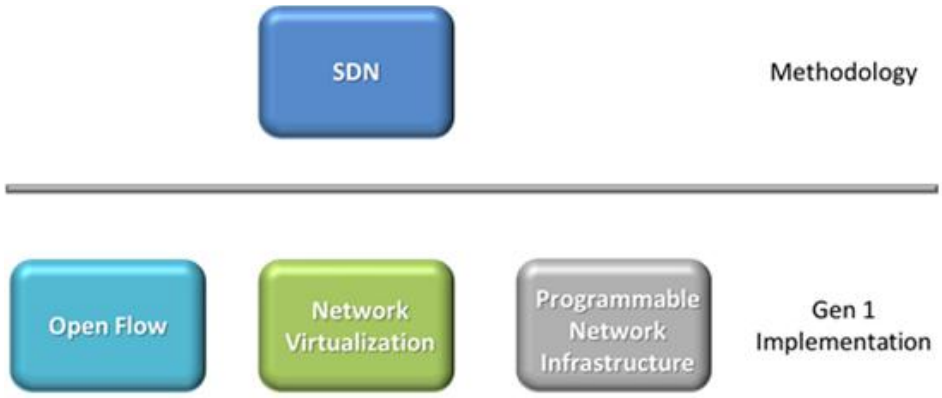
TRUE SOFTWARE DEFINED NETWORKING (SDN)

The world is, and has been, buzzing about software defined networking. It's going to revolutionize the entire industry, commoditize hardware, and disrupt all the major players. It's going to do all that… some day. To date it hasn't done much but be a great conversation, and more importantly identify the need for change in networking.

In its first generation SDN is a lot of sizzle with no flash. The IT world is trying to truly define it, much like we were with 'Cloud' years ago. What's beginning to emerge is that SDN is more of a methodology then an implementation, and like cloud there are several implementations: OpenFlow, Network Virtualization and Programmable Network Infrastructure.



OpenFlow

Open Flow focuses on a separation of control plane and data plane. This provides a centralized method to route traffic based on a 5-tuple match of packet header information. One area OpenFlow falls short is in its dependence on the independent advancement of the protocol itself and the hardware support below. Hardware in the world of switching and routing is Application

Specific Integrated Circuits (ASIC) based, and those ASICs typically take three years to refresh. This means that the OpenFlow protocol itself must advance, and then once stabilized silicon vendors can begin building new ASICs to be available three years later.

Network Virtualization

Network virtualization is a faithful reproduction of networking functionality into the hypervisor. This method is intended to provide advanced automation and speed application deployment. The problem here arises in the new tools required to manage and monitor the network, the additional management layer, and the replication of the same underlying complexity.

Programmable Network Infrastructure

Programmable network infrastructure takes the configuration of devices from human to machine CLI/GUI interfaces to APIs and programming agents. This allows for faster, more powerful and less error prone device configuration from automation, orchestration and cloud operating system tools. These advance the configuration of multiple disparate systems but are still designed based on network operating system constructs intended for human use, and the same underlying network complexities such as artificial ties between addressing and policy.

All of these generation 1 SDN solutions simply move the management of the underlying complexity around. They are software designed to operate in the same model, trying to configure existing hardware. They're simply adding another protocol, or protocols, to the pile of existing complexity.

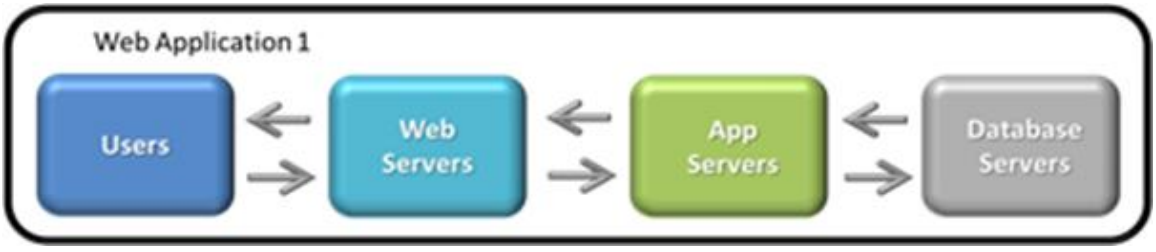
Truly software defined networks

To truly define the network via software you have to look at the entire solution, not just a single piece. Simply adding a software or hardware layer doesn't fix the problem, you must look at them in tandem starting with the requirements for today's networks: automation, application agility, visibility (virtual/physical) security, scale and L4-7 services (virtual/physical.)

If you start with those requirements and think in terms of a blank slate you now have the ability to build things correctly for today and tomorrow's applications while ensuring backwards compatibility. The place to start is in the software itself, or the logical model. Begin with questions:

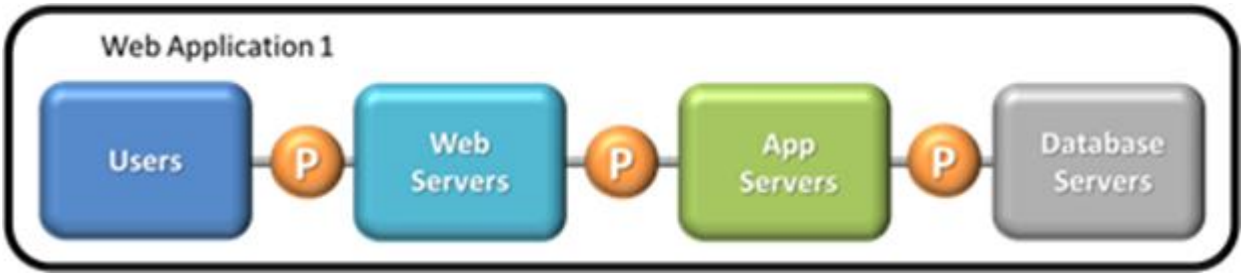
1. What's the purpose of the network?
2. What's most relevant to the business?
3. What dictates the requirements?

The answer to all three is the application, so that's the natural starting point. Next you ask who owns, deploys and handles day two operations for an application? The answer is the development team. So you start with a view of applications in a format they would understand.



That format is simple provider/consumer relationships between tiers or components of an application. Each tier may provide and consume services from the next to create the application which is a group of tiers or components, not a single physical server or VM.

You take that idea a step further and understand that the provider/consumer relationships are truly just policy. Policy can describe many things, but here it would be focused on permit/deny, redirect, SLAs, QoS, logging and L4-7 service chaining for security and user experience.



Now you've designed a policy model that focuses on the application connectivity and any requirements for those connections, including L4-7 services. With this concept you can instantiate that policy in a reusable format so that policy definition can be repeated for like connections, such as users connecting to a web tier. Additionally the application connectivity definition as a whole could be instantiated as a template or profile for reuse.

You've now defined a logical model, based on policy, for how applications should be deployed. With this model in place you can work your way down. Next you'll need network equipment that can support your new model. Before thinking about the hardware, remember there is an operating system (OS) that will have to interface with your policy model.

Traditional network operating systems are not designed for this type of object oriented policy model. Even highly programmable or Linux based operating systems have not been designed for object programmability that would fully support this model. You'll need an OS that's capable of representing tiers or components of an application as objects, with configurable attributes. Additionally it must be bale to represent physical resources like ports as objects abstracted from the applications that will run on them. An OS that can be provisioned in terms of policy

constructs rather than configuration lines such as switch ports, QoS and ACLs. You'll need to rewrite the OS.

As you're writing your OS you'll need to rethink the switching and routing hardware that will deliver all of those packets and frames. Of course you'll need: density, bandwidth, low-latency, etc. More importantly you'll need hardware that can define, interpret and enforce policy based on your new logical model. You'll need to build hardware tailored to the way you define applications, connectivity and policy. Hardware that can enforce policy based on logical groupings free of VLAN and subnet based policy instantiation.

If you build these out together, starting with the logical model then defining the OS and hardware to support it, you'll have built a solution that surpasses the software shims of generation 1 SDN. You'll have built a solution that focuses on removing the complexity first, then automating, then applying rapid deployment through tools usable by development and operations, better yet DevOps.

If you do that you'll have truly defined networking based on software. You'll have defined it from the top all the way down to the ASICs. If you do all that and get it right, you'll have built Cisco's Application Centric Infrastructure (ACI.)

Source: http://www.definethecloud.net/true-software-defined-networking-sdn/