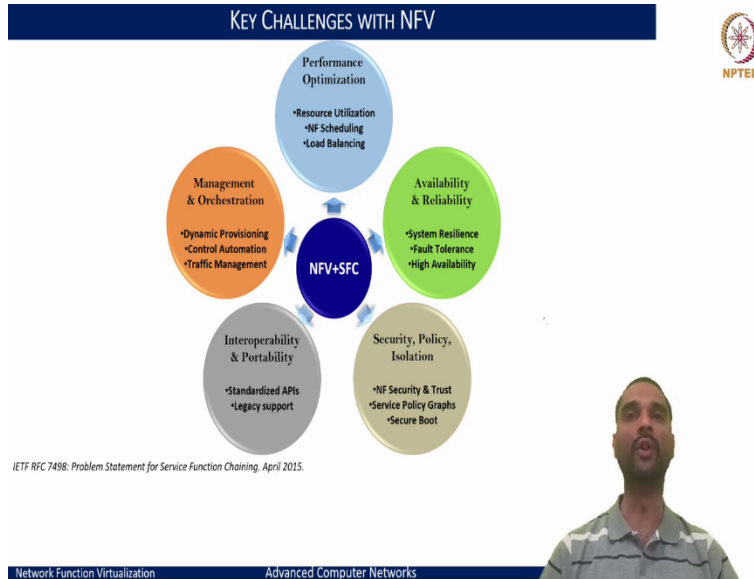


Advanced Computer Networks
Professor Doctor Sameer Kulkarni
Department of Computer Science Engineering
Indian Institute of Technology Gandhinagar
Lecture 41

Network Function Virtualization – Road Ahead and Key Challenges

(Refer Slide Time: 0:18)



So far, we looked at what is NFV and what were the standards, the frameworks, and all the positive sides of how NFV is able to help decouple and deossify the network infrastructure for the telecommunication and operator networks. Now let us try to look at the other side of the coin, that is although we say that NFV promises a lot of benefits. When it comes to real-world deployment, it also brings along a lot of challenges with it. And as I mentioned before, these NF services, NFV would be deployed as NF services that are used to build typically the network services which include a chain of network functions. So, for the practical deployment purpose, we need to look at an NFV infrastructure that is going to constitute of multiple of these virtualized network functions, and then they are chained to provide a set of services, as we show or discussed before.

So, from this perspective, let us try to see what are the challenges that we see with the real-world deployment of this NFV and service function chains. First is the management and orchestration. So, we did discuss about how the NFV mono framework tries to help automate or softwarize the aspect of management and orchestration.

But when we look into the details of what it means to really manage and orchestrate these network functions and chain them together, we need to think of trying to associate this with the real-time workload. So, when the workloads change, that is the network traffics, which is dynamics that keep changing, and also the systems, the infrastructure that we have, we may see the ups and downs. So, whenever there are any changes either onto the infrastructure or onto the traffic characteristics or the demand perspective, we need to provision the resources on demand.

And this is where a dynamic provisioning aspect needs to be supported. How exactly to support? What is the time scale and which we need to provision and how do we really automate such control and how do we monitor and manage this traffic so that we are able to do these updates in time? All are the associated aspects that need to be relooked and have to be worked in tandem, make sure that NFV and SFC as a thing can succeed. Hence management and orchestration, in essence, require a lot of efforts to make this happen in real-world deployment.

And second most critical aspect is this performance optimization, and when we speak of performance optimization, we are talking about how to schedule the resources. How to load balance the traffic on these network functions and how to maximize or optimize on the resource utilization so that you are neither over provisioning nor under provisioning the systems.

So, all these aspects need a major consideration, especially when we move away from the hardware infrastructure which used to process the packets to a software function that now needs to process a packet performance optimization becomes the most essential aspect to look at.

Third is the availability and reliability aspects, and when we see about availability, it is about how much of the time the system is up and what is the guarantee that it is providing in terms of in a given day or in a given month or a year, how much of the time is it always up. Is it 100%, or is it lower than that and what is the value that defines the availability and reliability in terms of whenever there is a failure, like whenever the service fails, how quickly can we make sure that the data is durable?

Whenever it fails, we ensure that the data and the applications have their states saved and are able to come back up properly, ensuring that there is no loss of user information or no loss of functionality, and that is about reliability.

And in these aspects, we need to understand how we can build the NFV infrastructure, be it the physical or the virtual appliances that we spoke about, to have resilience for the failures so that we are able to accommodate the failures and have a tolerance for any of the faults that occur in the system and ensure the availability of the systems for the majority of the time and these aspects need to be relooked because now we are having a one more glued component, these are virtual infrastructures that are all running on commodity hardware. Not the carrier-grade dedicated or proprietary network appliances that we thought of and saw before.

And the other crucial aspects when it comes to the challenges which NFV is, in fact, security, policy, and isolation. On one side where whenever you want to deploy because now we have said that these software functions are going to be built by independent software vendors, there has to be an implicit trust in which of the network functions can be really deployed that means like when we think of having these android apps that we build and see it on our google play store there has to be a trust in terms of whatever the apps that are being put are really trustworthy to be run otherwise they may jeopardize the entire security of the system and hence with NFV infrastructure, with this virtualization that we are bringing and talking about it becomes more important, also play emphasis on security on this newer layer rather than just at the infrastructure layer itself.

And hence the security and trust and how to build this framework or ecosystem becomes a major aspect to look at. The other side is also the policy graphs because now, when we said we want to realize a network service, we will typically build a policy graph spanning multiple network functions, but these network functions may be developed and deployed by different software vendors.

In a sense, now we need to make sure that we are able to make these two talks and work together, and that is where the interoperability and portability aspects also come into shape, and there is often confusion that I hear about when it stopped spoken about interoperability and portability. So, let me try to put a perspective of how these two, although coupled are different aspects.

First, when we want to define interoperability, it is about the capability of two or more functional units that can process the data in a cooperative fashion. What that means is we may think of having two different network functions, maybe from different independent vendors, maybe on

different locations, but when they want to update or function together as a chain, we need to ensure whatever the data they are going to process and whatever the communication they are going to have they should be able to understand each other and work together. And that is about interoperability.

So, if you think of like the most common scenarios that we may have is, we may have a private cloud, and we may have a public cloud. And when we deploy some of these services on private cloud and some of the services on public cloud like Amazon, Google, or even on multiple of the public clouds that we may deploy these services, we want the services that are deployed on Amazon to be able to talk and work with the functions or services that are going to be run on Google and likewise with our private infrastructure. That is about interoperability.

The other aspect is portability which is defined as basically a capability of a program that we want to execute on our machine or our node to be able to do the functionality in the same way when it is moved or shipped to another instance. And that is where the aspect of the virtual machine comes in.

But think of this as what data it wants to use and what functions you want to use. So, portability can be defined in aspects of either data portability or function portability, and for us, when we have the network functions which contain both states, which is data and functionality that is operating on the packets and updating of these states, we want to ensure that a network function that is built for one of the NFV architecture as we discussed about OPNFV. So, if I have deployed a network function on OPNFV-based architecture and am able to run that function, I should be able to also run it on my private network function with much of ease without having to modify any of the aspects revolving the core functionality of the network appliance that we are building.

Although they may require some custom operation level of updates for very little modifications in terms of configurations but the functionality as such should be able to work on either of the deployments of the frameworks or of NFV frameworks. So, these are, in a nutshell the key challenges that revolve around the NFV.

(Refer Slide Time: 9:03)

CHALLENGES WITH VIRTUALIZED NETWORK FUNCTION CHAINS

NFV+SFC

Performance & Scale


Management & Orchestration

High Availability & Fault Tolerance


	Middleboxes		Standard Server Machines
	CG-NAT [3]	Firewall [4]	
Performance (Throughput)	320 Gbps	640 Gbps	10-40Gbps
Scale (Connections)	480 Million	576 Million	< 1 Million
Availability(%)	99.999	99.999	< 99.9

- Standard Servers => No or limited hardware accelerations:
 - Hard to meet Telecommunication grade performance & scale requirements [1].
- Virtualized network services => Provisioning and Orchestration
 - New mechanism to control, orchestrate and manage network services [1,2].
- Standard Servers => More prone to hardware and software failures.
 - Hard to meet the Telecommunication grade availability requirements [1].

<http://www.f5.com/pdf/products/big-ip-cnat-datasheet.pdf> [4] <http://www.f5.com/pdf/products/big-ip-advanced-firewall-manager-datasheet.pdf>



Network Function Virtualization Advanced Computer Networks



Let us try to look at least the performance aspects in detail to understand what it means and what we mean by performance challenges and how the community has evolved over the last decade to address these performance challenges. And also try to look in a bit about the orchestration and reliability aspects.

So, first when I speak of performance and scale, what we are really trying to show here is we are moving away from dedicated hardware or the middleboxes that we discussed earlier, which were the hardware infrastructures of the network elements. And now, we are implementing the same as software functions on these commodity hardware for the standard server machines. So, we need to now compare how devices that were the dedicated hardware appliances used to work versus how they would be behaving on the standard software machines.

And what we can see here is that when we think of these carrier-grade NATS as a middlebox that were deployed on the network infrastructures. They were able to support around 320 gigabits per second worth of processing capabilities and were able to scale up to 480 million connections worth of data that they could store.

So, now when we want to move this to a standard server machine, we have to see are we able to keep up with this rate and, likewise with the firewalls that dedicated firewalls from F5, another company that used to ship, they are able to support almost around 640 gigabits per second performance, that is, packet processing capabilities would go about 640 gigabits per second. And

also, they are able to maintain the connection and scale up to around 576 million connections worth of information they can maintain and be able to process simultaneously.

But now, when we move to the standard server, think of these are basically the server-grade processors that we are deploying that would be like 2 gigahertz to 3 gigahertz processing capability with again the commodity memory that we would plug in which maybe again span few gigabytes.

And now, when we want to use the network appliances that are running on top of it, what we start to see the network is facilitated through the ethernet switches that ethernet first high-speed processing ethernet that we may have, and what we see here is roughly the order of 10 to 40 gigabits which is a standard. Now we have moved all the way up to few hundred gigabits per second in terms of the standard server machines able to have these ethernets and infinity band support.

But this is still order less than what we would expect from the middleboxes, and likewise, when we want to store the information of or scale the number of connections, it really becomes limited in terms of what is the server address space that we can provide, what is the state that we can build on each of these either in a virtual appliance or in a container model and that seemed to be very limited in fact, and this is where the performance and scale aspects demand, innovations in terms of how we can overcome and adapt this.

And a very simple way that we can think, if we have just one middlebox, a carrier-grade NAT or a firewall which used to cost around 20 or 30 thousand of dollars, now we are replacing that with the standard server machine, which is around less than a thousand dollars. We can scale equally the same number of instances.

So, instead of one carrier-grade NAT, I can think of having 4 to 5 standard server machines, each with 100 Gbps support. In a way that we are now going to manage multiple of these devices, so that becomes another challenge although we could address one way of making it to be worked out with scaling the individual instances or what we call as a scale-out but when we scale, we also have to manage and orchestrate these network functions on which server would it run, where would it run, how do you plan to build this aspect of provisioning the resources and trying to see where is the demand, all the bookkeeping or cataloging for all of these services and automation

become a major challenge with the management and orchestration of these scaled out standard server machines.

And most important, again is with respect to availability and fault tolerance. So, if we think of these middleboxes, which were carrier-grade NAT or firewalls, the telecommunications what is the requirement of availability of finite that is 99.999 percent of the time in a year these should be available.

But when we move out from these dedicated hardware to the standard server machines, which are known to have like less than three nines of availability time and this we see although it is now like big of a difference when we see three nines to five nines it is a huge 100x order of availability time that goes, that shifts when we move from standard middleboxes to the standard server machine.

So, we look into it this challenge also in a way, what it really means, but now we can see that it becomes really hard to meet the telecommunication grade availability requirements, and hence all of these aspects need to be relooked when we want to really deploy these NFV frameworks.

(Refer Slide Time: 14:36)

COMPUTE REQUIREMENT OF NETWORK WORKLOAD >>> COMPUTE CAPACITY		
Networking Workloads x VMs x Traffic >>> Compute		
Networking Workload	Approx. Compute Cycles/Packet	Achievable Throughput on (2Ghz) Processor
L2 Forwarding	75	26.6 Million Packets/sec
IP Routing	175	11.4 Million Packets/sec
L2-L4 Classification	750	2.6 Million Packets/sec
TCP Termination	1500	1.3 Million Packets/sec
Stateful Firewall	2250	0.8 Million Packets/sec
IDS/IPS	5000	0.4 Million Packets/sec
NextGen Firewall	8500	0.23 Million Packets/sec
IPSec/SSL	9500	0.21 Million Packets/sec
Firewall + SSL	18000	0.11 Million Packets/sec

So, let us look at the performance requirement and what it really means or how we can derive or try to get a first-cut attempt on how we can think of what is the performance requirements for these network functions. And we can characterize these network functions based on the network workloads and kind of network functionality.

So, what here in this table is trying to show is we have the L2 forwarding machine, which is the very basic switch, nothing else, and if we implement a switch on software in the machine be it, you can just bare physical process running on a bare metal device and if we measure the number of cycles that it takes to get a packet and move the packet out, provided the packet is already available, and you just want to put the packet out on the NIC and what we see is the computation involved not in terms of taking the packet in or moving the packet out.

Once you have the packet the network function which has to make a decision on which of the port that it has to send. Basically, the forwarding functionality. So, just the forwarding functionality when it is written as a code, what it really means is, you receive a packet on port 1, so you read what is the port on which the packet came or the incoming port, you look up in one of the tables basically the forwarding table to say which is the output port on which you want to push the packet out.

And just this functionality, when it is run on a CPU, it was observed to see roughly around 75 to 80 cycles that you need to process per packet to make a decision of where to forward this packet out. And if we have such a cycle requirement on the compute and if forwarding has a NFV instance running on one core.

Let us assume a 2 gigahertz processor, then how many of such packets can be really processed in a second matters a lot, and we can work it out to say that if I have 75 compute cycles and on a 2 gigahertz processor basically, you can see 1 over 2 gigahertz or almost around 50 nanoseconds that you would put per cycle and if we are spending 15 nanoseconds per cycle and we need 75 such cycles to process one packet; if we do the multiplication math and get the values we can see that the 75 cycles would account to around roughly helping us process around 26.6 million packets per second.

And these 26.6 million packets per second are good enough in terms of supporting around 20 gigabits worth of processing 20 Gbps support line with a very minimal packet size. And as the packet size vary these numbers could change.

So just be able to forwarding maybe a fit enough candidate to meet around 20 gigabits as we see on a single core, and as we scale the cores, maybe we will be able to do it better. But when we go to the real network appliances that we want, taking a layer 3 IP routing and suddenly the layer 3

routing which would require us to look at the IP destination at a minimum and do the destination-based forwarding, and if we have to do that then the number of cycles that we would need on the compute end increases drastically.

And with this, we will see that there is a drop almost half the drop in the throughput that we can achieve on commodity hardware. And as the functionality keeps growing, and that is where the middleboxes really do, like L2 to L4 classification, then you will see that there is a 5x increase in terms of the cycles that you would need just to do the compute part of how to classify.

Look at the layer 2, layer 3, and layer 4 headers, pass them, and then update the classification table or bits in the packet, and then you would drop down almost 10x times compared to L2 forwarding in terms of what is the throughput that we can achieve.

And further, like these middleboxes, like we said, do a lot of stateful firewalls wherein you have rules, you look up the table, you process, you drop the packets, or you allow the packets to forward, and we can even have a layer 7 processing where you are trying to process all the information up to the HTTP and try to do these functionalities for security functions like IDS and IPS where you are trying to proven for every packet and match specific signatures to see whether there is a need to halt the packet processing or forward it further.

All of these call for much more extensive compute cycles, and as the compute cycles keep growing, we can see that the packet rates drastically drop. That means the essence that we are trying to see here is that the compute cycles that we have are at prime, and the throughput that we can achieve is really becoming very, very constrained when we really want to look at what network workloads, we need to support for many of these forwarding functions or loading functions or any of the security other network functions that we can think of. Hence it becomes important to see how this can be optimized or what are the means where we can do things in a better way.

(Refer Slide Time: 20:03)

NFV PERFORMANCE CHALLENGES

Typical performance


- 3-4 Gbps per CPU core assuming very light per-packet processing
- An order of magnitude less than what the hardware could do (more than 10Gbps per core, 40+ Gbps per x86 server)

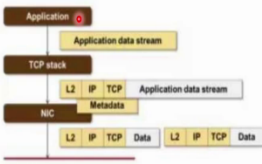
Bottlenecks

- TCP stack and Linux kernel in NFV virtual machines
- Hypervisor virtual switch
- NIC TCP offload works only with VLANs


Solutions

- Optimized virtual switches (example: Intel DPDK)
- Dedicated virtual NICs (hypervisor bypass)
- Dedicated packet processing CPU cores
- User-mode packet processing (example: PF_RING)





Source: Ivan Pepelnjak



Network Function Virtualization
Advanced Computer Networks

And if we look at NFV performance challenges, what we really see as opposed to what we just saw in the table. The typical packets may be around 200 to 500 bytes worth of exchange that really happen over the internet, and with that, the real performance that you really get is around 3 to 4 gigabits per second processing capability per core, and this is true for most lightweight packet processing which is much less than what the internet interface, the network interface which is 40 gigabits or 100 gigabits, allows.

So, if I have to catch up to 40 gigabits per second, I have to ensure that the network function utilizes multiples of the cores on a machine so that if it is you are able to process 3 to 4 gigabits per core, you would end up putting network function on 10 different cores at least to ensure that you are able to keep up to the 40 gigabits and thus we know that network interface speeds have increased drastically to around 400 gigabits per second as we speak today.

So, keeping up the processing of just one machine or multiple of the hardware also becomes really difficult. And why it is happening is, we need to understand and this was thoroughly studied in the early works of 2012-2014, and the primary bottlenecks were identified to be the TCP stack in the Linux kernel, and especially when we think of the virtualized framework, there is a hypervisor which is talking to the NIC, getting the packets and then passing it on to the virtual machines.

So, there is a two-way by-passing of the packets and processing of the interrupts and all of this. That really adds a lot of latency and processing overheads, and hence there were several of

solutions that were thought of and were uncovered and many of them being the most optimized method being the Intel DPDK.

Although DPDK was started by Intel and now it is under the Linux Foundation but this as a framework of these solutions work is an interesting aspect to understand in the context of NFV, and there was also a dedicated virtual mix which was meant to bypass the hypervisor, provide the packets directly to the VM's that were running the packets from the NIC. This also helped improve VMDQ, and SRIOV, where some of the aspects that were brought by Intel as hardware manufacturers and were also done by several of the network interface card manufacturers to support them.

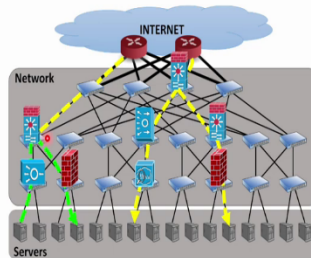
And we also will see that interrupt-based processing always incurs latency because you have to switch the context, get the processing, and then take the packets out. And instead, these alternatives were like dedicated packet processing CPU cores were added, which would avoid the inter-processing and do the polling on the packets whenever they arrive, or you are able to quickly process them and pass them.

And even in fact, it had a good impact on how the Linux APIs were built for sockets, and there were changes that were made called new Linux APIs that were being presented to provide faster packet processing capabilities in user mode, and we will see that PF_RING is one of the major things that came up with an API in the Linux, to facilitate user space packet processing.

We may not have the luxury to run through all of these, but let us try to get a glimpse of at least the most celebrated thing that is the DPDK framework, and understand how it helps address or evade some of the performance challenges that we spoke about.

(Refer Slide Time: 23:32)

NFV ORCHESTRATION CHALLENGES



- NF chain resource management and orchestration:
 - When and Where to Place/Instantiate/Consolidate NF?
 - Where (which NF instance) to steer an arriving flow?
 - When and Where to redirect a flow?



Network Function Virtualization

Advanced Computer Networks

Next let us look at what we mean by orchestration challenges in a bit. So, this is how a typical infrastructure looks like when we have like the community data center or a mapping it to a typical enterprise or any other infrastructure. What we really have is the Edge gateways, routers that are connecting you to the internet, and what we have inside are basically the switches consider these as the virtualized instances that we may have built on our NFV framework on top of the physical servers. And they may be connected in a logical topology in one or the other fashion.

Now when we want to deploy the network functions on top of such infrastructure, what it really means is where would you place these network functions in the first. So, consider all of these network functions to be virtual instances. We may place them randomly anywhere on the service, but would that be fine and what really matters is, when these are the packet processing intermediaries, it would be better to say that we would have the functions wherein you are able to forward the packets in one direction rather than going back and forth.

So, if we had these functions we do not want to be ending up circulating the packets back and forth on the links to have lower efficiency of link utilizations. Instead, we would want these functions that we want to change placed in such a way that we are able to process them all at one place, or as we progress, we are going to have minimum links that we travel to pass these functions. Hence when and where to place these or instantiate or consolidate these NFVs when

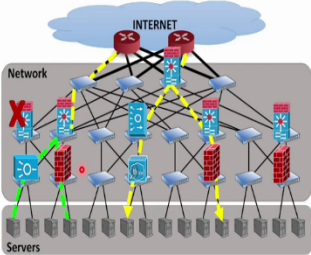
the workload is lower, all of these questions become a prime importance to address in the orchestration.

Second, typically like what we see is when the flows arrive through the router and then they are going to be processed, we have to make sure that a chain is built properly otherwise, like what I just mentioned, we would end up with having the challenges of unnecessary traffic that is going back and forth adding to the bandwidth tags on the given system. Hence how do we steer an arriving flow and ensure that we are able to also meet the policy requirements as we steer the packets becomes a major aspect.

And third, often, there will be traffic dynamics as well as the dynamics of the physical infrastructure. Some links, some updates need to be done, or some servers are seeing a high load in such a situation we need to shift the traffic. Hence how do we redirect the flows from one instance of a network function to the other instance so that we are able to balance the load becomes a major aspect. So, we need to look at all of these as the key orchestration challenges when it comes to your deployment of these network functions.


(Refer Slide Time: 26:22)

NFV RESILIENCY CHALLENGES




The diagram illustrates a network architecture where multiple servers at the bottom host various network functions (represented by blue and red icons). These functions are interconnected and connected to an 'INTERNET' cloud at the top. A red 'X' is placed over one of the network function instances, indicating a failure. Below the diagram, a list of challenges is provided.

- Failure Resiliency framework for NF and NF chains:
 - How to quickly detect NF (software) and Node, Link (Hardware) failures?
 - How to provide redundancy and efficient NF state migration?
 - How to provide correct and swift NF service failover?



Network Function VirtualizationAdvanced Computer Networks



Further, if we have addressed all of these challenges, the other imminent pressing need would be, how do we address the resiliency aspect? So, consider we have this kind of a framework, and there may be various things that can fail here. So, it could be the network function instance that we have brought that could fail, or it would be either of the links that we have that are connecting

these devices be it the physical links that we are speaking could be failing, there can be linked flaps that can happen and also infrastructure, the hardware that is hosting many of the servers itself may fail.

So, we may see the failures at different levels. So, the foremost thing now becomes, whenever there are failures, we have to be having the ability to detect such failures in a real quick time, and only when we are able to detect and identify and isolate what are the actual failures, we will be able to recover. So, recovery means that we would want to have alternate instances that are able to take the load of whatever the field instances were managing.

And we spoke that these network functions, unlike the network elements where there is no state, here they are full of state, that means whatever the flows and connections that they were keeping the information, we need to ensure that there is a backup, there is an alternate instance that also knows precisely what is the state at which the previous instance was working before it failed. Hence redundancy through replication becomes an important aspect to work out that we are able to carry forward the state and work even in the instance of failures of any network functions.

And that is not just enough because if we have a state, but we also need to now migrate the flows from the failed instance to a new instance, and all of this means that whatever the state that was updated, the replication has to be in real-time and the state loss should be almost 0 or at best some minimal state which needs to be recovered in real-time again and the redirection that is steering of the flows towards the newly updated or redundant infrastructure needs to be also done in real-time. So, how to provide the correct and swift NFV service failure becomes a major challenge when it comes to the real deployment of resilient NFV frameworks.

(Refer Slide Time: 28:59)

CARRIER GRADE AVAILABILITY REQUIREMENTS				
Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	36.53 days	73.05 hours	16.80 hours	2.40 hours
99% ("two nines")	3.65 days	7.31 hours	1.68 hours	14.40 minutes
99.5%	1.83 days	3.65 hours	50.40 minutes	7.20 minutes
99.9% ("three nines")	8.77 hours	43.83 minutes	10.08 minutes	1.44 minutes
99.95%	4.38 hours	21.92 minutes	5.04 minutes	43.20 seconds
99.99% ("four nines")	52.60 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.995%	26.30 minutes	2.19 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	5.26 minutes	26.30 seconds	6.05 seconds	864.00 milliseconds
99.9999% ("six nines")	31.56 seconds	2.63 seconds	604.80 milliseconds	86.40 milliseconds
99.99999% ("seven nines")	3.16 seconds	262.98 milliseconds	60.48 milliseconds	8.64 milliseconds
99.999999% ("eight nines")	315.58 milliseconds	26.30 milliseconds	6.05 milliseconds	864.00 microseconds
99.9999999% ("nine nines")	31.56 milliseconds	2.63 milliseconds	604.80 microseconds	86.40 microseconds



So, overall what we are seeing when we think of this resiliency we have to meet the availability requirements, and what those availability requirements really boil down to is the drastic difference that we see between the server-grade machines that we operate in a virtualized infrastructure to the carrier-grade requirements that they impose when you see from the telecom operators networks where you are running with the dedicated hardware.

To put it in context, what a carrier-grade for a carrier-class refers to is a system or a hardware or software, a combination of the two or any component that is being put in a infrastructure needs to be extremely reliable and in the sense that these systems have to be thoroughly tested and engineered to meet the requirements of what we call as five 9s of availability.

And the five 9s of availability means this 99.999% time, the device, the software or hardware that component needs to be up, and what does it really mean is if we take a downtime per year, it should be less than 6 minutes down within a given year, or if we take it per day it has to be up 24 hours bearing at a burst about 864 milliseconds of downtime in a given day, that is less than one second of downtime that you expect in a given day.

But these are possible with the carrier-grade proprietary hardware that were built, tested over the years, and then deployed, but now, when we move to the server-grade machines, I said earlier that the server-grade machines at best offer about 99.9% of the availability. While the carrier-grade requirement is at least five nines, which can be anything further up from five nines but nothing below.

So, now we are operating when you virtualized and running it on commodity hardware; our availability requirements from the hardware point of view have really gone down, that is, having around 99.9% of the times available only. Now if you compare the same with the down times that you see, this is 5.26 minutes per year as opposed to 8.77 hours.

So, if you try to put this in perspective, you can see that this is almost 100x, that is, if you take a downtime per week, you can see that it is around 10.08 minutes, that is, around 600 odd seconds, while if you take the finite availability which is the minimum class that we need to meet that is around 6 seconds of a down time in a week. So, you can see that there is a 100x shift with this 99.9, that is, three nines to five nines. And we are speaking of the range, which can be much below this when we are talking of the commodity hardware.

So, it becomes a major challenge in terms of how do we make these systems available, and the only means is basically you ensure that there is enough redundancy and we can fail over to the instances much quickly. But when it comes to carrier-grade machines, it also imposes a constraint of the fault recovery through redundancy models, and what it really says is, if I fail at instant x, I should be able to recover to instance y within 50 milliseconds of time, and that is the carrier-grade or carrier class requirements.

These are very essential when we speak from the telecommunication networks perspective, but now meeting the same requirements with the commodity hardware becomes a major concern. So, we need to say now how we can ensure the redundancy that is fine. Besides redundancy, we also need to ensure that we are able to recover our boot-up in less than 50 milliseconds.

If you take your Linux machines as of today and you want to boot it up, it takes the order of a few seconds at the least, even with the very 10 minimal Linux kernel, and if we think of virtual machines that you want to run on a where minimal aspect, it would still take some around a second or less than a second to bring it but in the order of several milliseconds.

Even if you want to spawn a container, the containers take around hundreds of milliseconds to come up, and hence whatever the virtualization and deployment models that we saw make it really impractical if we have to bring the system on the fly. We have to have like a backup that is there and running so that you are only able to switch over in a short span of time, and that is where the unikernels also gained a lot of attraction in terms of how we can boot it up in less

around tens of milliseconds at first, and this is where also like several of the research focused on. What are the alternatives that if we containerize and keep the containers in active-active mode or active standby mode? What are the means to update the state quickly and ensure that you are able to start the processing in less than 50 milliseconds of time? These become major aspects also in terms of research which is still actively perceived, but these are the key challenges as we think of need to be addressed from the NFV point of view.