Macro Trends, Complexity, and Software Defined Networking An Architectural View

David Meyer

dmm@1-4-5.net

Abstract

A series of powerful *Macro Trends* are reshaping our ideas about how we architect, design and engineer network systems. This has lead network researchers, designers and operators to propose novel new architectures, including Software Defined Networking, to address the consequences of these trends. This paper explores these trends and associated architectures and applies ideas from diverse fields such as biological systems theory and quantitative risk management (as well as traditional network theory) with an eye towards gaining a deeper understanding of the implications for network operators. One perhaps surprising consequence is that rather than creating enhanced stability, those network systems that reside "above" the narrow waist of the Internet protocol stack are the source of increasing volatility and uncertainty. We argue that this volatility and uncertainty is a direct consequence of the increasing role of software in the design, deployment, and operation of network systems.

1 Introduction

A series of powerful *Macro Trends* are reshaping our ideas about how we architect, design and engineer network systems. This has lead network researchers, designers and operators to propose novel new architectures such as Software Defined Networking (SDN) to address the consequences of these trends. This paper explores these trends and associated architectures and applies ideas from diverse fields such as biological systems theory and quantitative risk management (as well as traditional network theory) with an eye towards gaining a deeper understanding of the implications for network operators. One perhaps surprising consequence is that rather than creating enhanced stability, those network systems that reside "above" the narrow waist of the Internet protocol stack are the source of increasing volatility and uncertainty. We argue that this volatility and uncertainty is a direct consequence of the increasing role of software in the design, deployment, and operation of network systems.

The remainder of this paper is organized as follows Section 2 describes the SDN problem space and hypothesis. . Section 4 discusses several of the macro trends and architectural features that are shaping current thinking. Section 5 describes the history of SDN technology and Section 6 describes the current state of SDN technology and thinking. Finally, Section 7 concludes with a discussion of where SDN and its associated technologies are headed.

2 The SDN Problem Space and Hypothesis

Network architects, engineers and operators are being presented with the following challenge:

Provide state of the art network infrastructure and services while minimizing TCO

The *SDN Hypothesis*, then, is that it is the lack of ability to innovate in the underlying network coupled with the lack of proper network abstractions that is the root cause of the inability to keep pace with user requirements and to keep TCO under control [9]. A key observation was that network requirements were in most cases stated informally in an out of band manner and were static in nature suggested that these tasks are better done by machine (i.e., programmatically). In particular, programatic automation of configuration, management, and monitoring, as well as robust feedback channels to the network are required for scale and resilience.[1] In addition, networks were seen to suffer from the following three problems [15]:

- No common data plane abstraction; every device was different
- Few network-wide abstractions; the network provided an myopic,box centric view
- Configuration and policy were deeply intertwined

The lack of data plane programmability and network-wide abstractions resulted in, according to the SDN hypothesis, the design and implementation of networks that are complex, fragile, and close to impossible to evolve[2]. As we will see in Section 5, a new network architecture, OpenFlow/SDN (OF/SDN) [9] was proposed to address these shortcomings. The rest of this paper analyzes the OF/SDN architecture in light of an emerging set of macro trends and new insight into complexity and architecture gleaned from the biological system theory and quantatative risk management fields.

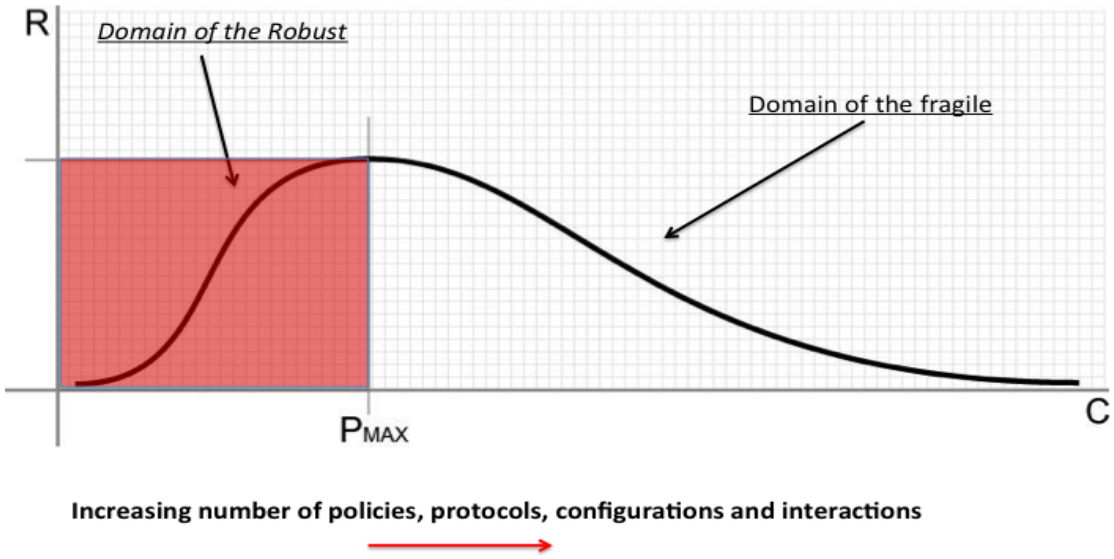

Figure 1: Robustness vs. Complexity – Systems View

2.1 Robustness and Fragility

Figure 1 depicts a typical "Complexity-Robustness" curve for a class of familiar systems which includes the Internet and many biological systems, among others [2]. As we can see such systems need complexity to acquire robustness. However, beyond a certain point (P_{max}) additional complexity causes the fragility and brittleness, two of the properties OF/SDN is designed to mitigate. While precise detailed discussions of robustness, fragility, and complexity are beyond the scope of this paper, the next sections provide an overview of these topics.

2.2 Robustness

Definition: A *[property]* of a *[system]* is robust if it is *[invariant]* with respect to a *[set of perturbations]*, up to some limit [1].

[1] In fact, as many as 80% of unscheduled outages are caused by people and process errors [11].
[2] As we shall see, this paper presents a somewhat different view.

Interestingly, a system can have the property that it is robust to one set of perturbations and yet fragile for a different property and/or set of perturbations. Such a system is called *Robust Yet Fragile* (or RYF-complex). For example, a possible *RYF tradeoff* is that a system with high efficiency (i.e., using minimal system resources) might be unreliable (i.e., fragile to component failure) or hard to evolve. Examples of systems exhibiting RYF-complex behavior include many biological signaling paths and the Internet [1].

Note that we can characterize many familiar system properties as forms of robustness. For example:

- **Reliability** is robustness to component failures
- **Efficiency** is robustness to resource scarcity
- **Scalability** is robustness to changes to the size/complexity of the system as a whole
- **Modularity** is robustness to structure component rearrangements
- **Evolvability** is robustness of lineages to changes on long time scales

2.3 Fragility

Fragility can be most easily seen in an example. Consider a coffee cup on a sitting on a table. The cup is *fragile* because suffers non-linearly more from large deviations than from the cumulative effect of smaller events. More concretely, the cup is dropped on average 1 cm every time it is set down. Picking it up and setting it down say, 300 times, has no effect on the cup. However, dropping the cup from 3 meters destroys it (i.e., the damage grows in a non-linear fashion). Another example: consider the difference in damage to a human who jumps off something that is, 0.3 meters high 30 times versus jumping off something that 9 meters high one time; in the first case, no real damage; in the second case the person is dead.

In the network world, we see several examples of fragility, including

- ARP storms [13]
- Micro-loops in the Internet routing system [12]
- TCP congestion collapse [5]
- The AS 7007 route leakage incident [17]

2.3.1 Formal Definition

Let z be some stress level, p some property, and let *H(p,z)* be the (negative valued) harm function Then for the fragile the following must hold

$$H(p, nz) < nH(p, z) \text{ for } 0 < nz < K \tag{1}$$

where K is the level at which the system collapses. We say such a system is *K-fragile*. The intuition here is that the cumulative damage to a fragile system from a series of small shocks is non-linearly less that the damage from one large shock [14].

Note that Equation 1 is a variation on Jensens Inequality [3] and importantly is not mean-preserving for convex H. In particular

$$H(p, (z1 + z2)/2) \neq (H(p, z1) + H(p, z2))/2 \tag{2}$$

That is, *the convex function of the mean is not equal to the mean of convex functions.* Assuming that a distribution's mean is preserved by averaging is a common technique when the underlying distribution is unknown (perhaps the most common case). As shown here however, such averaging leads to model error and hence additional uncertainty.

Finally, there is a close relationship between fragility as defined here and what we typically think of as scaling. For example, when we say something scales like $O(n^2)$ what we mean is the damage to the network has constant acceleration (2), and as such is of concern as n gets large.

3 Defining Complexity

Alderson and Doyle [1] provide the following insightful description of complexity:

> *In our view, however, complexity is most succinctly discussed in terms of functionality and its robustness. Specifically, we argue that complexity in highly organized systems arises primarily from design strategies intended to create robustness to uncertainty in their environments and component parts.*

That is, complexity is really about *structure* that is "designed" to create robustness to uncertainty. We will return to this theme later in this paper.

4 Macro Trends

In this section we describe six *macro trends* that are shaping the SDN landscape.

4.1 The Rise of Software and Associated Intelligence

In many ways the rise of Software and its associated growth of intelligence closely mimics what we find in the biological world. For example, the evolution of the Precambrian (reptilian) brain to mammalian neocortex is very much akin to the shift from hardware-centric to

software-centric infrastructures [10]. Not surprisingly, there are many architectural themes which are shared by large dynamic systems which are both scalable and evolvable. These include thin-waist architectures (see Section 4.1.1), massively distributed control planes, highly layered architectures and a high degree of component reuse.

4.1.1 Bowties and Thin Waist Architectures

[4] [7]

The view of architecture as constraints that deconstrain (17, 18) originated in biology, but it is consistent with engineering (16) and illustrated by clothing. A robust architecture is constrained by protocols, but the resulting plug and play modularity that these shared constraints enable deconstrain (i.e., make flexible) systems designed using this architecture. Constraints give a convenient starting language to formalize and quantify architecture and ultimately, a mathematical foundation (19). Concretely, consider a given wardrobe that is a collection of garments and the problem of assembling an outfit that provides suitable robustness to the wearer's environment. Three distinct but interrelated types of constraints are universal in clothing as in all architecture (16): (i) component (garment) constraints, (ii) system (outfit) constraints, and (iii) protocol constraints. Therefore, in combination, diverse, heterogeneous components (garments) that are constrained by materials and construction combine synergistically (through protocols) to yield outfits that satisfy system constraints not directly provided by any single component. We will use outfit to describe a functional, robust set of garments, and heap (Craver uses aggregates) (47) to describe a random collection not required to have any other system features.

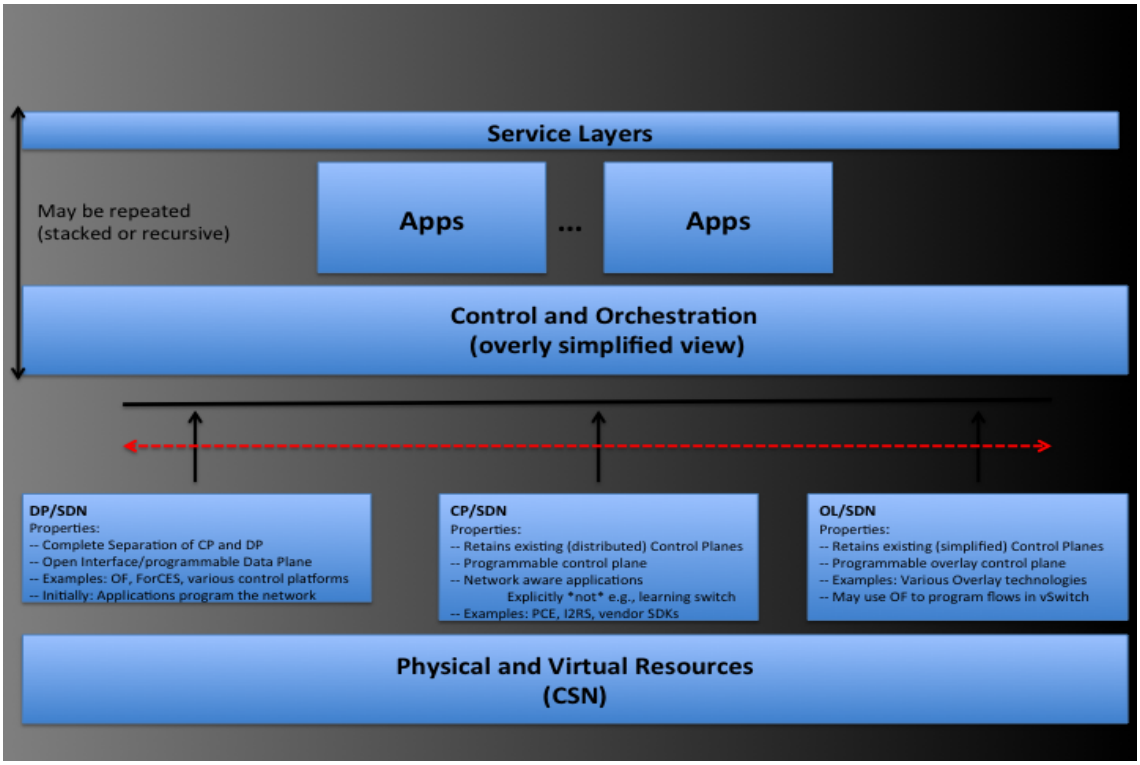

Figure 2: The SDN Continuum

While *Software Defined Networking (SDN)* is a concept that has escaped precise definition. However, initial definitions we're comprised of three basic architectural components[16]:

- An open and standardized interface to the switching data plane
- Separation of control and data planes
- Centralized control

5 How Did We Get Here?

[8] [6]

6 What is the Current State of Affairs

7 A View to the Future

References

[1] D. Alderson and J. Doyle. Contrasting views of complexity and their implications for network-centric infrastructures. In *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Human*, volume 40 Issue 4, pages 839– 852, 2012.

[2] J. Carlson". Highly optimized tolerance:a mechanism for power laws in designed systems. *Physical Review E*, 60(2):1412–1427, 1999.

[3] I Csiszar. A note on jensen's inequality. *Studia Sci. Math. Hungar*, 1, 1966.

[4] John C. Doyle and Marie Csete. Architecture, constraints, and behavior. In Donald W. Pfaff, editor, *Proceedings of the National Academy of Sciences of the United States*, volume 108, pages 15624–15630, Septemeber 2011.

[5] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. Netw.*, 7(4):458–472, August 1999.

[6] Infocomm. *Software-Defined Networking*, April 2009.

[7] Marc Kirschner and John Gerhart. Evolvablity. In *Proceedings of the National Academy of Sciences of the United States*, volume 95, pages 8420–8427, July 1998.

[8] Teemu Koponen, Scott Shenker, Hari Balakrishnan, Nick Feamster, Igor Ganichev, Ali Ghodsi, P. Brighten Godfrey, Nick McKeown, Guru Parulkar, Barath Raghavan, Jennifer Rexford, Somaya Arianfar, and Dmitriy Kuptsov. Architecting for innovation. *SIGCOMM Comput. Commun. Rev.*, 41(3):24–36, July 2011.

[9] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[10] R. Glenn Northcutta and Jon H. Kaas. The emergence and evolution of mammalian neocortex the emergence and evolution of mammalian neocortex the emergence and evolution of mammalian neocortex the emergence and evolution of mammalian neocortex the emergence and evolution of mammalian neocortex. *Trends in Neurosciences*, 18(9):373–379, Septemeber 1995.

[11] D. Scott. Making smart investments to reduce unplanned downtime. Technical Report TG-07-4033, March 1999.

[12] M. Shand and S. Bryant. IP Fast Reroute Framework. RFC 5714 (Informational), January 2010.

[13] S.Vidya and R.Bhaskaran. Arp storm detection and prevention measures. In *IJCSI International Journal of Computer Science Issues*, volume 8, March 2011.

[14] Nassim Nicholas Taleb. Black swans and the domains of statistics, 2007.

[15] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: a language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 43–48, New York, NY, USA, 2012. ACM.

[16] Kok-Kiong Yap, Te-Yuan Huang, Ben Dodson, Monica S. Lam, and Nick McKeown. Towards software-friendly networks. In *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, APSys '10, pages 49–54, New York, NY, USA, 2010. ACM.

[17] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. An analysis of bgp multiple origin as (moas) conflicts. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 31–35, New York, NY, USA, 2001. ACM.