SOFTWARE DEFINED NETWORKING AND CLOUD – ENABLING GREATER FLEXIBILITY FOR CABLE OPERATORS

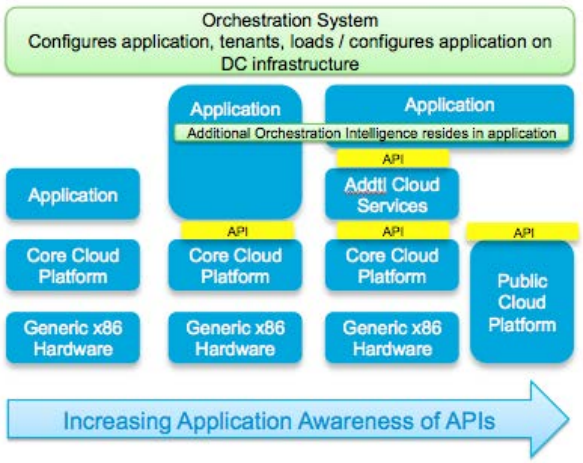
David Lively
Cisco

Abstract

Software-defined networking (SDN) promises tremendous flexibility for operators, especially as their application environments become more dynamic with cloud computing. As more content is becoming available to consumers via IP every day – from sources both internal and external – managing traffic flows for content and for the associated bandwidth across both the regional network is becoming more critical. Gaining real-time access to network analytics allows the network to optimize routing when connecting clients to content. This paper focuses on the content delivery use case, and discusses how cloud computing and SDN can work independently and together to deliver the highest quality experience to their customers.

CLOUD AND SDN – DIFFERENT BUT THE SAME

Cloud computing has delivered tremendous flexibility for application developers. The general definition of "cloud" is resources, abstracted from the physical hardware, delivered elastically and on demand. Because of this abstraction, applications can be deployed and scaled without the need to physically configure and deploy new hardware. Initially, this flexibility was primarily under the control of the administrators, and presented to users through portals to request services. Traditional and legacy applications could benefit from this newer, more flexible deployment model, but the real breakthroughs started to come when new applications were developed to call APIs exposed by the cloud platform directly. Subsequently, intelligence was written into the application itself to call APIs from the cloud platform to allocate additional resources when the application needed, and to turn them off when they were no longer needed. The cloud model initially applied mostly to compute, with the virtual machine, but has since expanded to include network and multiple different storage models (object storage, volume storage, etc.), as well as higher level services such as database services and security. These services further abstract the physical infrastructure beyond compute and storage to services that the application can directly use.



This model of higher layer services abstracted from the physical infrastructure starts to more closely resemble software-defined networking. Constructs have existed in networking for some time to allow a single physical router or switch to be segmented into multiple virtual and logical routers or switches – VLANs, VRFs, etc. SDN takes this concept one step further by providing the ability to define and request resources across a larger scope of the network. As applications became decoupled from the physical infrastructure and were able to move, grow,

and contract as needed, the network connectivity between them needed to move, grow, and contract accordingly, and automatically, with the application. A heavily virtualized application world, coupled with increasingly mobile users, means that we don't know when the network flows are going to come in or where they're going to come from. Leveraging business intelligence to help drive network behavior through SDN can help the network deliver the flexibility and adaptability to handle these new dynamic flows.

GAINING THE MOST FROM SDN WITH BETTER NETWORK INTELLIGENCE

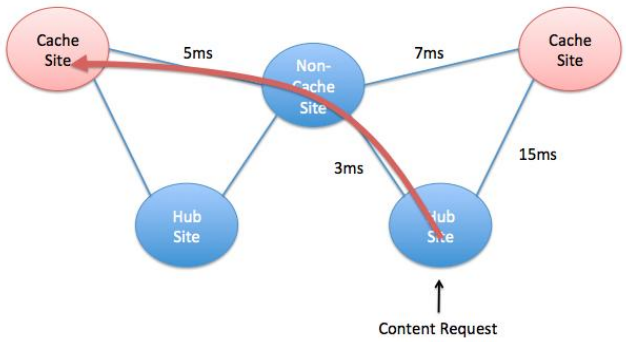
Consumer devices are accessing content from the network from both fixed and mobile locations. As fixed and wireless broadband speeds increase, this increase in the number and range of devices, along with the average bandwidth of the content flows, is having a dramatic affect on the network and its available bandwidth. Software-defined network can potentially help optimize flows on the network, but only if the SDN controller has access to information about the state of the network. Traditionally, this data has been collected by the network and stored for later analysis, with access to information delayed on the order of minutes, or even hours / days depending on the analysis being done. While access to historical data can be tremendously beneficial for future network planning, the ability to analyze the data in real time and make decisions on that analysis can have a dramatic affect on the performance of bandwidth or delay-sensitive applications such as video. For example, many operators are using caching techniques with CDNs to minimize the impact of the increasing content streaming on their networks, but as more and more users are accessing content from a greater range of locations, network intelligence can be used to further optimize those requests.

POTENTIAL SDN USE CASE – OPTIMIZED REQUEST ROUTING IN DYNAMIC CDNS

Routing content requests to the cache that optimizes the end-user's experience while at the same time minimizing the impact on the network requires consideration of multiple factors from the physical layers of the network up through to the application layer. In addition to determining which caches have available capacity for handling incoming requests, the state of the network can be utilized to optimize the experience.

Determining the optimal cache to source content from

When choosing which cache to route a content request to, the network and application must consider more than just the "closest" cache from a network or hop count perspective. When multiple caches are available over disparate paths, the closest cache may be sub-optimal from a network standpoint for delivering the video. Latency and jitter can play a large role in the end user's viewing experience, causing buffering issues, etc. Available bandwidth is another consideration. A network link with low latency and a shorter routing path, metrics normally considered by dynamic routing protocols, might still provide an unsatisfactory viewing experience for the end user, especially for single bitrate streams that are not able to adjust to congestion conditions.



Latency-Optimized Cache Choice

Additionally, if the initial cache request is known or assumed to be part of a longer duration flow, that can also be taken into account when routing requests to the optimal cache. The SDN controller could ask the routers to add timestamp information to packets and start building a database of latency information between different nodes, and along different paths between the same two nodes. Once the operator has obtained quantitative analytics, such as latency, from the network, they now have the ability to define policies that leverage that business-level intelligence to choose the optimal location for routing content requests, and ultimately to program the network itself in response.

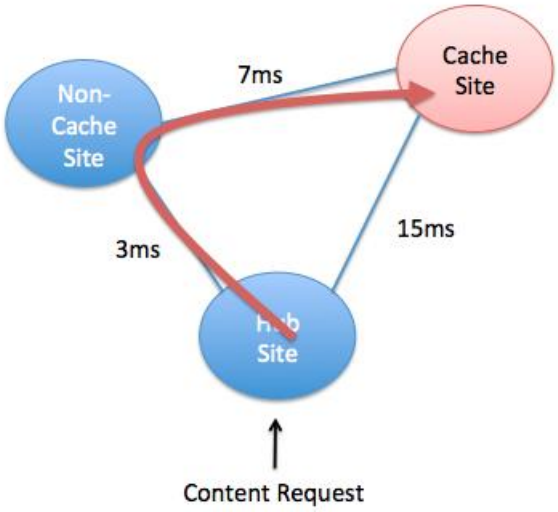
SDN Options for Content Requests

In addition to using network intelligence to determine the optimal cache to route requests to, we also have the option with SDN to specify and optimize the routing of specific flows to be different from what typical routing protocols would specify. We will discuss two potential approaches for using SDN technologies in this CDN use case. The first involves "forcing" the content flow to take a different path to the cache than would be chosen by the network using standard routing metrics. The second approach focuses on optimizing the primary path for the content flow by affecting the routing or handling of other traffic on the primary path. These two scenarios can also apply outside of CDN use cases, and can be used for any scenario where multiple paths exist to deliver content, and where the path that is most optimal from a routing protocol perspective is not the most optimal for the viewing experience.

Modifying the path for a Specific Content Flow

There are many conventional methods that network engineers use today to modify the path of particular flows through the network based on specific policies. Engineers can influence next-hop behavior for packets, can use traffic engineering to define specific paths through the network, can used QoS marking to specify specific output queues for traffic, etc. However, all of these methods typically require pre-configuring specific features on the devices in the network and won't scale to handle large numbers of per-flow requests on demand. SDN provides us with a way to programmatically have the network move the flow to a different path through the network, and then remove those routes or override metrics once the flow is done.

In this use case, we are going to assume that the primary path to the cache has become congested, or is undesirable for other reasons, and a secondary path exists that meets the required policies exists. These policies could be network-oriented (such as bandwidth requirements, delay requirements, etc.), or could even be business-oriented policies. Perhaps some links and routes are being leased from other carriers or have more expensive peering costs, so even financial decisions can be used to influence traffic patterns.



Latency-Optimized Path Choice

Thus, for a specific flow or class of flows, we want to have the routers send the traffic via an alternate path. We do not want to alter the path for any of the other traffic, so we can't change the general routing metrics for the links overall for all traffic. The first step is to identify a particular flow, or class of flows, and there are multiple mechanisms that exist today to do this. In the most trivial scenario, a flow could simply be identified by source and destination IP address. In practice in cable networks, many end-points are behind NAT devices such as home routers, and many end points may look like a single IP end point. In this case, the operator can get more granular and identify a flow by source and destination IP, plus flow type as inferred by the source and destination TCP / UDP ports. These are both readily doable using information in the packet header itself, and routers can generally make routing decisions in hardware using information from the standard header. You can also get even more granular and use deep packet inspection techniques to identify a particular flow.

Once you've identified a particular flow, you need to process packets in that flow according to the technique being used to modify the routing of the packets. This could be done by encapsulating the packets with new header information and placing them into a specific tunnel across the network using GRE or MPLS. They could also be "tagged" with a new service header or identifier in the packet header that is specific to the service. Minimally they could simply be assigned to a policy that governs their next-hop routing interface.

One method to leverage the programmatic capabilities of SDN would be to use the controller to dynamically program ACLs into the routers, which would be used to map packets to the appropriate policy. An external SDN controller would have the broader view of the overall network, including where the caches are located, and what various network metrics such as latency, delay, bandwidth, etc. look like on each segment. When a request for content is made, the controller can determine the optimal route to the selected cache.

Another method would be to use the service header described above and have routers assign packets to a particular path based on the service header. These "service paths" through the network can be programmed into the routers from the SDN controller. Since the service paths only need to exist for the duration of the flow, they can be added and removed as needed by the SDN controller.

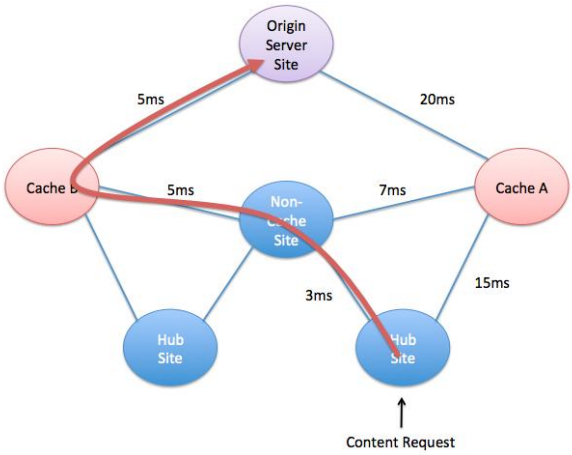
Optimizing the Primary Path by Dealing with Other Traffic

Similarly to moving a single flow or class of flows to an alternate path, it's possible to optimize the "favored" path for a flow by using SDN to modify the forwarding and QoS behavior for other types of traffic. The simplest method is to simply mark the priority flows such that they are processed by higher priority queues on a device. However, in some serious congestion scenarios it may not be desirable to simply drop lower priority traffic, but instead re-route that traffic over a secondary path. In this case you could use the same technique described above for a single content flow, but use it instead to classify "other" traffic that is less bandwidth and delay sensitive (or a lower priority for any number of other reasons) and to modify the routing metrics or path selection for that overall class of flows. In this way, the overall traffic on the link goes down, providing a better experience for flows that need to stay on the link.

Other Factors – Path to Origin Server

In a caching scenario, it's not always the path from the user to the cache that needs to be optimized, but the path from the cache to

the content origin server. In the case that high amounts of new content suddenly need to be cached (such as a new viral video or other event), the paths from the origin servers to the caches may become the bottlenecks for performance. A cache (cache 'A') that looked ideal when evaluating the network performance and load between the cache and the user may no longer be optimal if all of the content needs to be source from the origin server, and the path between the origin server and cache 'A' is highly congested. In this case, cache 'B' might be a better cache if the path from it to the origin server is better able to handle the load, and if the path from cache 'B' to the end user can support the required policies.



Latency-Optimized Cache Choice Considering Path to Origin Server

SDN for Modifying Flows in CDNs

As discussed at the beginning of this section, many approaches already exist today for both classifying packets, and modifying the routing of those packets through the network on a per-flow basis. The various approaches use existing routing protocols, policies, tunneling technologies, etc. However, most, if not all, of these approaches would be impossible to do dynamically, on a per-flow basis (as requests are made) without having both access to real time network analytics to make the decision as well as programmatic interfaces to the routers to apply the changes. SDN provides us with a way to have a controller with a broader view of the network, including non-network analytics such as business policies that can access devices in the network through programmatic interfaces to modify packet routing behavior.

LEVERAGING CLOUD COMPUTING FOR DYNAMIC APPLICATIONS

While networks are being enhanced to take more advantage of the dynamic capabilities of programmatic interfaces with the network, and enhanced analytics, the applications themselves are becoming more dynamic and programmed to take advantage of this additional flexibility in the network. By working together, applications can take advantage not only of the cloud computing capabilities of spinning up additional application instances on demand, but also of the intelligence available in the network to spin up resources in the best location when multiple are available.

Caching Example – Using Cloud for the Spikes

Referring back to the CDN example discussed in the CDN section, one of the first things that the CDN could check is whether or not a cache has enough capacity to support servicing incoming content requests. Caches are typically designed and engineered to handle typical loads on the network plus a predicted amount of burst capacity, based on analyzing historical data such as number of consumers in a given area, type consumer viewing behavior, busy hour requests, etc. While this will handle the vast majority of requests for content, there exist both seasonal "expected" dramatic increases in content viewing (such as around major sporting events like the Super Bowl or the World Cup) as well as dramatic unexpected spikes in viewing for things like suddenly viral videos.

If the network operator has built enough caching capacity to handle these short term, dramatically spiked increases in capacity, much of the caching capacity would sit idle and wasted the rest of the time. Since many of the caching servers are now able to be run as virtual machines on standard x86 hardware, cloud computing technologies can be used to "spin up" additional caches for both expected and unexpected events, and can then be brought back down to allow the compute resources to be used for other applications that see spikes in demand at different times, or whose usage is not as time-critical as video delivery. This capacity can also be brought up at locations where there is excess bandwidth to handle the additional load being placed upon the network. Operators can both maximize the number of content requests that can be handled by caches, and also minimize the impact on the network by locating those caches in the most strategic locations from a network perspective.

As additional caching capacity is brought online, it can dynamically be added to controller that is assigning content requests to caches. Additionally, the additional capacity is made known to the SDN controller(s) which can then take those caches into account (as well as their locations in the network) when determining how to deal with new flows.

Geographic location can also be used as a part of the decision process. For example, if there is an unexpected spike in content requests during prime time television viewing on the east coast of the United States due to a promotion, additional capacity could be proactively brought online in the central and west coast regions to handle the anticipated increased demand. Additionally, content requests could be sent to the new and existing caches to allow them to start caching content ahead of anticipated demand. If the demand did not materialize as expected, the additional caches that were started up can be shut down, with their compute resources being returned the available pool, and existing caches will simply age out the content as other requests come in.

LEVERAGING CLOUD COMPUTING APPLICATION DESIGN PRINICPALS IN NETWORKING

Cloud computing capabilities are helping to bring about complementary capabilities in networking. Cloud computing is further enhanced when it starts taking advantage of the new APIs available from the network, and using network location and state information in its decision-making process on when and where to spin up new capacity for applications. Additionally, the software architecture design principals being used by new applications being developed specifically for the cloud can provide additional hints on how to look at network design for the cloud. Several design principals are emerging, including:

- Fault-tolerant design that is adaptive and programmable through APIs
- Dynamic instantiation of new services and capabilities for the application
- Scaling out (adding additional nodes for a distributed function) vs. scaling up (adding compute or storage capacity to individual nodes)
- Heavy focus on automation

The new application design principals are both influencing network designs (including SDN) as well as being designed to take advantage of the more programmable nature and design of today's networks.

Fault-Tolerant Designs

Cloud applications are being architected such that the application will continue to run even in the event of physical infrastructure failures. Failures of servers or server components, networking gear, and even physical data centers won't bring down well-

designed cloud applications. The applications are designed using multiple loosely coupled components that are distributed across multiple different availability zones. Typically each component can be loaded dynamically as needed, and scaled up or down as needed. All of this is starting to translate to network design, as networks need to be able to accommodate application communication across multiple servers, racks, and entire data centers. When loads shift due to outages, networks need to be able to quickly accommodate new flows, and to rapidly apply changes in network policy.

Dynamic Instantiation of New Services

Many applications need to leverage network-based load balancing to help scale individual application components and distribute loads across availability zones. As the application scales out, additional load balancers need to be dynamically instantiated and added into the architecture to handle the new node addresses. As load drops, load balancers can be taken offline, with their resources freed up for different applications or services.

Newer video applications being delivered from the cloud can also take advantage of the dynamic instantiation of services. For example, when a user requests a particular piece of content, that content may need to pass through a just-in-time transcoding or encryption application. As these applications are developed to be loaded dynamically in a cloud computing platform, a base amount of capacity can be kept online, with additional capacity being brought online in "standby" mode as loads increase. Understanding of network analytics is important for network-based services such as these that require significant bandwidth, and can greatly benefit from intelligent placement or routing due to the bandwidth demands.

Scaling Out vs. Scaling up

The historical approach to handle capacity growth, called vertical scale or scaling up, involves turning up an application on bigger and faster hardware to handle a larger load. Modern cloud applications are being developed to handle increased loads by scaling out - adding additional instances of the application on additional, smaller compute nodes as load increases and balancing between them. Networks are now often being designed in similar fashion, with smaller fault-domains per switch pair, and scaling through the addition of more switch-pairs. In this fashion, network capacity can be added as compute and storage capacity is added without increasing the risk of a single failure taking out mission critical applications.

Heavy Focus on Automation

Finally, applications would not be able to scale dynamically without a heavy focus on automating every step of the process. Manually adding or removing capacity, or bringing up new services for the application can take significant time, and has significantly more potential for errors to be introduced into the process through manual data entry. This applies to all aspects of the application and its deployment, from the compute to the storage to the network connectivity, network services, and ultimately application configuration itself. The other half of the automation focus is monitoring. Without the ability to accurately monitor and measure application and network loads, it would be impossible to know when to scale up or down capacity. Programmatic APIs make both monitoring and automation possible. Application designers, as well as network designers, set the business policies and rules up front, and then let the automation systems take care of things from there.

CLOUD AND SDN – WORKING TOGETHER FOR GREATER CONTROL AND FLEXIBILITY

Advances in cloud compute brought about significant improvements in both the speed and flexibility of application deployment by abstracting the resources used by the application from the physical infrastructure providing those resources. This abstraction helped bring about the ability to dynamically bring up and down resources as needed, without the need to configure physical hardware. As developers learned how to use this flexibility, software architectures started evolving to take advantage of the new capabilities. This new generation of applications and architecture thinking helped bring out similar changes in network applications and design. The introduction of programmatic interfaces on network devices enables software to define network behavior, utilizing business intelligence and policies in lieu of just network metrics. In some cases the intelligence resides within the application itself, and the application is able to program and control the network directly. In other cases the intelligence resides within a specialized SDN controller that acts as an abstraction between the application and the network.

However, before you can take full advantage of the network abstractions and programmatic capabilities available with SDN, you need to start with monitoring and visibility of the network – analytics. Once you have a view of what the historic and real-time state of the network is, and you understand the business rules required by the application, you can use the SDN controller to enact custom routing on the network.

It starts with setting policy in the controller, which communicates the policy down to the network devices who enforce it by applying new forwarding rules, encapsulating packets into tunnels, etc. As the application developers start to understand the network analytics and abstractions available, they can use business rules to define policies in the controller and let the controller define network routing based on those policies.

While SDN adds flexibility, the use cases we have talked about in this paper are more about using SDN to manage the exceptions rather than replacing standard network protocols and router control planes altogether. The network is already highly adept at routing and moving packets based on network state. But when applications need higher-level business intelligence to make a better decision for the application, SDN can be used to program the exceptions that need specific or different handling. Exception-based SDN use cases allow the network to continue to do what it does best, and allow the SDN controller (and ultimately the application) to define behavior for specific flows, such as content flows within CDNs.