

Internet Research Task Force
Internet-Draft
Intended status: Informational
Expires: May 07, 2014

M. Behringer
Cisco
G. Huston
Asia Pacific Network Information Centre
November 03, 2013

A Framework for Defining Network Complexity
draft-irtf-ncrg-complexity-framework-01.txt

Abstract

Complexity is a widely used parameter in network design, yet there is no generally accepted definition of the term. Complexity metrics exist in a wide range of research papers, but most of these address only a particular aspect of a network, for example the complexity of a graph or software. There is a desire to define the complexity of a network as a whole, as deployed today to provide Internet services. This document provides a framework to guide research on the topic of network complexity.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 07, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	3
2. General Considerations	3
2.1. The Behavior of a Complex Network	3
2.2. Robust Yet Fragile	4
2.3. The Complexity Cube	4
2.4. Related Concepts	4
2.5. Technical Debt	5
2.6. Layering considerations	6
3. Tradeoffs	6
4. Structural Complexity	7
5. Components of Complexity	7
5.1. The Physical Network (Hardware)	7
5.2. State in the Network	7
5.3. Churn	8
5.4. Algorithms	8
6. Location of Complexity	8
6.1. Topological Location	8
6.2. Logical Location	8
6.3. Layering Considerations	8
7. Dependencies	8
7.1. Local Dependencies	9
7.2. Network Wide Dependencies	9
7.3. Network External Dependencies	9
8. Management Interactions	9
8.1. Configuration Complexity	9
8.2. Troubleshooting Complexity	9
8.3. Monitoring Complexity	9
8.4. Complexity of System Integration	9
9. External Interactions	10
9.1. User Interactions	10
9.2. Interactions on End Systems	10
9.3. Inter-Network Interactions	10
10. Examples	10
11. Security Considerations	10
12. Acknowledgements	11
13. Informative References	11
Authors' Addresses	12

1. Introduction

During the design phase of a network complexity plays a key role. Network designers generally seek to find the simplest design that fulfils a set of requirements. As no objective definition of network complexity exists, subjective measures are used to come to a conclusion. The resulting diverging views on what constitutes complexity subsequently lead to conflicts in design teams. While most people would agree that complexity is an important factor in network design, today's design decisions are made based on a rough estimation of the network's complexity, rather than a solid understanding.

The goal of this document is to define a framework for network complexity research. This framework describes related research and current understanding of the topic, as well as outlining some ways research could be taken forward. Specifically, contributions are invited in all of the areas mentioned.

Many references to existing research in the area of network complexity are listed on the Network Complexity Wiki [wiki]. This wiki also contains background information on previous meetings on the subject, previous research, etc.

2. General Considerations

2.1. The Behavior of a Complex Network

While there is no generally accepted definition of network complexity, there is some understanding of the behavior of a complex network. It has some or all of the following properties:

- o Self-Organization: A network runs some protocols and processes without external control; for example a routing process, failover mechanisms, etc. The interaction of those mechanisms can lead to a complex behaviour.
- o Un-predictability: In a complex network, the effect of a local change on the behaviour of the global network may be unpredictable.
- o Emergence: A network has an emergent property if a small local change produces a large scale, seemingly unrelated state or result.
- o Non-linearity: An input into the network produces a non-linear result.

- o Fragility: A small local input can break the entire system.

2.2. Robust Yet Fragile

Networks typically follow the "robust yet fragile" paradigm: They are designed to be robust against a set of failures, yet they are very vulnerable to other failures. Doyle [Doyle] explains the concept with an example: The Internet is robust against single component failure, but fragile to targeted attacks. The "robust yet fragile" property also touches on the fact that all network designs are necessarily making trade-offs between different design goals. The simplest one is articulated in "The Twelve Networking Truths" RFC1925 [RFC1925]: "Good, Fast, Cheap: Pick any two (you can't have all three)." In real network design, trade-offs between many aspects have to be made, including, for example, issues of scope, time and cost in the network cycle of planning, design, implementation and management of a network platform. Tradeoff between various parameters are discussed in section 3.

2.3. The Complexity Cube

Complex tasks on a network can be done in different components of the network. For example, routing can be controlled by central algorithms, and the result distributed (e.g., OpenFlow model); the routing algorithm can also run completely distributed (e.g., routing protocols such as OSPF or ISIS), or a human operator could calculate routing tables and statically configure routing. Behringer [Behringer] defines these three axes of complexity as a "complexity cube" with three axes: Network elements, central systems, and human operators. While different functions can be shifted between these axes of the network, the overall complexity may change.

2.4. Related Concepts

When discussing network complexity, a large number of influencing factors have to be taken into account to arrive at a full picture, for example:

- o State in the network: Contains the network elements, such as routers, switches (with their OS, including protocols), lines, central systems, etc. The number and algorithmical complexity of the protocols on network devices for example.
- o Human operators: Complexity manifests itself often by a network that is not completely understood by human operators. Human error is a primary source for catastrophic failures, and therefore must be taken into account.

- o Classes / templates: Rather than counting the number of lines in a configuration, or the number of hardware elements, more important is the number of classes from which those can be derived. In other words, it is probably less complex to have 1000 interfaces which are identically configured than 5 that are completely different configured.
- o Dependencies and interactions: The number of dependencies between elements, as well as the interactions between them has influence on the complexity of the network.
- o TCO (Total cost of ownership): TCO could be a good metric for network complexity, if the TCO calculation takes into account all influencing factors, for example training time for staff to be able to maintain a network.
- o Benchmark Unit Cost is a related metric that indicates the cost of operating a certain component. If calculated well, it reflects at least parts of the complexity of this component. Therefore, the way TCO or BUC are calculated can help to derive a complexity metric.
- o Churn / rate of change: The change rate in a network itself can contribute to complexity, especially if a number of components of the overall network interact.

Networks differ in terms of their intended purpose (such as is found in differences between enterprise and public carriage network platforms, and in their intended role (such as is found in the differences between so-called "access" networks and "core" transit networks). The differences in terms of role and purpose can often lead to differences in the tolerance for, and even the metrics of, complexity within such different network scenarios. This is not necessarily a space where a single methodology for measuring complexity, and defining a single threshold value of acceptability of complexity, is appropriate.

2.5. Technical Debt

Many changes in a network are made with a dependency on the existing network. Often, a suboptimal decision is made because the optimal decision is hard or impossible to realise at the time. Over time, the number of suboptimal changes in themselves cause significant complexity, which would not have been there had the optimal solution been implemented.

The term "technical debt" refers to the accumulated complexity of sub-optimal changes over time. As with financial debt, the idea is

that also technical debt must be repaid one day by cleaning up the network or software.

2.6. Layering considerations

In considering the larger space of applications, transport services, network services and media services, it is feasible to engineer responses for certain types of desired applications responses in many different ways, and involving different layers of the so-called network protocol stack. For example, quality of Service could be engineered at any of these layers, or even in a number of combinations of different layers.

Considerations of complexity arise when mutually incompatible measures are used in combination (such as error detection and retransmission at the media layer in conjunction with the use TCP transport protocol), or when assumptions used in one layer are violated by another layer. This results in surprising outcomes that may result in complex interactions. This has lead to the perspective that increased layering frequently increases complexity [RFC3439].

While this research work is focussed network complexity, the interactions of the network with the end-to-end transport protocols, application layer protocols and media properties are relevant considerations here.

3. Tradeoffs

>[I-D.irtf-ncrg-network-design-complexity] describes a set of trade-offs in network design to illustrate the practical choices network operators have to make. The amount of parameters to consider in such tradeoff scenarios is very large, thus that a complete listing may not be possible. Also the dependencies between the various metrics itself is very complex and requires further study. This document attempts to define a methodology and an overall high level structure.

To analyse tradeoffs it is necessary to formalise them. The list of parameters for such tradeoffs is long, and the parameters can be complex in themselves. For example, "cost" can be a simple unidimensional metric, but "extensibility" or "optimal forwarding state" are harder to define in detail.

A list of parameters to trade off contains metrics such as:

- o Cost: How much does the network cost to build (capex) and run (opex)

- o Bandwidth / delay / jitter: Traffic characteristics between two points (average, max, ...)
- o Configuration complexity: How hard to configure and maintain the configuration
- o Susceptibility to Denial-of-Service: How easy is it to attack the service
- o Security (confidentiality / integrity): How easy is it to sniff / modify / insert the data flow
- o Scalability: To what size can I grow the network / service
- o Extensibility: Can I use the network for other services in the future?
- o Ease of troubleshooting: How hard is it to find and correct problems?
- o Predictability: If I change a parameter, what will happen?
- o Clean failure: When a problem arises, does the root cause lead to deterministic failure

The list of the above criteria can be seen as forming an n-dimensional design space, where each network is represented in one intersection of all parameters.

4. Structural Complexity

tbc

5. Components of Complexity

Complexity can be found in various components of a networked system. For example, the configuration of a network element reflects some of the complexity contained in this system. Or an algorithm used by a protocol may be more or less complex. When classifying complexity the first question to ask is "WHAT is complex?". This section offers a method to answer this question.

5.1. The Physical Network (Hardware)

tbc

5.2. State in the Network

tbc

5.3. Churn

The frequency of change in a network intuitively contributes to its complexity: A network which is not subjected to change tends to be more stable [need ref here]. While there is permanently a certain base complexity in the network, this complexity is "under control" and does not lead to negative side effects.

[I-D.sircar-complexity-entropy] describes how entropy metrics can be used to describe changing complexity in a network. The fundamental thesis is that change itself constitutes complexity. When a network undergoes change, the network entropy and the complexity increases. This is also true when the change has simplification as a goal. The entropy increases during change, and decreases in periods of stability. It can therefore be used to measure the impact of change on complexity.

5.4. Algorithms

tbc

6. Location of Complexity

The previous section discussed in which form complexity may be perceived. This section focuses on where this complexity is located in a network. For example, an algorithm can run centrally, distributed, or even in the head of a network administrator. In classifying the complexity of a network, the location of a component may have an impact on overall complexity. This section offers a methodology to the question "WHERE is the complex component?"

6.1. Topological Location

tbc

6.2. Logical Location

tbc

6.3. Layering Considerations

tbc

7. Dependencies

Dependencies are generally regarded as related to overall complexity. A system with less dependencies is generally considered less complex. This section proposes a way to analyse dependencies in a network.

For example, [Chun] states: "We conjecture that the complexity particular to networked systems arises from the need to ensure state is kept in sync with its distributed dependencies."

In this document we distinguish three types of dependencies: Local dependencies, network wide dependencies, and network external dependencies.

7.1. Local Dependencies

tbc

7.2. Network Wide Dependencies

tbc

7.3. Network External Dependencies

tbc

8. Management Interactions

A static network generally is relatively stable; conversely, changes introduce a degree of uncertainty and therefore need to be examined in detail. Also, the trouble shooting of a network exposes intuitively the complexity of the network. This section proposes a methodology to classify management interactions with regard to their relationship to network complexity.

8.1. Configuration Complexity

tbc

8.2. Troubleshooting Complexity

tbc

8.3. Monitoring Complexity

tbc

8.4. Complexity of System Integration

tbc

9. External Interactions

The user experience of a network also illustrates a form of complexity. A network can expose certain tasks to the user, or deal with them internally, hidden to the user. This section describes how user interactions can be analysed to expose complexity.

9.1. User Interactions

tbc

9.2. Interactions on End Systems

tbc

9.3. Inter-Network Interactions

tbc

10. Examples

In the foreseeable future it is unlikely to define a single, objective metric that includes all the relevant aspects of complexity. In the absence of such a global metric, a comparative approach could be easier.

For example, it is possible to compare the complexity of a centralised systems where algorithms run centrally, and the results are distributed to the network nodes with a distributed algorithm. The type of algorithm may be similar, but the location is different, and a different dependency graph would result. The supporting hardware may be the same, thus could be ignored for this exercise. Also layering is likely to be the same. The management interactions though would significantly differ in both cases.

The classification in this document also makes it easier to survey existing research with regards to which area of complexity is covered. This could help in identifying open areas for research.

11. Security Considerations

This document does not discuss any specific security considerations.

12. Acknowledgements

The motivations and framework of this overview of studies into network complexity is the result of many meetings and discussions, with too many people to provide a full list here. However, key contributions have been made by: John Doyle, Jon Crowcroft, Mark Handley, Fred Baker, Paul Vixie, Lars Eggert, Bob Briscoe, Keith Jones, Bruno Klauser, Steve Youell, Joel Obstfeld.

The authors would like to acknowledge the contributions of Rana Sircar, Ken Carlberg and Luca Caviglione in the preparation of this Research Group document.

13. Informative References

[Behringer]

Behringer, M., "Classifying Network Complexity",
Proceedings of the ACM Re-Arch'09, December 2009.

[Chun]

Chun, B-G., Ratnasamy, S., and E. Eddie, "NetComplex: A Complexity Metric for Networked System Design", 5th Usenix Symposium on Networked Systems Design and Implementation NSDI 2008, April 2008,
<<http://berkeley.intel-research.net/sylvia/netcomp.pdf>>.

[Doyle]

Doyle, J., "The 'robust yet fragile' nature of the Internet", PNAS vol. 102 no. 41 14497-14502, October 2005.

[I-D.irtf-ncrg-network-design-complexity]

Retana, A. and R. White, "Network Design Complexity Measurement and Tradeoffs", draft-irtf-ncrg-network-design-complexity-00 (work in progress), August 2013.

[I-D.sircar-complexity-entropy]

Sircar, R. and M. Behringer, "Using Entropy as a Measure for Changes in Network Complexity", draft-sircar-complexity-entropy-00 (work in progress), October 2013.

[RFC1925]

Callon, R., "The Twelve Networking Truths", RFC 1925, April 1996.

[RFC3439]

Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, December 2002.

[wiki]

, "Network Complexity Wiki", ,
<<http://networkcomplexity.org/>>.

Authors' Addresses

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Geoff Huston
Asia Pacific Network Information Centre

Email: gih@apnic.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2013

A. Retana
Cisco Systems
R. White
Verisign
March 14, 2013

A Framework for Measuring Network Complexity
draft-retana-network-complexity-framework-00.txt

Abstract

Network architecture revolves around the concept of fitting the design of a network to its purpose; of asking the question, "what network will best fit these needs?" A part of fitting network design to requirements is the problem of complexity, an idea often measured by "seat of pants" methods. When would adding a particular protocol, policy, or configuration be "too complex?" This document suggests a series of continuums along which network complexity might be measured. No suggestions are made in measuring complexity for each of these continuums are provided; this is left for future documents.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 15, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements notation	3
3. Control Plane State verses Optimal Forwarding Paths (Stretch)	3
4. Configuration State verses Failure Domain Separation	4
5. Policy Centralization verses Optimal Policy Application	6
6. Configuration State verses Per Hop Forwarding Optimization	7
7. Reactivity verses Stability	7
8. Conclusion	9
9. Security Considerations	9
10. Normative References	9
Authors' Addresses	10

1. Introduction

Network complexity is a systemic, rather than component level, problem; complexity must be measured in terms of the multiple moving parts of a system, and complexity may be more than the complexity of the individual pieces, examined individually, might suggest. There are two basic ways in which systemic level problems might be addressed: interfaces and continuums. In addressing a systemic problem through interfaces, we seek to treat each piece of the system as a "black box," and develop a complete understanding of the interfaces between these black boxes. In address a systemic problem as a continuum, we seek to understand the impact of a single change or element to the entire system as a set of tradeoffs. While network complexity can profitably be approached from either of these perspectives, the authors of this document have chosen to approach the systemic impacts of network complexity from the perspective of continuums of tradeoffs. In theory, modifying the network to resolve one particular problem (or class of problems) will add complexity which results in the increased liklihood (or appearance) of another class of problems. Discovering these continuums of tradeoffs, and then determining how to measure each one, become the key steps in understanding and measuring systemic complexity in this view.

This document proposes five such continuums; more may be possible. Others may be added into this document in future revisions, or documented in other drafts, as circumstances dictate.

- o Control Plane State verses Optimal Forwarding Paths (or it's opposite measure, stretch)
- o Configuration State verses Failure Domain Separation
- o Policy Centralization verses Optimal Policy Application
- o Configuration State verses Per Hop Forwarding Optimization
- o Reactivity verses Stability

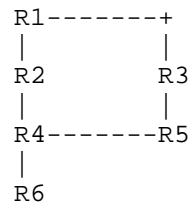
Each of these continuums is described in a separate section of this draft.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", "OPTIONAL" in this document are to be interpreted as in [RFC2119].

3. Control Plane State verses Optimal Forwarding Paths (Stretch)

Control plane state is the aggregate amount of information carried by the control plane through the network in order to produce the forwarding table at each device. Each additional piece of information added to the control plane --such as more specific reachability information, policy information, additional control planes for virtualization and tunneling, or more precise topology information-- adds to the complexity of the control plane. This added complexity, in turn, adds to the burden of monitoring, understanding, troubleshooting, and managing the network. Removing control plane state, however, is not always a net positive gain for the network as a system; removing control plane state almost always results in decreased optimality in the forwarding and handing of packets travelling through the network. This decreased optimality can be termed stretch, which is defined as the difference between the absolute shortest (or best) path traffic could take through the network and the path the traffic actually takes through the network. Stretch is expressed as the difference between the optimal and actual path. The figure below provides an example of this tradeoff.



Assume each link is of equal cost in this figure, and:

- o R4 is advertising 192.0.2.1/32 as a reachable destination not shown on the diagram
- o R5 is advertising 192.0.2.2/32 as a reachable destination not shown on the diagram
- o R6 is advertising 192.0.2.3/32 as a reachable destination not shown on the diagram

For R1, the shortest path to 192.0.2.3/32, advertised by R6, is along the path [R1,R2,R4,R6]. Assume, however, the network administrator decides to aggregate reachability information at R2 and R3, advertising 192.0.2.0/24 towards R1 from both of these points. This reduces the overall complexity of the control plane by reducing the amount of information carried past these two routers (at R1 only in this case). Aggregating reachability information at R2 and R3, however, has the impact of making both routes towards 192.168.2.3/32 appear as equal cost paths to R1; there is no particular reason R1 should choose the shortest path through R2 over the longer path through R3. This, in effect, increases the stretch of the network. The shortest path from R1 to R6 is 3 hops, a path that will always be chosen before aggregation is configured. Assuming half of the traffic will be forwarded along the path through R2 (3 hops), and half through R3 (4 hops), the network is stretched by $((3+4)/2) - 3$, or .5, a "half a hop."

Traffic engineering through various tunneling mechanisms is, at a broad level, adding control plane state to provide more optimal forwarding (or network utilization). Optimizing network utilization may require detuning stretch (intentionally increasing stretch) to increase overall network utilization and efficiency; this is simply an alternate instance of control plane state (and hence complexity) weighed against optimal forwarding through the network.

4. Configuration State verses Failure Domain Separation

A failure domain, within the context of a network control plane, can be defined as the set of devices impacted by a change in the network topology or configuration. A network with larger failure domains is more prone to cascading failures, so smaller failure domains are normally preferred over larger ones. The primary means used to limit the size of a failure domain within a network's control plane is information hiding; the two primary types of information hidden in a network control plane are reachability information and topology information. An example of aggregating reachability information is summarizing the routes 192.0.2.1/32, 192.0.2.2/32, and 192.0.2.3/32 into the single route 192.0.2.0/24, along with the aggregation of the metric information associated with each of the component routes. Note that aggregation is a "natural" part of IP networks, starting with the aggregation of individual hosts into a subnet at the network edge. An example of topology aggregation is the summarization of routes at a link state flooding domain boundary, or the complete failure to advertise topology information in a distance-vector protocol.

While limiting the size of failure domains appears to be an absolute good in terms of network complexity, there is a definite tradeoff in configuration complexity. The more failure domain edges created in a network, the more complex configuration will become. This is particularly true is redistribution of routing information between multiple control plane processes is used to create failure domain boundaries; moving between different types of control planes causes a loss of the consistent metrics most control planes rely on to build loop free paths. Redistribution, in particular, opens the door to very destructive positive feedback loops within the control plane. Examples of control plane complexity caused by the creation of failure domain boundaries include route filters, routing aggregation configuration, and metric modifications to engineer traffic across failure domain boundaries.

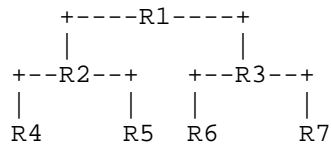
Returning to the network described in the previous section, aggregating routing information at R2 and R3 will divide the network into two failure domains: (R1,R2,R3), and (R2,R3,R4,R5). A failure at R5 should have no impact on the forwarding information at R1. A false failure domain separation occurs, however, when the metric of the aggregate route advertised by R2 and R3 is dependent on one of the routes within the aggregate. For instance, if the metric of the 192.0.2.0/24 aggregate is taken from the metric of the component 192.0.2.1/32, then a failure of this one component will cause changes in the forwarding table at R1 --in this case, the control plane has not truly been separated into two distinct failure domains. The added complexity in the illustration network would be the management of the configuration required to aggregate the control plane information, and the management of the metrics to ensure the control plane is truly separated into two distinct failure domains.

Replacing aggregation with redistribution adds the complexity of managing the feedback of routing information redistributed between the failure domains. For instance, if R1, R2, and R3 were configured to run one routing protocol, while R2, R3, R4, R5, and R6 were configured to run another protocol, R2 and R3 could be configured to redistribute reachability information between these two control planes. This can split the control plane into multiple failure domains (depending on how, specifically, redistribution is configured), but at the cost of creating and managing the redistribution configuration. Further, R3 must be configured to block routing information redistributed at R2 towards R1 from being redistributed (again) towards R4 and R5.

5. Policy Centralization verses Optimal Policy Application

Another broad area where control plane complexity interacts with optimal network utilization is Quality of Service (QoS). Two specific actions are required to optimize the flow of traffic through a network: marking and Per Hop Behaviors (PHBs). Rather than examining each packet at each forwarding device in a network, packets are often marked, or classified, in some way (typically through Type of Service bits) so they can be handled consistently at all forwarding devices. Packet marking policies must be configured on specific forwarding devices throughout the network. Distributing marking closer to the edge of the network necessarily means configuring and managing more devices, but produces optimal forwarding at a larger number of network devices. Moving marking towards the network core means packets are marked for proper handling across a smaller number of devices. In the same way, each device through which a packet passes with the correct PHBs configured represents an increase in the consistency in packet handling through the network as well as an increase in the number of devices which

must be configured and managed for the correct PHBs. The network below is used for an illustration of this concept.



In this network, marking and PHB configuration may be configured on any device, R1 through R7. Assume marking is configured at the network edge; in this case, four devices, (R4,R5,R6,R7), must be configured, including ongoing configuration management, to mark packets. Moving packet marking to R2 and R3 will halve the number of devices on which packet marking configuration must be managed, but at the cost of consistent packet handling at the inbound interfaces of R2 and R3 themselves. Thus reducing the number of devices which must have managed configurations for packet marking will reduce optimal packet flow through the network. Assuming packet marking is actually configured along the edge of this network, configuring PHBs on different devices has this same tradeoff of managed configuration verses optimal traffic flow. If the correct PHBs are configured on R1, R2, and R3, then packets passing through the network will be handled correctly at each hop. The cost involved will be the management of PHB configuration on three devices. Configuring a single device for the correct PHBs (R1, for instance), will decrease the amount of configuration management required, at the cost of less than optimal packet handling along the entire path.

6. Configuration State verses Per Hop Forwarding Optimization

The number of PHBs configured along a forwarding path exhibits the same complexity verses optimality tradeoff described in the section above. The more types of service (or queues) traffic is divided into, the more optimally traffic will be managed as it passes through the network. At the same time, each class of service must be managed, both in terms of configuration and in its interaction with other classes of service configured in the network.

7. Reactivity verses Stability

The speed at which the network's control plane can react to a change in configuration or topology is an area of widespread study. Control plane convergence can be broken down into four essential parts:

- o Detecting the change

- o Propagating information about the change
- o Determining the best path(s) through the network after the change
- o Changing the forwarding path at each network element along the modified paths

Each of these areas can be addressed in an effort to improve network convergence speeds; some of these improvements come at the cost of increased complexity.

Changes in network topology can be detected much more quickly through faster echo (or hello) mechanisms, lower layer physical detection, and other methods. Each of these mechanisms, however, can only be used at the cost of evaluating and managing false positives and high rates of topology change. If the state of a link change can be detected in 10ms, for instance, the link could theoretically change state 50 times in a second --it would be impossible to tune a network control plane to react to topology changes at this rate. Injecting topology change information into the control plane at this rate can destabilize the control plane, and hence the network itself. To counter this, most fast down detection techniques include some form of dampening mechanism; configuring and managing these dampening mechanisms represents an added complexity that must be configured and managed.

Changes in network topology must also be propagated throughout the network, so each device along the path can compute new forwarding tables. In high speed network environments, propagation of routing information changes can take place in tens of milliseconds, opening the possibility of multiple changes being propagated per second. Injecting information at this rate into the control plane creates the risk of overloading the processes and devices participating in the control plane, as well as creating destructive positive feedback loops in the network. To avoid these consequences, most control plane protocols regulate the speed at which information about network changes can be transmitted by any individual device. A recent innovation in this area is using exponential backoff techniques to manage the rate at which information is injected into the control plane; the first change is transmitted quickly, while subsequent changes are transmitted more slowly. These techniques all control the destabilizing effects of rapid information flows through the control plane through the added complexity of configuring and managing the rate at which the control plane can propagate information about network changes.

All control planes require some form of algorithmic calculation to find the best path through the network to any given destination.

These algorithms are often lightweight, but they still require some amount of memory and computational power to execute. Rapid changes in the network can overwhelm the devices on which these algorithms run, particularly if changes are presented more quickly than the algorithm can run. Once the devices running these algorithms become processor or memory bound, it could experience a computational failure altogether, causing a more general network outage. To prevent computational overloading, control plane protocols are designed with timers limiting how often they can compute the best path through a network; often these timers are exponential in nature, allowing the first computation to run quickly, while delaying subsequent computations. Configuring and managing these timers is another source of complexity within the network.

Another option to improve the speed at which the control plane reacts to changes in the network is to precompute alternate paths at each device, and possibly preinstall forwarding information into local forwarding tables. Additional state is often needed to precompute alternate paths, and additional algorithms and techniques are often configured and deployed. This additional state, and these additional algorithms, add some amount of complexity to the configuration and management of the network. In some situations (for some topologies), a tunnel is required to pass traffic around a network failure or topology change. These tunnels, while not manually configured, represent additional complexity at the forwarding and control planes.

8. Conclusion

This document describes various areas of network and design where complexity is traded off against some optimization in the operation of the network. This is (by it's nature) not an exhaustive list, but it can serve to guide the measurement of network complexity and the search for other areas where these tradeoffs exist.

9. Security Considerations

None.

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Alvaro Retana
Cisco Systems
2610 Wycliff Road
Raleigh, NC 27607
USA

Email: aretana@cisco.com

Russ White
Verisign
12061 Bluemont Way
Reston, VA 20190
USA

Email: russw@riw.us