Software Defined "X"

Sukanta Ganguly, *Research Engineer, San Jose, California 95120, USA, Email: sukanta@locomobi.com*

doi: 10.22545/2014/00051

The power of computing has been realized in two main directions, 1) The growth of hardware devices and systems 2) The enhancements in software technologies. The growth of hardware has branched out into two areas, specialized hardware for vertical focus and generic hardware with applicability in multiple areas. While the growth in the software arena also started in the same path the morphs that software technologies have taken are much faster and much more wider in scope. Hence we do see a faster growing software economy. The Software Defined "X" trend that we have seen has been evolving in the last decade started getting recognition with the Software Defined Radios (SDRs). SDRs, which offered capabilities to make the hardware radio frequency layer or layer 1 in the OSI stack controllable via software and offering the same hardware to latch onto multiple radio frequency channels led the way to more software defined technologies to be researched and invented. In this paper we would study the Software Defined Networking effort that has caused a major disruption in the networking world and how it addressed a big issue that has been lying dormant in this industry for more than two decades. The technological innovations applied to the Software Defined Networking segment helps build-out a wider industry application base, hence creating a level playing field for a larger yet targeted software application base helping the consumers get better choices to decide on their consumption needs.

Keywords*:* Software, networking platform, network policies, networking systems.

1 Introduction

Software Defined "X" is a means for disruptive forces to push innovation in areas that were rigid and structured via hardware as the fundamental basis for development. We have seen a tremendous growth in software and hardware in the last decade. The growth has been backed by the insatiable demand for computing in not only the normally existing areas where computing usage has been a key driver but also areas where computing could be used for various reasons ranging for feasibility, form-factor, practical closures of the applications, connectivity, lack of reasonable interfaces, etc. The growth also has seen multiplicative effect with the revolution of hardware and software innovations. We will focus in the Software Defined "X" with some details covering the Software Defined Networking area, but we will also expand into other areas of Software Defined technology innovations.

Networking itself is a wide area to cover and we will talk about the Wide Area Networking, which has seen a lot of research and applications already actively deployed. We will address aspects of Security, Wireless Networking and Storage as well. The basic definition of software driving functionalities within hardware on which is operates is not a new area. We find that the last decade or so we have seen more software penetration of software in controlling hardware. Hardware has always existed as a purpose built entity, architected, designed and developed for a specific job. Due to the way hardware gets designed and developed it is difficult to bring in changes into the functionality of hardware easily

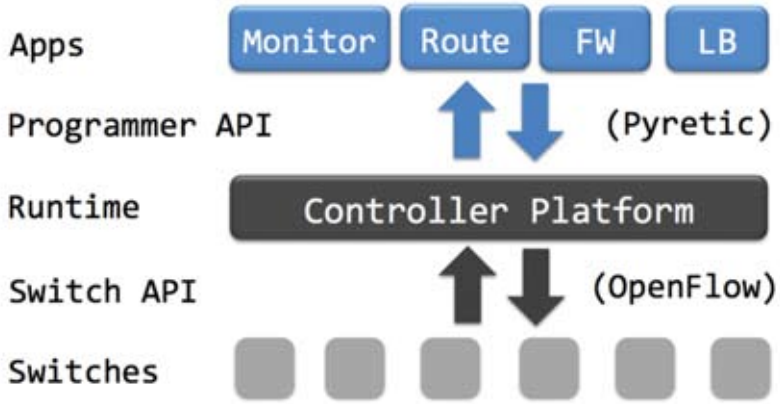

Figure 1: Software defined networks.

and in less time. The arduous process involved in designing a good hardware system makes it difficult to make changes on the fly. Software, on the other hand, has different characteristics. Software reuse is a very important characteristic of a good software design. Given that modularity and reuse are important items that software industry as followed making changes and getting newer and different implementations to be added are expected modes of software driven operations.

2 System Elements

Software defining [3, 5] functionalities have been experimented in many areas. It starts with things like languages [3] and frameworks [7] that are used to develop these software systems. Languages used to build new software defined modules are domain centric hence of the language constructs [3, 8] will be more application centric. Domain-aware programming languages were developed to make it easy to program applications with constructs that are directly applicable to the applications with the specific domain. In [3] the authors present a domain specific language used to develop software defined networking applications. Software Defined Networking platform is being designed to house new applications along with the existing ones with certain structural programming model changes that will help evolve and expose things within networking framework which were considered hard-wired. Although the characteristics of the networking framework don't change the ways the functionality is exposed and managed change significantly.

Pyretics [16] policy language has a number of features that are designed to make it easy to construct and combine policies in a modular way. Most networks perform multiple tasks, such as routing, monitoring, and access control. Ideally, programmers would be able to implement these tasks independently, using separate modules. But the programming interfaces available today make this difficult, since packet-handling rules installed by one module often interfere with overlapping rules installed by another module. It allows the networking switch to do the following; 'query the current network state', 'express new network policies' such that the networking data plane reacts according to the fed in policies and 'reconfigure the network' such that different behavior can be implemented. All of these can be done on an ad-hoc basis and fed into the switches in real time. There is no downtime expressed in reprogramming the network. With the help of the language the authors have created a new environment for the applications to be created by many programmers who are not original owners of the network switch. The abstractions offered helps reduce the barrier to creativity.

In Figure 1 above, using Pyretics applications like Monitor (monitors traffic flowing through the network switch), Route (feeds in specialized routes that are manually injected for traffic to follow instead of routes being discovered from within the network), etc. An application developer can easily use the Pyretics language to develop simple applications that can be developed externally and injected into a switch that is complaint with the framework. A two-tiered software framework is shown with the

Table 1: Sample programming.

Syntax	**Summary**
identity	returns original packet
drop	return empty set
match ($f = v$)	identity if field matches v, drop otherwise
modify ($f = v$)	returns packet with field f set to v
fwd(a)	modify (port = a)
flood ()	returns one packet for each local port on the network spanning tree

architecture.

A domain specific syntax supported by the Pyretic language is shown Table 1. A subset of syntax is listed on the left hand side of the diagram. As the syntax suggest they are very specific operations that can be closely tied to the network switch domain. For e.g. the syntax '**flood**' is generally used to broadcast the same packet across all the networking ports within the switch. The syntax '**drop**' is used to drop a packet once it is received on a port. There could be qualifiers that can be passed to make sure that the action of dropping a packet is performed after certain filters are matched. The syntax '**modify**' would allow for a modification of packet content by replacing one or more field values with another. The syntax '**fwd**' would instruct the system to forward a specific type of a packet to be forwarded to a specific port. The forwarding of the packet to a port can be time driven or with certain matching fields values.

Languages play a strong part in building software-defining actions within the network segment and several others segments has had languages with syntax that are applicable for their domains. Such domain specific languages do form framework and/or platform via which such value-added applications can be developed. The frameworks generally come with the language syntaxes and interpreter or compiler arranged to create intermediate binary.

Software defined systems required modes in which new software base logic can be injected within an existing, live environment. This fundamental need exist no matter which domain it has been applied to. The model of building software needs modularity in such environment where software components are decomposed into smaller modules. These smaller modules can be replaced; enhancement or new control flows can be inserted or added into the system. Those derivations help upgrade, make it more relevant per target applications, or morph the device for business relevant needs.

3 Application Domains

Various application domains have already seen early stage impacts of software defined networking architectures being implemented. Some of the areas where Software defined solutions are being actively researched and implemented and discussed in this paper are Wide Area Networking[9, 11] (WAN), Optical Networking Transport [5], Soft-Radio implementation[1], Orthogonal Frequency based wireless encoding [2, 15], Storage Networking [12, 14].

Wide Area Networking systems are a critical piece of the entire Internet connectivity to be maintained. A good operating WAN needs to have a uniform utilization across all the links in a WAN. High utilization requires frequent updates to the networks data plane. The data plane is the path where data traffic flows in real time. The data plane is established during a flow when the path is discovered via the route discovery process. Changes in traffic demand based on flow needs as well as changes in the network topology based on route outages or reroutes based on policy changes are common in WAN. A key challenge is to implement these updates without causing transient congestion that can hurt latency-sensitive traffic. The underlying problem is that the updates are not atomic, as they require changes to multiple switches in the network. Even if the before and after states are not congested, congestion can occur during updates if traffic that a link is supposed to carry after the update arrives before the traffic that is supposed to leave has left. The extent and duration of such congestion is worse when the network is busier and has larger RTTs (which lead to greater temporal disparity in the application of updates). Both these conditions hold for our setting, and we find that uncoordinated updates lead to severe con-

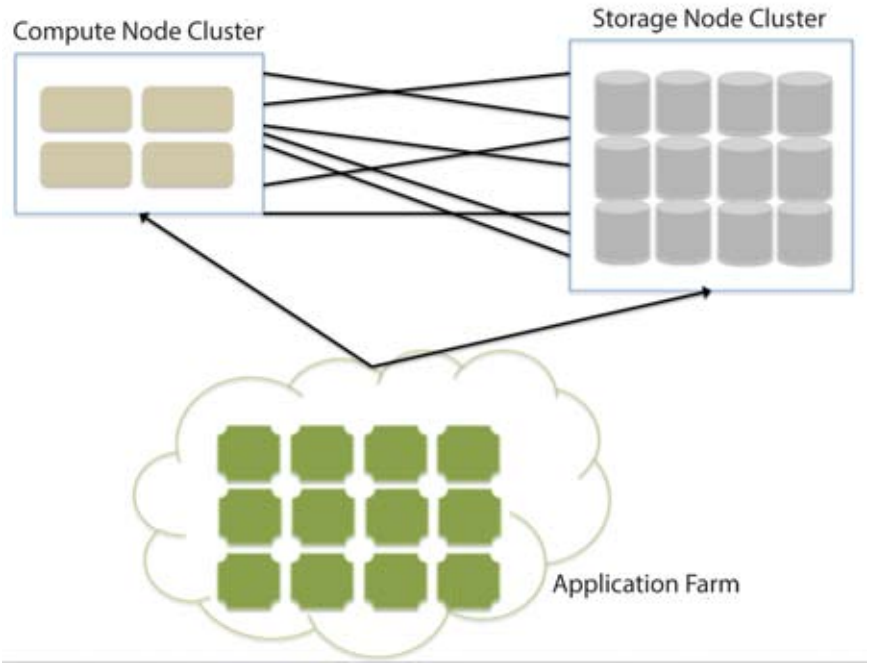

Figure 2: Software networking in virtual machine clusters.

gestion and heavy packet loss. Some of the issues that need to dealt with are fully using network capacity requires many forwarding rules at switches, to exploit many alternative paths through the network, but commodity switches support a limited number of forwarding rules.

Analysis of a production inter-data center WAN shows that the number of rules required to fully use its capacity exceeds the limits of even next generation SDN switches. We address this challenge by dynamically changing, based on traffic demand, the set of paths available in the network. On the same WAN, our technique can fully use network capacity with an order of magnitude fewer rules. Due to the distributed approach of rules designed it is possible to have them farmed properly to the right links. The distribution of the rules relevant for specific segment can be fed into the system. WAN optimization is quite a challenging problem as the state of a link within a WAN segment changes constantly. Due to the change it is hard to predict what type of rules will have what impact for any route flows. Any applied rule will have to deal with re-prioritization of traffic flow of some type(s) and rerouting the flows can create turbulence within the entire WAN.

Storage Networking and IO's within a storage area network has some very interesting applications that software defined networking can bring to the table. Storage area networks deal with IO traffic from the application to the storage systems. These storage entities are network attached at various levels within the network stack. Each class of storage device has its own performance criteria called ratings. The ratings are utilized in architecting a storage networking sub-system. Software defined systems within a storage networking area helps in building frameworks which can be used in developing dynamic data and IO flows. Virtual Machines are a big factor in data centers and enterprise networks. Storage systems are decoupled from compute systems and in the world of virtualizations compute nodes, IO nodes and application processing can all be virtualized. This virtualization provides some very exciting ways for applications to be consuming storage and networking resources. Figure 2, shows an example of application farm or cluster virtualized from the compute nodes that they will be using as well as the IO nodes that will be used during the functioning of the application. As is shown in the figure, the application(s) are not tied to one specific compute entity or a one specific storage block. Due to abstractions being introduced within the layers it is possible to decouple them and perform dynamic mappings based on the needs of the applications and the time the application is running. A policy engine is needed to facilitate policies that can be injected into the

system. Some policy implementation is dynamic and some are static. Static policy engine need the system to be bootstrapped with the policies already fed into it. The ones that support dynamic injection of policies add more dynamism within the entire storage-networking framework.

Implementations of policies act as key driver into the software defined networking systems. Although, policy based networking existed for a much longer time than Software Defined Networking, what allows the policy engine implementation to be adding more value is for the system behavior change that can be implemented with and SDN is far more granular and more impactful for applications to leverage the environment. The customizability from the run-time point of view is far more superior with SDNs. Application level queues as well as system level queues are introduces at the end-points where policies are implemented. While network devices have implemented queue management algorithms based on network traffic, i.e. packet header inspection, for storage networking these are not that easily implementable per existing architectures. Some of the limitations may be due to changes in packet envelope data identifiers, which impact the forwarding on a link-by-link basis. Hence uniform policies can be devised and pushed into the storage networks. One of the key expectations of the queues would be to make sure rate limiting behavior is expected on the type of the queue. Rate limiting is hard in traditional storage networks due to the unpredictable relationship between processing of the request and the rate at which IO is consumed. Factors that effect these could be data locality, ratings of the actual physical block on which data resides, the type of virtualization layer that will be encountered within the IO stack, etc. Given all these, various types of controls need to be added at various IO processing layers as well as at different systems, some closer to the hardware and some much higher and distributed within the entire system. The distributed nature also adds additional network traffic for proper compliance of the system. With SDN mode of development the modularity and the abstractions are built per the specific infrastructure and as the storage network topology changes, it can be reconfigured and reprogrammed with extreme ease.

In Figure 2 we see that the applications use compute resources as deemed required from the compute pool and the storage pool provides its applicability per the need from the applications. The virtualization layer that exists between these layers is completely transparent to the application. As software developers design their application there is cognizance of realistic limits of resources availability but no practical programming models offer ways to limit the implementation based on the restrictions made available. Furthermore, the restrictions change depending on the state of the environment. Virtualization leads to multi-use and multiplexing of resources and hence workload prediction becomes really hard. Multi-tenancy within these environments makes it even harder to manage resources and user experiences. The application server instance becomes the front-end for the abstracted storage volumes and the compute processing nodes. Storage is usually virtualized i.e. a VMs are often unaware of the details of the interconnect fabric and the storage configuration. Virtual machines are presented with virtual hard disks that will simplify large files on the storage servers. These types of storage virtualization will reduce the complexity of management like volume and block migration including the entire virtual machine migration.

Wireless networking [2, 15] has seen some research and investigations of software-defined environments to figure out the benefits for this domain. Wireless networking, in this discussion applies to all the ranges of RF protocols in the wireless domain that is in business play, like 2.4 GHz, 5.0 GHz. As other frequency bands get into the consumption range similar work may be applicable to them, albeit there may be some new protocol related optimizations that may be special to the new frequency bands that are difficult to generalize.

Figure 3(a) shows an SDN controller for a wireless controller. In this example we show a controller software stack with some of the major components and their layering. For this discussion we will ignore the physical layer and the media access control layer as well as their software/firmware modules including the encoding/decoding methods. The SDN controller is the main software block, which coordinates with the end-points in its range. The controller has a important task of coordinating with the end-points, reading their current status, making sure any network policies are passed on to the end-points and they are implementable. Like an air-traffic controller it handles reroutes, end-to-end discoveries and destination knowledge.

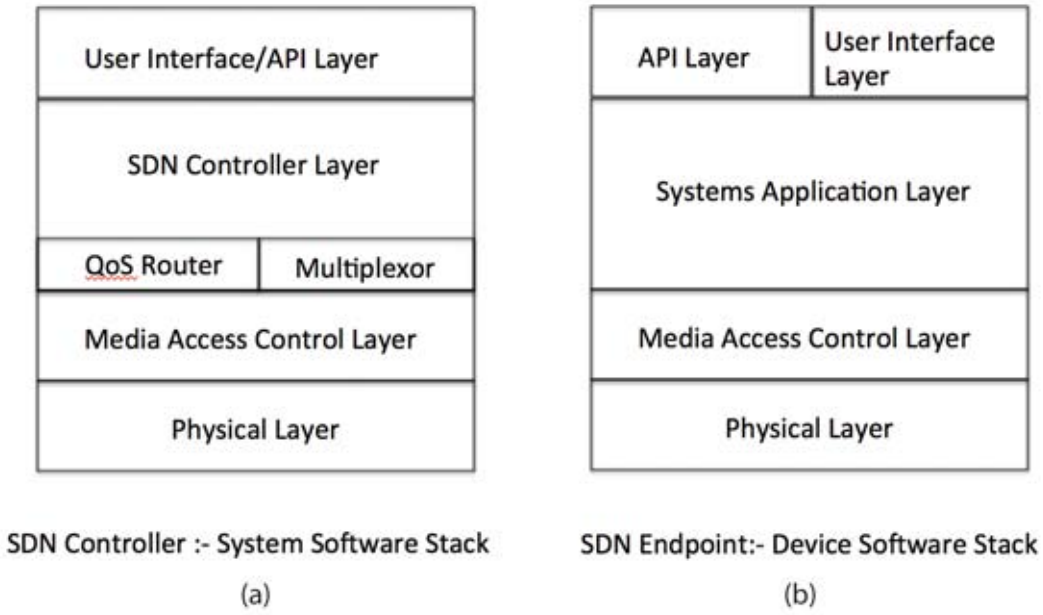

Figure 3: (a) SDN controller for a wireless networking environment, (b) SDN endpoint of a wireless networking environment.

Figure 3(b) shows the SND endpoint devices software stack. As in the case of Figure 3(a), this is also one specific design listing the major software modules or components. There can be many more low-level details that may or may not be listed here and other designs may have emphasis which is either not listed here or not relevant for our discussion.

The system applications layer is the main target module that houses the details of the endpoint system logic. One of the key functionality the endpoint performs is packet forwarding. The process of forwarding includes making changes in the meta-data of the packet. Most of the endpoints do not function as a bridge and hence no bridging logic is added into it. The controller plays and important to make a decision as to which endpoint should get the packet.

The wireless networking infrastructure when deployed has the goals of making sure packets over the RF channels moves from the sender to the network backhaul or to the same RF channel to reach the destination. While doing this, it also has to make sure that the local information from the endpoints are collected and forwarded to the central controller, which can either use the data or forward it to the aggregated uplinks. This is also called topology discovery process. In the wireless networking world where the endpoints are not tethered to specific locations or switches we do expect them to move and the moves can be as complex as going from one controller administrative zone to another one or zone hopping. The process of communication between the endpoint and a controller requires each endpoint to know which exactly is its primary controller. Since the endpoints typically work with one and only one controller the primary controller is the only controller. Beacon packets are generated by the endpoints that are broadcasted. The beacon packets have values like the receive signal strength indicator, battery power, etc. Through the beacon packet broadcast all the other nodes deduce the hops to reach each other. The endpoints store the information about its neighbors in a local table that it shares with the controller from time to time. The central controller deduces the network topology at any point in time with the help of the information it collected form the local tables received from the endpoints.

With a programmable interface that allows systems software to be embedded within the network controllers, which in turn feeds them into the endpoints we can form a real-time and dynamically reacting wireless environment. The behavior shown via these systems can change its functionalities and the total system behavior at any point in time, which is not possible in the current systems. New ranges of protocol and implementation design are being

researched as well as accepted within the wireless networking domain to create reactive systems with low latency impersonations being injected into the environment.

4 Technology Impact

The SDN framework, which is appearing in terms of software system that is domain independent as a base yet has many different vertical segment impacts are creeping in from research world into the consumer/business world. It has started making waves in environments where things were expected to be rigid and fixed in nature. It has brought into light the gains that can be shown and the richness that can be housed within systems that are not point and packaged products anymore but play-beds for real-time changes based on environment where it is exercised. These environments are rich and real. They are live and yet changeable. The amount of research in technology in various areas of computing has given rise to tremendous amount of cross-domain innovation.

Businesses deal with the innovation in terms of new products that are created and evolved into the market place. Research leads the way and business follows one research stabilizes and has end systems that can be build and deployed in the real world.

5 Conclusion

Software-Defined Network and Software defined frameworks to be more generic lead to things that can be created from within the existing systems. SDN does not proclaim change in the functionality. It only lays new ideas and new ways for make systems more agile and reactive to the environments that need them. These changes are valuable in end up creating new market segments. It allows existing markets to grow and help us provide better modes of learning new things and implementing new things can result in better and smarter utilization. The non-existence of these modes of operations was due to the fact that the level of maturity in protocols, hardware architecture, communicating interfaces were not taken into account at atomic levels. The transformations of the same components were due to the fact that the system was decomposable and the decompositions helped it to evolve into actionable elements that can be externally interfaced.

SDNs have already seen quite a bit of growth over the last few years. With the maturity in some areas and with the understanding of SDN in multiple domains the rate at it could grow is far more effective. It has demonstrated some of its values in domains that were well mature already. It has opened doors for innovation and refueled the market with new ideas that has started the process of rejuvenation of the technology. We see this effecting academia as well as the industry.

References

[1] Shriram K. V., Sivaraman R., Alex, Z. C., 2012. Software defined radio implementation (with simulation and implementation). International Journal of Computer Applications (IJCA) ISSN: (0975 - 8887), Vol. 4, No. 8, Aug 2010.

[2] Yun, M., Yuxin, B., 2010. Rapid prototyping of a SDR based reconfigurable MIMO-OFDM testbed. SDR'10 Technical Conference, Washington, DC.

[3] Foster, N., Freedman,J. M., Guha,A., Harrison, R., Praveen K. N., Monsanto, C., Reich, J., Reitblatt, M., Rexford,J., Schlesinger, C., Story, A., Walker,D., 2013. Languages for software-defined networks. IEEE Communications.

[4] Kreutz, D. Fernando, M. V. Ramos, Verissimo,P., 2013. Towards secure and dependable software-defined networks. HotSDN' 13, ACM 978-1-2178-5/13/08.

[5] Gringeri, S., Bitar, N., Xia, T. J., 2013. Extending software defined network principles to include optical Transport. IEEE Communication Magazine, March.

[6] Ma, N. Lu, Z., Pang, Z., Zheng, L., 2010. System-level exploration of mesh-based NoC architectures for multimedia applications. IEEE Internation SOC Conference, pp. 99-104.

[7] McGeer, R., 2012. A safe, efficient upate protocol for open-flow networks. HotSDN' 12, ACM 978-1-4503-1477-0/12/08, Helsinki, Finland.

[8] Kazemian, P., Varghese, G., McKeown, N., 2012. Header space analysis: static checking for networks. Usenix Conference on Network Design and Implementation.

[9] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Holzle, U., Stuart, S., Vahdat, A., 2013. B4: Experience with a globally-deployed software defined WAN. SIGCOMM' 13, Hong Kong, China, ACM 978-1-4503-2056-6/13/08.

[10] Gude, N., Koponen, T., Pettit, J., Pfafe, B., Casado, M., Mckeown, N., Shenker, S., 2008. NOX: towards an operating system for networks. SIGCOMM' CCR, July.

[11] Hong, C., Kandula, S., Mahajan, R., Zhang,M., Gill, V., Nanduri, M., Wattenhofer, R., 2013. Achieving high utilization with software-driven WAN. SIGCOMM' 13, Hong Kong, China, ACM 978-1-4503-2056-6/13/08.

[12] Thereska, E., Ballani, H., O'Shea, G., Karagiannis, T., Rowstron, A., Talpay, T., Black, R., Zhu, T., 2013. IOFlow: A software-defined storage architecture. SOSP' 13, Farmington, PA, USA, ACM 978-1-4503-2088-8/13/11.

[13] Ballani, H., Costa,P., Karagiannis, T., Rowstron, A., 2011. Towards predictable datacenter networks. ACM SIGCOMM', Toronto, Ontario, Canada, pp. 242-253.

[14] Volk, T., Frey, J., 2014. Obstacles and priorities on the jurney to the software-defined data center. An Enterprise Management Associates Research Report January.

[15] Costanzo, S., Galluccio, L., Morabito, G., Palazzo, S., 2012. Software defined wireless networks: unbridling SDNs. IEEE DOI 10.1109/EWSDN.2012.12.

[16] Reich, J., Monsanto, C., Foster, N., Rexford, J., Walker, D., 2013. Modular SDN programming with pyretic. Login Magazine, 38(5), pp. 128-134.

About the Author

Dr. Sukanta Ganguly is a technocrat and has been an entrepreneur forming new businesses. He as taken ideas from concept to a product with go-to-market business planning and revenue modeling for several businesses. He has a Doctorate in Engineering from Texas Tech University in Transdisciplinary area with concentration in the Semantic searching, Data-mining with Context-aware information retrieval. He has an MBA in Finance and a Masters in Computer Science. He has published three books and many academic papers and has been granted several patents.

Dr. Ganguly has 20 years of industry experience. He focuses in the areas of Network & Data Security, Data/Information mining, multi-media protocols and model driven applications.

Copyright © 2014 by the author. This is an open access article distributed under the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.