Affinity Networking in an SDN World
Introduction
Software-defined networking (SDN) is all the rage these days. Over the past several months nearly every established networking vendor and a number of start-ups have announced SDN strategies and product plans.
Most SDN discussions revolve around either OpenFlow or Virtual Network Overlays (VNOs). While architecture is an important detail to consider for an SDN, it is also important to remember the reasons for pursuing these new networks in the first place. Perhaps one of the most important of these reasons is to provide networks that can be explicitly driven based on a set of user needs, or simply stated, defined by the user intent.
Plexxi, a new network vendor, has introduced Affinity Networking into the SDN dialog. This paper reviews the Affinity Networking concept and explains how this type of SDN relates to both OpenFlow and VNOs in providing intent-based networks.

SDN: A Steering Wheel for the Network
Legacy networks leverage a set of distributed protocols that allow system components to communicate state and reachability information consistently. Historically, this approach has allowed networks to scale well (at least to the point we’re at today) and enabled many disparate systems to work together seamlessly. The obvious drawback of the distributed system approach is that the entire system behavior is based on how these protocols operate; a characteristic that is fixed when the protocol is written, leaving network operators and ultimately users with very few mechanisms to influence system behavior at run-time.

Rather than relying on autonomous distributed routing and switching protocols, SDNs provide a set of centralized, common controls that devices, end-users, and external applications can use to drive network behavior. This can be thought of as a steering wheel for the network. While debate continues on a common definition of SDN, a fundamentally agreed functional objective is to enable this intent-based networking – the ability to establish deliberate network capabilities to support an expressly desired outcome.
Unfortunately most SDN discussions quickly dissolve into architectural and implementation dissertations about decoupling forwarding and control plane functionality and providing centralized programming interfaces. By going directly to architectural issues, we’ve put the cart before the horse, and lost sight of the higher-level functional goals. Consider the two most commonly discussed SDN approaches: OpenFlow-based SDNs and Virtual Network Overlays.

OpenFlow
OpenFlow is gathering industry attention as a potential standards-based protocol for constructing SDNs. Simply put, OpenFlow is a flow-forwarding API between a controller and a set of forwarding devices. It provides a set of controls that can be used to direct how flows are managed across the various network devices (switches and routers) that are responsible for forwarding traffic.
By itself a flow forwarding API does not provide a steering wheel for the network. This would be akin to steering a car by manually adjusting each wheel. While manipulating flow entries can certainly provide networks with more cost efficient ways of handling some specifically arduous tasks (e.g. network tapping, redirecting for security processing, etc) controlling network-wide behavior on a flow-by-flow basis could be very difficult.

The delineation of control and data plane is an important step in the right direction, but building networks at scale that provide these direct levers to control the network requires whole new software and algorithms that effectively leverage the new architecture underneath. Simply defining a common flow forwarding API does not achieve that in isolation.

Virtual Network Overlays
Virtual Network Overlays (VNOs) allow physical infrastructure to be partitioned into logically isolated networks that can be independently programmed and managed, providing multi-tenancy and greater flexibility for cloud and hosted services. Because VNOs employ software-based forwarding entities (virtual switches) to construct a logical network on top of a physical network they are sometimes considered a form of software-defined networking.
At first glance the VNO approach may appear ideal for providing an inherently steerable network. A particular workload can be placed onto a virtual network tailored to meet its specific needs. But by definition, a VNO exists as a software-based entity. The intent that is captured in the VNO still needs to be communicated to the physical network that is carrying the network traffic for the logical network behavior to be manifest in the real world of traffic forwarding.

For example, a user may wish to leverage different VNOs to enable multiple classes of service for different types of workloads (e.g. development or test workloads versus production workloads). It might be fairly simple to construct different classes of VNOs, but as the application traffic is carried across the physical network
there may not be a way to express that intent to the physical network, in which case the intent is lost.

**Affinity Networking**

Affinity Networking is an SDN approach developed by Plexxi specifically with the intent to give the users of the network (users in this context could mean network operators, applications, or potentially even end users) the levers to manage its behavior directly. Affinity Networks leverage architectural principles from OpenFlow-based networks by implementing a decoupled control plane and data plane. While the specific OpenFlow flow forwarding APIs are not currently used in the current Plexxi implementation, they could easily be leveraged once the standard is sufficiently mature and capable. The more important point is that an Affinity Network leverages this decoupled architecture to build networks that have explicit controls for users to directly affect network behavior, thus satisfying a key driver for SDN.

In a Plexxi network, user intent is captured by defining “affinities.” Affinities describe the relationships and characteristics of the data center resources (physical or virtual compute, storage and network resources) required to execute a given application workload. In Affinity Networking, users (e.g. network operator, external application, developer) control the behavior of the network directly by defining these affinity relationships and attributing specific policies to those relationships.

For example, a set of VMs may be grouped into the tiers that define a multi-tier web application. The user that is constructing or deploying that web application may intend the network to deliver low latency capabilities between the web tier and the database tier. This specific intent can be described very easily by identifying the devices that comprise each tier and defining an affinity policy for latency. In traditional networking, this intent even if captured cannot be expressed to the network because there is no easy way to change the behavior of distributed protocols. SDNs should offer the ability to express this intent, but this goes beyond providing a simple set of flow forwarding rules using OpenFlow.

For Virtual Network Overlays, Affinity Networking provides a substrate that can easily understand the intent of specific virtual networks and ensure that intent is persistent through the end-to-end network. For example, if a virtual network is defined to correlate to a specific tenant in a multi-tenant environment, there may be specific service-level agreements (SLAs) that the provider needs to offer to that tenant. By defining those SLAs at the virtual network layer only, it is very difficult to ensure those SLAs will be met because actual network performance occurs in the physical network.

Certainly providing an over-abundance of physical network capacity is one way to meet any given SLA, but this model is often not economical in a resource-on-demand environment such as a cloud. In addition, capacity alone does not define all of the resources a network can provide to applications. Capabilities such as latency, isolation and even explicit path enforcement are just a few of the many resources a properly equipped network can provide over and above simple capacity. By building these levers into the network, the upper layer logical entities, in this case the virtual network overlay, can have much greater control over how individual virtual networks are presented and manifest.

**Conclusion**

While there are many different approaches to implementing software-defined networks, one trend is becoming very clear - users want a steering wheel that lets them drive and operate the network rather than cede control to distributed protocols.

New OpenFlow-based architectures provide guidance of how to build new networks that have the fundamentals for providing this level of control. But simply stopping at the flow-forwarding API is not by itself sufficient. Virtual Network Overlays provide some additional flexibility and logical network segregation, but still rely on a legacy physical network, which in most cases has no ability to offer these levers.

Affinity Networking delivers the architectural benefits of OpenFlow and enables the functional benefits of VNOs by providing direct intent-based networks. The ability to define a set of behaviors based on groupings of resources and associated policies gives users a powerful steering wheel to drive network performance and resource allocation in a way that is directly correlated to application needs.

As both OpenFlow and VNOs mature, the ability of both the data plane and the control plane infrastructure to define and execute a set of desired intents will become increasingly important as one of the key values of software-defined networks.