Programmierbare Netzwerke im Rechenzentrum

New Technology vs. Old Technology

Dirk Stöckmann, Dirk@cisco.com
Technical Solutions Architect - European Datacenter
October 2015
Agenda

• Digitale Transformation und die Evolution der IT
• Warum Programmierbare Netze?
• Wege der IT Transformation
• SDN Use Cases
• Programierbeispiel
Digitale Transformation und die Evolution der IT
The Social Networking and BYOD effect
A new Style of IT emerging

Operational Split of organizations

Every 60 seconds

- 98,000+ tweets
- 695,000 status updates
- 11 million instant messages
- 698,445 Google searches
- 168 million+ emails sent
- 1,820TB of data created
- 217 new mobile web users
Top Customer Challenges and Initiatives

- **Manage IT Change**
  - Cloud
  - Mobile
  - Social

  - “Shadow IT”
  - DevOps
  - Cloud-Scale
  - Flat Budgets

- **Optimize Cloud Strategy**
  - IT-as-a-Service

- **Organize and Analyze Data**
  - Big Data
  - IoT/IoE

- **Security and Compliance**
  - Securing Distributed Data and Applications

- **“Bi-Modal IT”**

- **Hybrid Clouds**

- **Data Insights**

- **Pervasive Security**
Die Evolution der IT

Herkömmliche IT
- Interne dedizierte IT
- Geringer Mehrwert
- Zeitaufwendig
- IT als Kostenfaktor
- z.B. eMail, Datenbanken, Backup etc.

Business relevante IT
- IT as a Service
- Social, Mobile, Analytics und Cloud
- Cybersecurity
- Agile Bereitstellung von Applikationen und Services
- API-driven und offene Architektur

IT ermöglicht Business Outcome
- Neue Business Modelle
- Digitale Umsatzgenerierung
- Customer Experience
- “Was können wir tun, um das Risiko zu minimieren und den ROI zu maximieren?”

IT wird über Probleme wahrgenommen

IT als interner Provider und Innovationsgeber

Digitale Transformation
Warum Programmierbare Netze?
Evolution of Server Configuration

In the Old Days ...

Today

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
Evolution of Network Configuration

1990s

- CAT6K>enable
- CAT6K# config terminal
- CAT6K(config)# interface fastethernet 1/1
- CAT6K(config-if)# ip address 1.1.1.1 255.255.255.0
- CAT6K(config-if)# no shutdown
- CAT6K(config-if)# exit
- CAT6K(config-router)# network 1.1.1.0
- CAT6K(config-router)# exit
- CAT6K(config)# exit
- CAT6K# copy run start

Today

- NEXUS>enable
- NEXUS# config terminal
- NEXUS(config)# interface ethernet 1/1
- NEXUS(config-if)# no switchport
- NEXUS(config-if)# ip address 1.1.1.1 255.255.255.0
- NEXUS(config-if)# no shutdown
- NEXUS(config-if)# exit
- NEXUS(config)# feature eigrp
- NEXUS(config)# router eigrp Test1
- NEXUS(config)# interface ethernet 1/1
- NEXUS(config-if)# ip router eigrp Test1
- NEXUS(config-if)# no shutdown
- NEXUS(config-if)# end
- NEXUS# copy run start

We need to better manage network devices programmatically
Why Programmability is important?

Save Time  Human Error  Customize  Innovate
Cisco Visual Networking Index

VNI Mobile Forecast by 2019:

- There will be 5.2 billion global mobile users, up from 4.3 billion in 2014.
- There will be 11.5 billion mobile-ready devices and connections, more than 4 billion more than were in 2014.
- The average mobile connection speed will increase 2.4 fold, 1.7 Mbps in 2014 to 4.0 Mbps by 2019.
- Global mobile IP traffic will reach an annual run rate of 292 exabytes, up from 30 exabytes in 2014.
Wege der IT Transformation
Digitale Transformation erfordert ein neues IT-Modell
Dual-Speed IT (McKinsey), Bimodal IT (Gartner)

**System of Engagement**
- Produkteinführungszeit
- Kundenerlebnis
- Ertragssteigernd
- Schnelle Releasezyklen
- Kundeninteraktion
- “Trial and Error”

**System of Record**
- Robust
- Prozessgetrieben
- Kostengetrieben
- Langzeitplanung
- IT Compliance
- Kein Raum für Fehler

**APIs als “Integration and Agility Layer”**
By 2017, **75%** of IT organizations will have a bimodal capability. **50%** will make a mess of IT. **Bimodal IT is NOT** a "nice to have."
Bimodal IT = Marathonläufer + Sprinter

<table>
<thead>
<tr>
<th><strong>Mode 1: Traditional</strong></th>
<th><strong>Mode 2: Agile</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reliability</strong></td>
<td><strong>Goal</strong></td>
</tr>
<tr>
<td>Price for performance</td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td>Waterfall, V-Model, high-ceremony IID</td>
<td><strong>Approach</strong></td>
</tr>
<tr>
<td>Plan-driven, approval-based</td>
<td><strong>Governance</strong></td>
</tr>
<tr>
<td>Enterprise suppliers, long-term deals</td>
<td><strong>Sourcing</strong></td>
</tr>
<tr>
<td>Good at conventional process, projects</td>
<td><strong>Talent</strong></td>
</tr>
<tr>
<td>IT-centric, removed from customer</td>
<td><strong>Culture</strong></td>
</tr>
<tr>
<td>Long (months)</td>
<td><strong>Cycle Times</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Agility</strong></td>
</tr>
<tr>
<td></td>
<td>Revenue, brand, customer experience</td>
</tr>
<tr>
<td></td>
<td>Agile, Kanban, low-ceremony IID</td>
</tr>
<tr>
<td></td>
<td>Empirical, continuous, process-based</td>
</tr>
<tr>
<td></td>
<td>Small, new vendors, short-term deals</td>
</tr>
<tr>
<td></td>
<td>Good at new and uncertain projects</td>
</tr>
<tr>
<td></td>
<td>Business-centric, close to customer</td>
</tr>
<tr>
<td></td>
<td>Short (days, weeks)</td>
</tr>
</tbody>
</table>

Think Marathon Runner

Think Sprinter
Two Market Transitions – One DC Network

Applications

- Physical + Virtual Machines
- LXC / Docker Containers

PaaS

- HyperScale Data Centers
- Application Centric Infrastructure (ACI)

Infrastructure

- Traditional Data Center Networking
- DC Switching
- Network + Services Abstraction & Automation

Network + Services

Abstraction & Automation

Apps

Policy

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
App Development via DevOps is Changing the Behavior

- **Change** (speed, risk, return)
- **DevOps**
  - Systems Of Innovation
  - Systems Of Differentiation
  - Systems Of Record
  - ITIL (SysOps)
  - CRM, BPM
  - ERP, HR, GL
  - WEB 2.0

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
Fabric Evolution

- Network Architecture
- ‘Tenant’ management
- Network Management
- Automation
# Fabric Evolution

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>100Mb/1Gb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STP &amp; blocking ports</td>
</tr>
<tr>
<td></td>
<td>Flooding &amp; full learning</td>
</tr>
<tr>
<td></td>
<td>3 tiers/protocol dependent</td>
</tr>
<tr>
<td></td>
<td>Dual servers access</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>‘Tenant’ management</th>
<th>VLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VRF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Management</th>
<th>CLI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Every equipment have to be managed</td>
</tr>
<tr>
<td></td>
<td>Pre provisioning</td>
</tr>
</tbody>
</table>

| Automation          | ≈ Null     |

## "Old-School" Fabric

2008
# Fabric Evolution

## Network Architecture
- 100Mb/1Gb
- STP & blocking ports
- Flooding & full learning
- 3 tiers/protocol dependent
- Dual servers access
- 1/10Gb
- STP & vPC (Mobility)
- Flooding & full learning
- 3/2 tiers-Protocol dependent
- Convergence & FCoE

## ‘Tenant’ management
- VLAN
- VRF
- VLAN
- VRF

## Network Management
- CLI
- Every equipment have to be managed
- Pre provisioning
- CLI
- Accesses are passive for optimizing ops (FEX)
- Pre provisioning

## Automation
- ≈ Null
- Customization

### “Old-School” Fabric
- 2008

### Fabric Gen-1 (vPC)
- 2011
# Fabric Evolution

## Network Architecture
- 100Mb/1Gb
- STP & blocking ports
- Flooding & full learning
- 3 tiers/protocol dependent
- Dual servers access
- 1/10Gb
- STP & vPC (Mobility)
- Flooding & full learning
- 3/2 tiers-Protocol dependent
- Convergence & FCoE
- 1/10/40G
- L2+L3 routing (Mobility)
- Flooding & conversation
- No topology constraints
- Convergence & FCoE

## ‘Tenant’ management
- VLAN
- VRF
- VLAN
- VRF

## Network Management
- CLI
  - Every equipment have to be managed
  - Pre provisioning
- CLI
  - Accesses are passive for optimizing ops (FEX)
  - Pre provisioning
- CLI & GUI
  - Accesses are passive for optimizing ops
  - Pre provisioning

## Automation
- Customization
- Auto-provisioning
  - No auto-configuration
  - No VM/Servers integration

---

<table>
<thead>
<tr>
<th>Year</th>
<th>Fabric Gen-1 (vPC)</th>
<th>Fabric Gen-2 (FabricPath)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

*“Old-School” Fabric*

- Null
- Auto-provisioning
- No auto-configuration
- No VM/Servers integration

*Next Generation Fabric*
# Fabric Evolution

## Network Architecture
- 100Mb/1Gb
- STP & blocking ports
- Flooding & full learning
- 3 tiers/protocol dependent
- Dual servers access

## ‘Tenant’ management
- VLAN
- VRF

## Network Management
- CLI
  - Every equipment have to be managed
  - Pre provisioning

## Automation
- ≈ Null

### Old-School Fabrics

<table>
<thead>
<tr>
<th>Year</th>
<th>Fabric Gen-1</th>
<th>Fabric Gen-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>(vPC)</td>
<td>(FabricPath)</td>
</tr>
</tbody>
</table>

### Fabric Gen-1 (vPC)
- 1/10Gb
- STP & vPC (Mobility)
- Flooding & full learning
- 3/2 tiers-Protocol dependent
- Convergence & FCoE

### Fabric Gen-2 (FabricPath)
- 1/10/40G
- L2+L3 routing (Mobility)
- Flooding & conversation
- No topology constraints
- Convergence & FCoE

## SDN
- L3 Fabrics
- Overlays
- Distributed
- Centralized

<table>
<thead>
<tr>
<th>Prog.</th>
<th>2008</th>
<th>2011</th>
<th>2014</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Fabric Evolution</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fabric Gen-1 (vPC)</td>
</tr>
<tr>
<td>Fabric Gen-2 (FabricPath)</td>
</tr>
</tbody>
</table>

*Null*
MSDC Defined: Main Drivers

- **TCO**
  - Common architectures
  - Automation: Operational efficiency

- **Scale**
  - Clos-based architectures with wide ECMP
  - 10G/40G/100G in dense packages
  - Servers at 10G, fabrics at 40G today

- **Agility**
  - Multi-generational fiber plant
  - Automation: Ease of deployment
  - Form factor / space

- **Visibility**
  - Power and thermals
    - Low-power devices
    - Ambient-air cooling
  - Modular designs (aka container)
  - Automation: Stats gathering
  - Multi-stage analysis
  - Automation: Time to market
  - System-level health analysis
  - Flow-level analysis
  - Packet delivery analysis
  - Buffer Performance

- **Common architectures**
- **Automation**
- **Time to market**
- **Stats gathering**
- **Multi-stage analysis**

Interrelated, multidimensional goals
SDN Use Cases
## Introduce SDN use cases

### Cloud provider and SP SDN V2

- Federating different Network Control Points (DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)
- Consistent Network Policy, Security, Threat Mitigation

### SDN origin (Web2.0/SP/CP) SDN V0

- Custom Routing & traffic processing (analytics, Encryption)
  - Online Traffic Engineering
- Infrastructure Management (large Scale)

### SDN V1

- Network Virtualization, Service Chaining
- Network Function Virtualization
  - +90% of the demand today (enterprise, SP)
- "Global Outcome" Smart network

### QUICK WINS

- Fast IT: Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual & Physical)
Different Audiences For Different Drivers

- Federating different Network Control Points (DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)
- Consistent Network Policy, Security, Threat Mitigation
- Custom Routing & traffic processing
  Online Traffic Engineering
- Infrastructure Management (large Scale)
- Network Virtualization, Service Chaining
- Network Function Virtualization (NFV)
- Fast IT: Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual & Physical)

Network OS / Service Developer

Application Developer, System Administrator, Network Operator

Extend, modify, customize the functionality of the network

Leverage the functionality of the network and integrate into new / existing software systems (applications & operations)
Software Defined Network
What means programmability? What is the ‘state of the art’?

Application Frameworks, Management Systems, Controllers, ...

DEVICE CONFIGURATION
- C/Java
- Python
- NETCONF
- REST
- Puppet

FABRIC/NETWORK CONFIGURATION
- OpenFlow
- OpenStack
- ACI Fabric
- WAN
- SERVICES

Management
Orchestration
Network Services
Control
Forwarding
Device

Operating Systems – IOS / NX-OS / IOS-XR

API and Data Models
- YANG
- JSON

Services
- OpenFlow
- Neutron
- OpFlex
- BGP-LS, PCEP, I2RS
- NSH
Third Party & Custom Application Integration

- All third party or custom applications will be packaged as RPMs:
  - Deployed natively in NX-OS underlying Linux
  - Deployed in an isolated within Secure Guestshell Environment

- Packages posted on CCO public repository

- Custom Application Building/Integration:
  - Download Yocto 1.2 SDK toolchain, available openly from Cisco DevNet or [www.yocto.org](http://www.yocto.org)
  - Install SDK on any Linux distribution server in your datacenter - Fedora/Ubuntu/CentOS
  - Build/Made application source/scripts, and package as RPM and deploy using yum install on infrastructure

- Deploy Application using Standard Linux Methods:
  - Start application manually using start script in
    - “/etc/init.d/app start
    - ” or “service app start”
  - To ensure application is started on reboot, create startup script file under
    - “/etc/rc.d/rc3.d/S10app
    - ”

- NX-OS patching infrastructure will also leverage rpm packaging
  - format
Third Party Application Integration

- Third Party Protocol Apps
- Devops Orchestration
- Automation
- Monitoring/Analytics

- Routing/Switching Packages
- Base NX-OS
- Third Party/Custom Applications

- Kernel Route
- Kernel ARP
- Kstack/Netdevs

- Switch Hardware

- OpenLLDP
- Ansible
- Chef
- Ganglia
- Splunk
- Yocto Based Linux Kernel

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
Enabling Technology
How to automate a complex system of complex subsystems
DC Automation by Abstraction

How do we control the infrastructure?
The workflow automated datacenter

Define Workflow

Manage Exceptions

Control Workflow Execution

Analyze SLAs

The nasty part
There are two approaches to Control Systems

**IMPERATIVE CONTROL**

Baggage handlers follow sequences of simple, basic instructions

**DECLARATIVE CONTROL**

Air traffic control tells where to take off from, but not *how* to fly the plane
The Policy Defined Datacenter

Eliminate the nasty terms from the equation

- Define Policies
- Apply Policies
- Analyze SLAs
- Manage Exceptions

You have to TRUST the (auto-)pilot!!

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
The automated DC with policy based infrastructure definition

End-User Service Catalogue

Need Infrastructure

Need Infrastructure

API

Standard Infrastructure-Service-Catalogue

Policy Driven Infrastructure

Physical + Virtual

High-Level IT Prozess-Automation
SIM Card
Identity for a phone

UCS Service Profile
Identity for compute

Service Profile
Network Policy
Storage Policy
Compute Policy

ACI Application Profile
Identity for the network
Group Based Policies in Openstack

Watch out: can you trust the pilot?
Programmierbeispiel
Mode 2 - Operations
Let’s take it a step further

- How to find a VM
- How about more info?
- Find the VM
  - ... and Hypervisor
  - ... and Leaf switch
  - ... and Attached Port
Now find a VM and details

vmname = 'paul-Ubuntu'
cq = cobra.mit.access.ClassQuery('compVm')
cq.subtree = 'full'
cq.propFilter = 'eq(compVm.name, "{}")'.format(vmname)
vm = md.query(cq)[0]

dnq = cobra.mit.access.DnQuery(list(vm.rshv)[0].tDn)
dnq.subtree = 'full'
hv = md.query(dnq)[0]
adj = list(hv.adj)[0]

print 'Virtual Machine Name: {}'.format(vm.name)
print 'Operating System:     {}'.format(vm.cfgdOs)
print 'Hypervisor Name:      {}'.format(hv.name)
print 'Cluster Name:         {}'.format(list(list(hv.dn.rns)[-2].namingVals)[0])
print 'Hypervisor IP:        {}'.format(adj.addr)
print 'Switch Topology Dn:   {}'.format(adj.ifName)
print 'Learned Interface:    {}'.format(adj.ifId)
print 'Learned Neighbor:     {}'.format(adj.nbrName)
And the output

Virtual Machine Name: paul-Ubuntu
Operating System: Ubuntu Linux (64-bit)
Hypervisor Name: 172.31.222.22
Cluster Name: ASDCC
Hypervisor IP: 172.31.222.22
Switch Topology Dn: topology/pod-1/protopaths-101-102/pathep-[vm-vpc10]
Learned Interface: Ethernet1/10
Learned Neighbor: leaf1
Fazit
Ein objektorientiertes Datenmodell ist die Grundlage für eine Netzwerk Programmierung die applikationsbezogene Dienste bereitstellen soll. Das bedingt, dass Netzwerk Elemente wie Switche von Grund auf darauf ausgelegt sind APIs wie REST zur Verfügung zu stellen. Dies APIs können dann von Kontroller Elementen als Management Einheit zusammengefasst werden.

Kontroller können dann für die jeweiligen Anforderungen (Bi-Modal IT: Mode 1 oder Mode 2) GUI oder REST basierte APIs sowie auch CLI zur Verfügung stellen. Um tägliche administrative Tasks zu absolvieren.

Das Objekt Model bietet den vollen Zugriff auf die darunter liegende Hard- und Software Elemente. Diese Objekte werde hierarchisch im Mgmt Information Tree MIT organisiert.

Dieser Ansatz bietet einen erweiterten Rahmen für die Netzwerkkontrolle und Programmierbarkeit die in anderen Systemen nicht existiert.
Ready to code? We've brought the Learning Labs straight to your door!

Over 20 step-by-step tutorials using the DevNet sandbox: some completed in minutes

Develop in the Learning Labs

Sandbox  IoT  Cloud  Networking  Data Center  Collaboration  Security  Services
Fragen ?