically daunting, requiring a highly scalable solution. Beyond this, forces such as competition and privacy come into play. In a network of any size, it is necessary to limit and optimize the collection and routing of information. More sophistication is needed.

We suggest that the KP architecture should implement a framework for knowledge management and routing characterized by two attributes. It is knowledge-driven - the routing system itself incorporates information about what knowledge is most useful in different circumstances, and uses scalable distributed techniques to filter observations and "attract" the most relevant observations towards potential customers. It understands tradeoffs - it may incorporate the concept of quality - reasoning about producing better or less good answers with correspondingly more or less effort, time, bandwidth, etc., rather than just producing "an" answer.

4.1.4 Reasoning about Trust and Robustness

The KP's combination of compositional structure and global perspective creates challenges to achieving a robust and trustworthy design. Because a functioning KP is formed at any time from the composition of the participating networks, the architecture must reflect the fact that parts of the KP may be corrupted or broken, that some participants may lie or export deliberately flawed reasoning, and that system actions must be based on inputs that may be partial, outdated or wrong.

This suggests that the KP may need to build, maintain, and reason about trust relationships among its components and participants. Portions of the KP that misbehave may be deemed untrustworthy by other portions, and this information may be propagated among portions that have decided to trust each other. In this way, a web of trust can grow that identifies KP elements that seem to be trustworthy and shuns elements that are not. This introspection would likely require the development of trust models, and the use of

[1] One early and related attempt, the DARPA-sponsored Automated Network Management (ANM) project, sought to build a network-wide MIB collector combined with AI tools [7]. The ANM experience was that collecting data was relatively easy, but getting the data to the right place was hard – it was easy to overwhelm links with management traffic if information was circulated too aggressively.

an example of how knowledge itself is embodied—explanations take observations and assertions and map them to conclusions.

To learn about and alter its environment, the knowledge plane must access, and manage, what the cognitive community calls *sensors* and *actuators*. Sensors are entities that produce observations. Actuators are entities that change behavior (e.g., change routing tables or bring links up or down). So, for instance, a knowledge application that sought to operate a network according to certain policies might use sensors to collect observations on the network, use assertions to determine if the network's behavior complies with policy, and, if necessary, use actuators to change the network's behavior.

The most central aspect of the knowledge plane is its support for cognitive computations. This is a challenging problem because the dynamic and distributed KP environment is not well matched to the classical knowledge level algorithms and agent architectures that underpin much of current AI. Most AI algorithms are not designed to work in a highly distributed context, and direct experience in building a large distributed data management system with embedded cognitive abilities is limited.[1] What is needed are robust, tractable and on-line algorithms for environments that are highly dynamic, partially observable, stochastic and error prone. The field of Multi-Agent Systems [22] has had some initial success in solving these problems, although those addressed to date typically lack the dynamicity required for the knowledge plane. Thus refinement of this portion of the knowledge plane architecture, its infrastructure support for a range of appropriate cognitive algorithms, is likely to progress in conjunction with further research in cognitive systems themselves.

4.1.2 Cross-Domain and Multi-Domain Reasoning

Where does the KP "run"? The composed, regional structure of the KP might suggest that a specific server would support the part of the KP that "reasons about" a region, for example an Internet AS. One possibility is that the administrator of the AS would run the KP that oversaw that AS. At a more level, one might state this structuring strategy as "each region is responsible for reasoning about itself."

This is a bad idea, for several reasons. If the AS is down, this could render the relevant KP information unreachable at exactly the wrong time. The administrator of an AS might wish to limit the conclusions that the KP reached about it, perhaps to remove knowledge that seems unflattering, while others may choose to reach those conclusions anyway. These examples show that reasoning about an AS occurs independently of the AS; a fact that should be reflected in the system structure. Different parts of the KP might independently reason about an AS, and compare answers, to detect that part of the KP is corrupted. This shows that there should be no specific physical relationship between a region of the network and the KPs reasoning engines related to that region.

A more radical possibility is that multiple entities compete to provide information about a given AS. Each entity collects its own data and sells its observations. The KP could seek information from whichever entity or entities it believes provides the most accurate and timely (or most cost effective) information. This ``knowledge marketplace" creates a host of architectural challenges, ranging from how to reason about information from multiple providers (even if three different companies tell you the same thing about an AS, it may turn out that they're all reselling data from one Internet weather service: if you really want a second opinion, how do you find the second weather service?) to how to design KP protocols to discourage different knowledge companies from subtly "enhancing" the KP protocols or data in ways that make it harder for users to concurrently use the servers of other knowledge providers?

This discussion demonstrates the potential richness of information flow in the KP. Messages need to flow to more than one location so that redundant reasoning can occur – and how a message flows may depend on who asks it. Different parts of the KP may reach different conclusions, and reconciling these is as important as is dealing with incomplete input data.

4.1.3 Data and Knowledge Routing

We have argued that the KP will benefit from gaining a global perspective on the network it serves. It is useful to consider how this perspective might come about. In a very small network, it might theoretically be possible to collect all relevant information, and flood that information to each node in the network (more precisely, in the distributed KP).

This idea is clearly impractical in larger networks. First, the sheer volume of information is technically daunting, requiring a highly scalable solution. Beyond this, forces such as competition and privacy come into play. In a network of any size, it is necessary to limit and optimize the collection and routing of information. More sophistication is needed.

We suggest that the KP architecture should implement a framework for knowledge management and routing characterized by two attributes. It is knowledge-driven - the routing system itself incorporates information about what knowledge is most useful in different circumstances, and uses scalable distributed techniques to filter observations and "attract" the most relevant observations towards potential customers. It understands tradeoffs - it may incorporate the concept of quality - reasoning about producing better or less good answers with correspondingly more or less effort, time, bandwidth, etc., rather than just producing "an" answer.

4.1.4 Reasoning about Trust and Robustness

The KP's combination of compositional structure and global perspective creates challenges to achieving a robust and trustworthy design. Because a functioning KP is formed at any time from the composition of the participating networks, the architecture must reflect the fact that parts of the KP may be corrupted or broken, that some participants may lie or export deliberately flawed reasoning, and that system actions must be based on inputs that may be partial, outdated or wrong.

This suggests that the KP may need to build, maintain, and reason about trust relationships among its components and participants. Portions of the KP that misbehave may be deemed untrustworthy by other portions, and this information may be propagated among portions that have decided to trust each other. In this way, a web of trust can grow that identifies KP elements that seem to be trustworthy and shuns elements that are not. This introspection would likely require the development of trust models, and the use of

[1] One early and related attempt, the DARPA-sponsored Automated Network Management (ANM) project, sought to build a network-wide MIB collector combined with AI tools [7]. The ANM experience was that collecting data was relatively easy, but getting the data to the right place was hard – it was easy to overwhelm links with management traffic if information was circulated too aggressively.