Security Level:

Carrier SDN: Security and Resilience Requirements

www.huawei.com

Author/ Email: Artur Hecker / artur.hecker@huawei.com

Version: V1.0(20131105)

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Scope of this talk

- **This talk**
  - Is about use of *cloud technologies* by *carriers (=telcos, mobile operators)*
  - With an explicit focus on SDN
  - SotA, security, resilience
  - Tries to give some outlook

ITG INFORMATIONSTECHNISCHE GESELLSCHAFT IM VDE

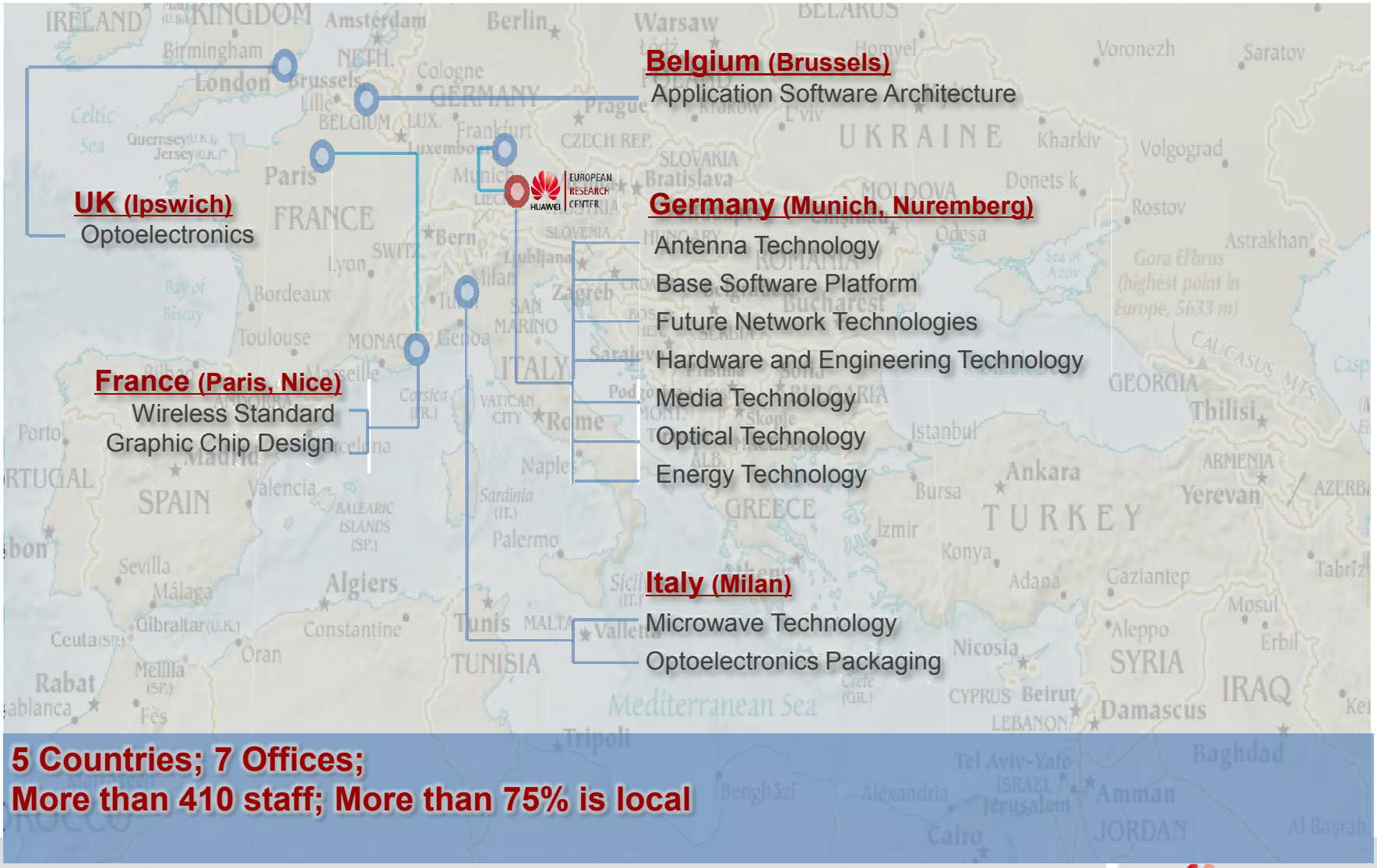
VDE

- **This talk is not about…**
  - Cloud security at large
    - Whether it's wise or not store your files in the US, Myanmar, etc.
      - But do look at PIRS, ORAM, encrypted search and fully homomorphic encryption
    - Legal and national considerations, certification and normalization
      - Safe Harbor, PRISM, etc.

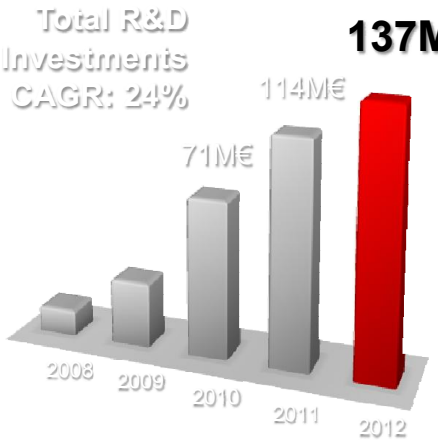
About the author/presenter

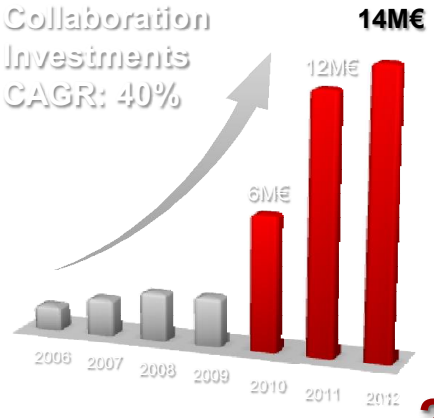
- **Senior Researcher at Huawei's European Research Centre (ERC)**
  - Working in the Future Carrier Networks (FCN) research team
  - In Munich, Germany

The Competence Centers in ERC



Huawei Research in Europe: milestones





Jan 2013 Total People exceeds 750+

Dec 2012 Establishment of Finland and Ireland branches

Jan 2012 Establishment of UK Branch

Sept 2011 Establishment of Nuremberg Branch

Jun 2011 Total People reaches 550

Apr 2010 Total people reaches 350

Apr 2009 Establishment of Belgium Branch

Mar 2009 Establishment of Gothenburg branch

Dec 2008 Major Movement from Bonn to Munich

Jun 2008 Establishment of Milan Branch

Mar 2008 Establishment of Bonn Centre

Dec 2007 Total People in Sweden 50+

2000 Establishment in Stockholm

Carriers and Clouds

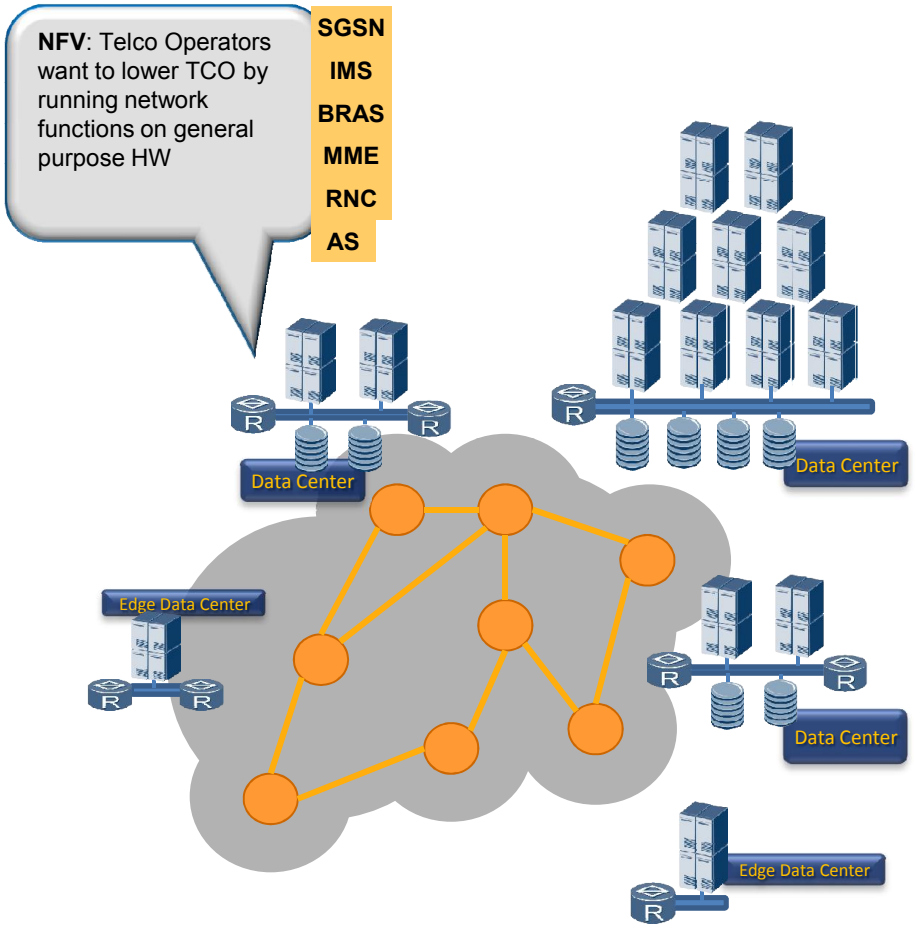
- **Carriers interested in Clouds in general**
  - Would like to use existing ***Cloud Offerings***
    - Example: use cloud storage to outsource legal long-term storage (data retention: connection logs, billing data, DNS data)
    - Example: complement your infrastructure with a leased line, elastically increase the capacity of your service platform, etc.
  - Would like to use ***Cloud Technologies*** themselves
    - Migrate service platforms to DCs
    - Use more flexible network functions on COTS hardware (➔ ETSI NFV)
    - Fully programmable network, carrier network as a Full SDN
      - Carrier infrastructure as interconnected DCs, resource orchestration
  - Would like to propose ***Cloud Services (XaaS)***
    - B2B: dynamically lease parts of your infrastructure
    - B2C: storage, computing, private network extensions, apps (mail, gaming, …)

Carriers and Cloud Technologies

- **Carrier Cloud: Carrier Network as an SDN**
  - *… is a class of cloud that integrates wide area networks (WAN) and other attributes of communications service providers' carrier grade networks to enable the deployment of highly demanding applications in the cloud.* (Wikipedia)

- **Look in the future – SDN API at UNI?**
  - Resolve the application blindness!
    - Concrete problems: a popular mobile app can clog the network

Carrier and Cloud Technologies



Carrier Grade SDN: requirements (*)

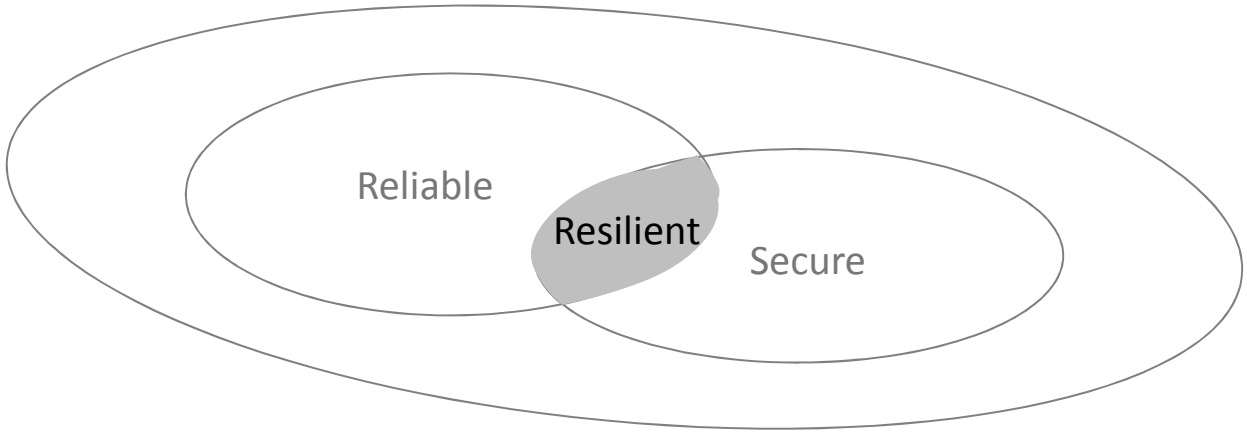
- **Deployment paths, integration in the existing infrastructures**
  - Legacy device support, hybrid operation modes
- **Scalability**
  - Large-scale networks: thousands of devices, multiple controllers, …
- **Service Level Agreements**
  - Quality of Service support, high availability
- **Manageability (=OAMP)**
- **Reliability, security, resilience**
  - Operations under faults, failures, misconfigurations
  - Security of data and security of the overall system
  - Resilience against maliciously planned attacks

(*) Cmp NTT, EU FP7 SPARC

CG-SDN:
Reliability-Resilience-Security

Reliable, resilient, secure

- **Reliable**
  - How good does it work under known/random faults?
- **Security**
  - How can one maliciously abuse an SDN?
- **Resilience**
  - Ability to operate under load, failures and attacks



Reliability and Resilience: different methodologies

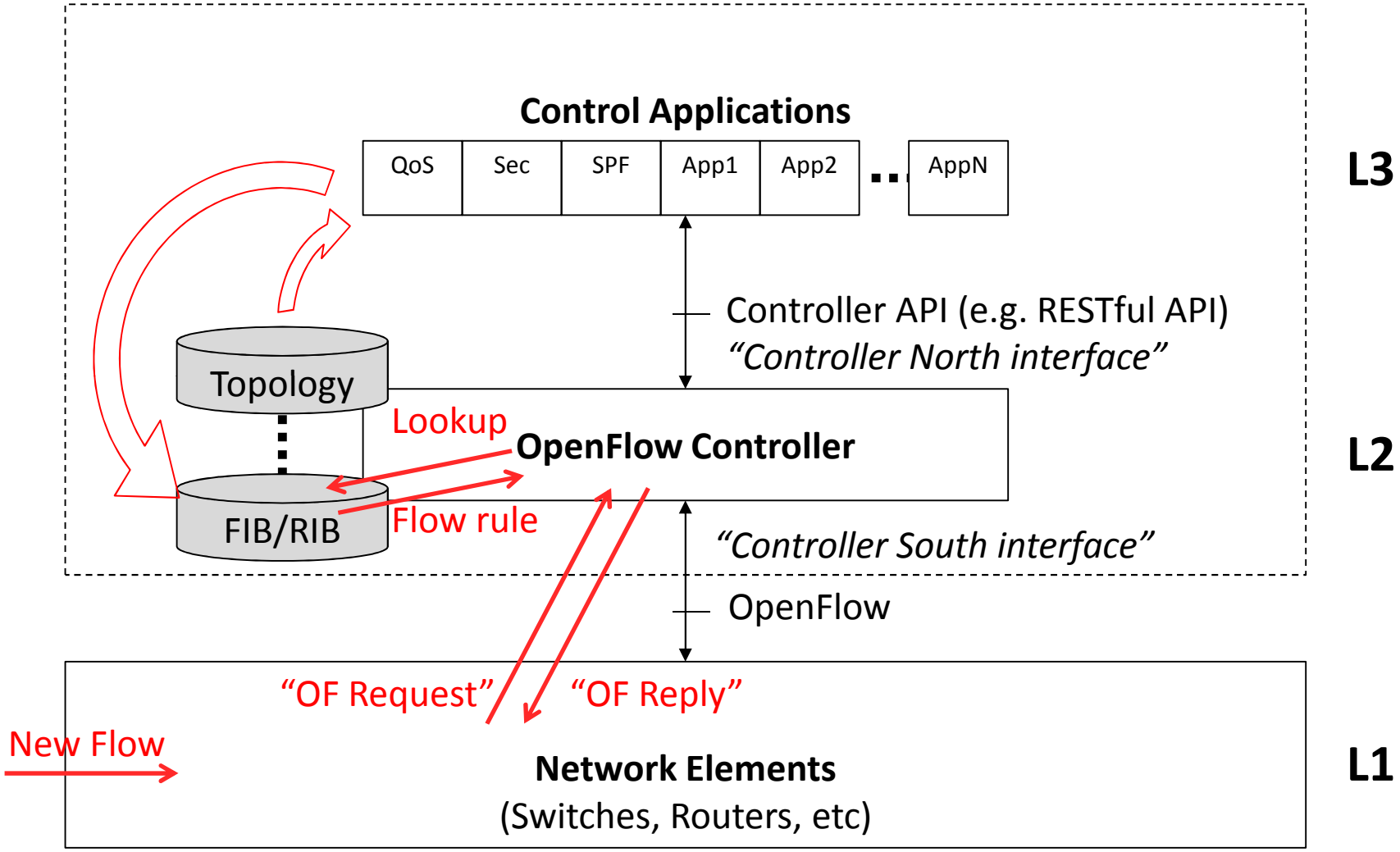
- **Reliability = service thinking**
  - "Make it work" attitude
  - Methodology
    - Requirement engineering
    - Formal methods (model driven design, model checking)
      - Allows for proofs under specific failure models
  - Implementation: per entity and structural redundancy
- **Resilience requires security thinking**
  - Experience-based: from laissez-faire to paranoia
  - Depends on the deployed base and context/environment
  - Methodology
    - Asset, value => high level risks
    - Context, usage => usage, fault and attack models
    - Concrete implementation => vulnerabilities
  - Implementation: protection/detection/reaction, Denning cycle (PDCA)

SDN today

OPEN NETWORKING FOUNDATION

- **Mainly driven through ONF's OpenFlow**
- **So far used as a new enabler technology in *computer networks***
  - LAN/MAN: academic environments, enterprise networks
  - Increasingly also in Data Centers (DC) – part of Cloud Technology
- **A very active area of research and development**
  - Compare recent developments in the research community
  - Several open source controller projects
  - Several vendors active
  - Expected interesting outcomes
    - Network programming: network languages & compilers, static analysis
- **How to bring that emerging know-how to Carriers?**

Current SDN Model with OpenFlow



Novelty and contribution of SDN/OF (*)

- **Usually identified with control and forwarding plane separation**
  - Split architecture, etc.
- **Yet, this is hardly new!**
  - Telco networks have been doing this for decades
- **SDN = fuse control and management and centralize both**
  - SDN makes both control and management data centrally available
  - Achieves consistency
    - Which in turn allows for full network programmability
    - Event-based, close to real time cross-layer control
- **But what about the CAP theorem?**
  - Which two of {Consistency, Availability, Partition Tolerance} do we need?
  - Carriers: {Availability, Partition Tolerance} – this contradicts SDN!

➔ **Need for research and prudent Engineering**

(*) Cmp Y. Stein, IRTF SDN list

SDN: value and high level risks

- **SDN main promise: full network programmability**
  - Network OS, network compilers, network-wide schedulers
    - E.g. promising: *Frenetic* language project
      - Supports implementation-agnostic specifications of network behaviour
      - Supports verifications and proofs (loop-freeness, etc)
      - Supports consistent updates
- **Programmability+complexity => error-proneness, vulnerability**
  - Openness of APIs
  - Difficult verification of program correctness (offline)
    - *Cyclomatic complexity* of apps as a measure of complexity
  - Very difficult enforcement of benign execution (online)
    - Definition of “benign” is not absolute
    - +Complexity of the physical graphs (network topology)
- **Motivation for attackers**
  - Interesting target: obtain full network control
  - Example: “rootnet” or *black hole networking*
    - Get your own, isolated, difficult to detect network

CG-SDN: deployment and usage

- **Can one deploy an SDN today?**
  - Switches are available on the market
    - Hybrid, native
    - Quality parameters:
      - Size of the flow table (size of TCAM = expensive memory)
      - Flow setup throughput (how many flow rules can be setup per second)
  - Controllers are available both on the market and open source
    - NOX, Beacon, FloodLight, some others
    - Quality parameters, today:
      - Flow setup latency (how much does it take for lookup and response)
      - Flow throughput (how many requests per second can be served)
- **Which are possible deployment models?**
  - What are central questions for a carrier in practice?

Current OF deployments (1/4) *

- **Native OpenFlow**
  - Dumb switches, all control plane messages handled by the SDN controller
- **Problems**
  - Deployment: Switches require IP connectivity to the controller:
    - Out of band (similar to Juniper's Qfabric, too expensive)
    - In band: to isolate from errors, this would need a VLAN (e.g. VLAN#1). Besides, switches would need to run STP (at least within VLAN#1)
  - Load on the controller, scalability
  - Resilience - error recovery
    - Fast Control Loops are hard to implement with a single central controller
    - E.g. cmp BFD, RFC5880, see Resilience in FP7 SPARC deliverables
- **Reality check (e.g. NEC)**
  - Max. ~50 switches per CTRL
  - 200msec rerouting around failed links (depends on network size)

(*) *Compare I. Pepelnjak's essay, http://blog.ipspace.net/2011/11/openflow-deployment-models.html*

Current OF deployments (2/4)

- **Native OpenFlow with Extensions**
  - Controller gives more general statements to switches
  - Switches perform some control-plane functions stand-alone
    - E.g. LLDP, LACP, link aggregation, balancing across multiple links
- **Problems**
  - Compliance: hardware capabilities and extensions to OpenFlow
  - Complexity: requires capability discovery extensions, etc.
- **Reality check**
  - Nicira used extended OF 1.1 with generic pattern matching, IPv6, load balancing
  - OpenFlow 1.1 supports multipathing, but it is rarely implemented in HW

Current OF deployments (3/4)

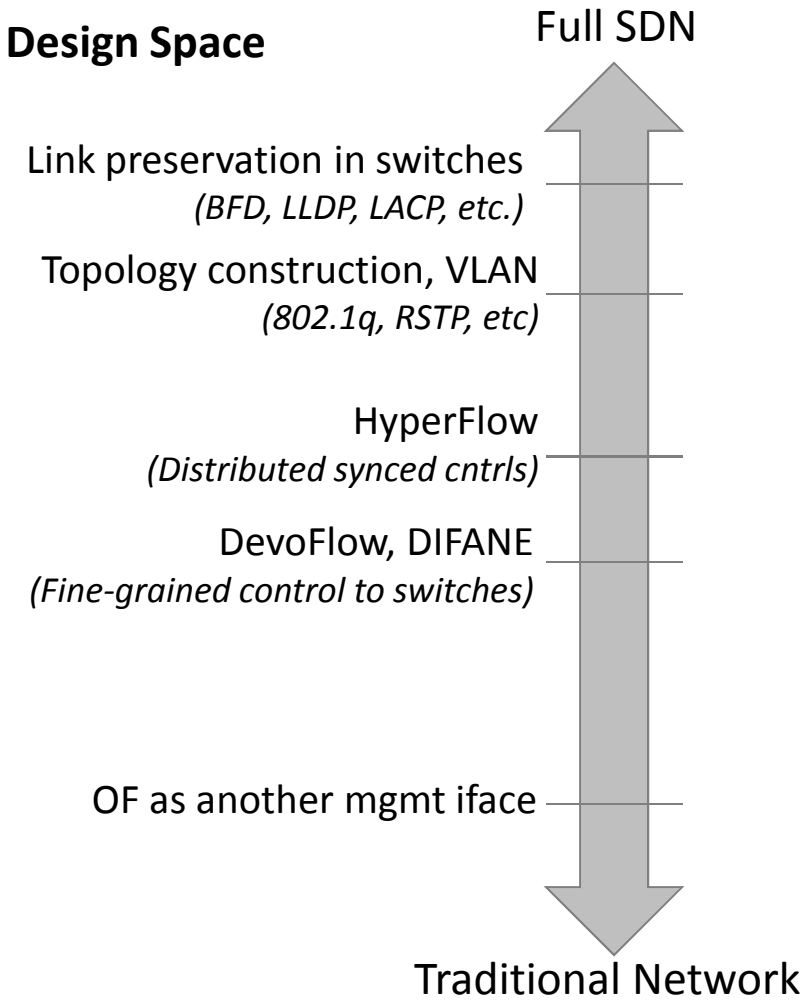
- **OpenFlow-enabled classic networks**
  - Switches have traditional control planes and run tedious periodic tasks
    - LACP, LLDP, BFD; only final state is reported to OF CTRL
  - OF Controller only manages certain ports, VLANs or subnets, etc.
  - Controller-Switch traffic uses a non-OF-controlled part
- **Problems**
  - Deployment: Not a holistic solution. How to manage the rest?
  - Programmability: some things are done by switches
- **Reality-check**
  - Often used in academic and testing environments
    - OF is running in parallel to a production network
  - Not applicable to a CG-SDN full deployment

Current OF deployment (4/4)

- **Integrated OpenFlow**
  - OpenFlow as a complementary, additional interface to the network
    - Controller-submitted entries go into usual provisions, e.g. in RIBs, LIBs, etc.
  - OpenFlow is “just another” interface, parallel to e.g. BGP-TE, ALTO and PCE
- **Problems**
  - Unclean Architecture
  - Integration: Hard to achieve inter-working at higher abstraction layers
  - Security: more open interfaces, need to control/enforce policy over all that
- **Reality Check**
  - Provides a straightforward deployment path
    - No need to develop control apps to re-do what’s already working

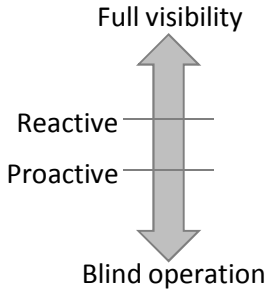
CG-SDN: Controller vs controlled

- **Reality**
  - Extremes do not work
- **Beyond architectural debates**
  - A tradeoff between a FullSDN (ideal) and network performance
- **What about transition path?**
  - Native vs. integrated
  - What impact of hybrid switches?



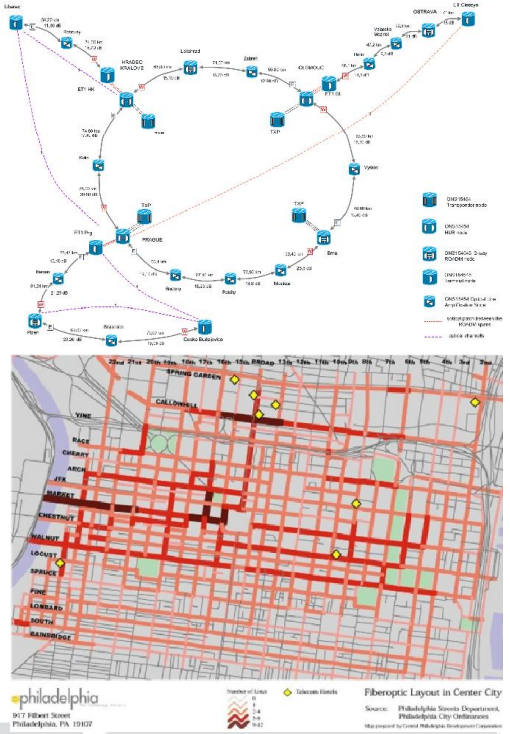
CG-SDN: improve control plane perf

1. **Optimize controller placement to reduce propagation delay**
   - Not treated here. But see HotSDN 2012, “Controller Placement Problem”.
2. **Add controllers in order to distribute flow processing**
   - See HyperFlow or P2P controllers, see Seattle
   - Elastic controllers, controllers in the cloud, etc.
     - See HotSDN 2013, “Elastic controllers”.
3. **Change operation modus from strictly reactive to strictly proactive in order to reduce number of/avoid requests**
   - Cmp. VMWare/Nicira (NVP)
   - Leads to a (partial) loss of reactivity / consistency
4. **Distinguish edge and core switches to reduce flow state**
   - See HotSDN 2012, “Fabric: a Retrospective on Evolving SDN”



CG-SDN: Edge – Core Separation

- **Telco: huge differences between the edge and the core**
  - Core: few connections, few large pipes, best effort, critical
  - Edge: capillar network, plethora of narrow pipes, access/admission
- **Problem**
  - Generalized per-flow state does not scale
  - And is hardly necessary
    - Only *few alternative paths* available in the core
- **Classical solution here**
  - Forwarding abstraction classes
    - == forwarding abstraction, MPLS fwd classes



Conclusions: OF for CG-SDN

- **Some of current deployment models unsuitable for Carriers**
  - Model 1: unrealistic, Model 2: non-standard extensions; Model 3: not an infrastructure control, too partial
- **Some intelligence will remain in switches**
  - In *any model*; but especially in Model 4.
  - Required by installation needs, migration and **resilience/recovery**
    - What is the right distribution for which carrier?
  - Results in
    - Complexity: *capability discovery* needed if support in switches/controllers differs
    - Limited Programmability: whatever goes to switches, goes against the SDN idea
- **Scalability: several controllers will need to exist**
  - For performance reasons, cmp. NTT requirements, HyperFlow, Onix
  - Different OF bases will handle access, backhaul, core, wireless/wired, etc.

State of the Art: Reliability

- **Protections for specific fault models, e.g.**
  - Controller failure
  - Link or port or switch failure
- **Reliability mechanisms**
  - Typically based on protection and restoration
    - Protection: pre-calculate alternative paths + install redundant controllers
      - Failover controllers supported by the OpenFlow standard since early versions
      - Degraded mode supported for switches that lose controller connections
    - Restore the communication on error
      - Detect and report error (e.g. local BFD module in switches)
      - Controller loads the new path into the switch
  - Policy and state verification
    - Formal verifications: policy compilers detect and resolve conflicts
      - Formal languages (e.g. Frenetic)

State of the Art: Switch vulnerabilities

- **What happens if an attacker takes over one of the switches?**
- **How?**
  - Frayed management interfaces
    - Complexity, complex verifications, possible race conditions
  - Difficulty of personalization, initial configuration, key management
  - Hardware theft
- **Impact**
  - Rogue switches could alter OF rules: mismatch between controller view and real world
  - Rogue switches could produce OF-relevant events

State of the Art: Security of OF (1/2)

- **Normally, switch to controller traffic is protected by TLS**
  - Authenticated channels from switches to controllers
  - Authenticated channels for data replication between controllers
- **OF TLS Support**
  - *Optional* since official versions of OpenFlow
  - Only few models on the market that fully support it
    - Switches: only NEC IP8800 according to HotSDN 2013 paper
    - Controllers: Open vSwitch
  - Some controllers support TLS connections but do not verify anything (call it opportunistic encryption if you want)
  - Difficult to activate and configure correctly

State of the Art: Security of OF (2/2)

- **Reactive controllers are more exposed than proactive**
- **Some messages are more "dangerous" than the others**
  - **Packet In** messages
  - **Flow mod** messages
- **OF 1.3 spec suggests switch->ctrl packet policing**
  - Possible policing: filtering, rate limiting, priority handling, redirects, duplications, etc.
    - But how?
- **What about all the usual network security things?**
  - ARP filters, broadcast and multicast limits, DHCP snooping prevention, STP filtering on ports – ***who will implement them***?

Example: OF network fingerprinting

- **Fingerprinting - SDN scanner**
  - Exploits the difference in timings of new flow and existing flow
  - Samples the network by sending a lot of different flows consisting of two packets
    - In SDN: the first packet would represent a new flow, the 2nd an existing flow. The response delay would be different.
    - Without SDN: there should be no substantial difference but noise.
  - Makes simple statistical testing of the sample sets.

Security State of the Art, briefly

- **How can one attack SDN – identified attack vectors**
  - *Until now* not different from any management platform (cmp. SNMP)
  - A0: Attack the OF traffic
    - Vectors: TLS config and implementation weaknesses, TLS environment abuse (trust)
    - Impact: credential extraction, snooping and injections, useful for multi-stage intrusions
  - A1: Get a switch and send rogue messages to controllers
    - Vectors: ID usurpation, existing switch vulnerabilities, weak authentication
    - Potential impact: full to partial network breakdown
  - A2: Attack the controller and own the network
    - Vectors: DDoS (produce rogue flows, rogue OF reqs), ctrl vulnerabilities, weak authentication
    - Potential impact: from DoS to full network control
  - A3: Attack the management platform
    - Vectors: weak admin auth, access to controller, controller vulnerability
    - Impact: full to partial network control

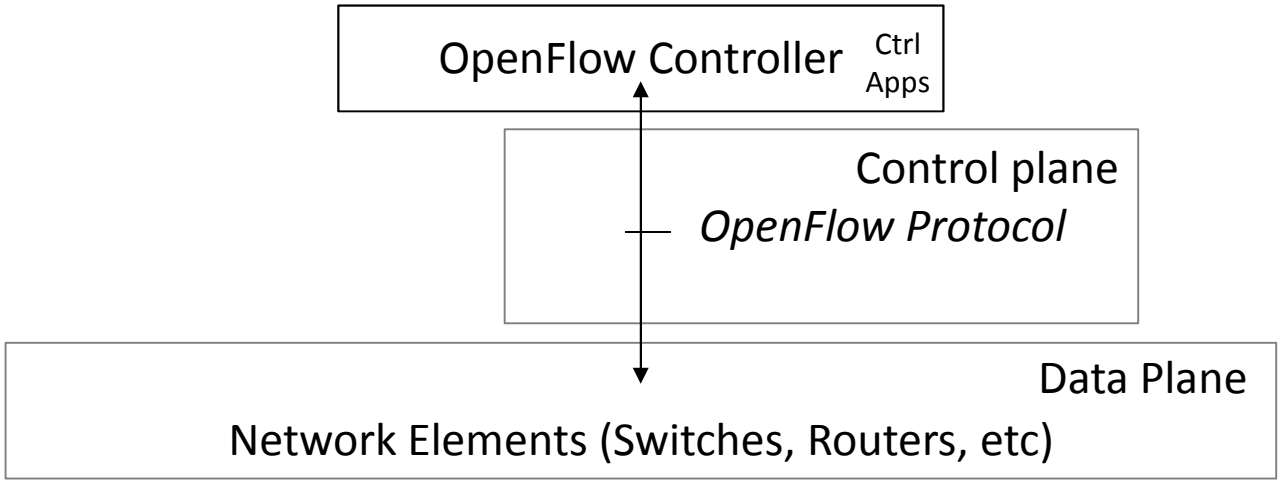
OF Security/Resilience: Conclusions

- **The research has only started in the community**
  - Hardly any interesting results so far
- **Typical (often implicit) assumptions**
  - One fault at a time
  - Limited ecosystem
  - Control path always works
  - Controller state is valid/converged
  - Recent work claims support for Byzantine fault models
    - See new results from HotSDN 2013.

- **But: Do these correspond to the carrier usage of OpenFlow ?**
  - What are carrier reality, the ecosystem/environment, the future?

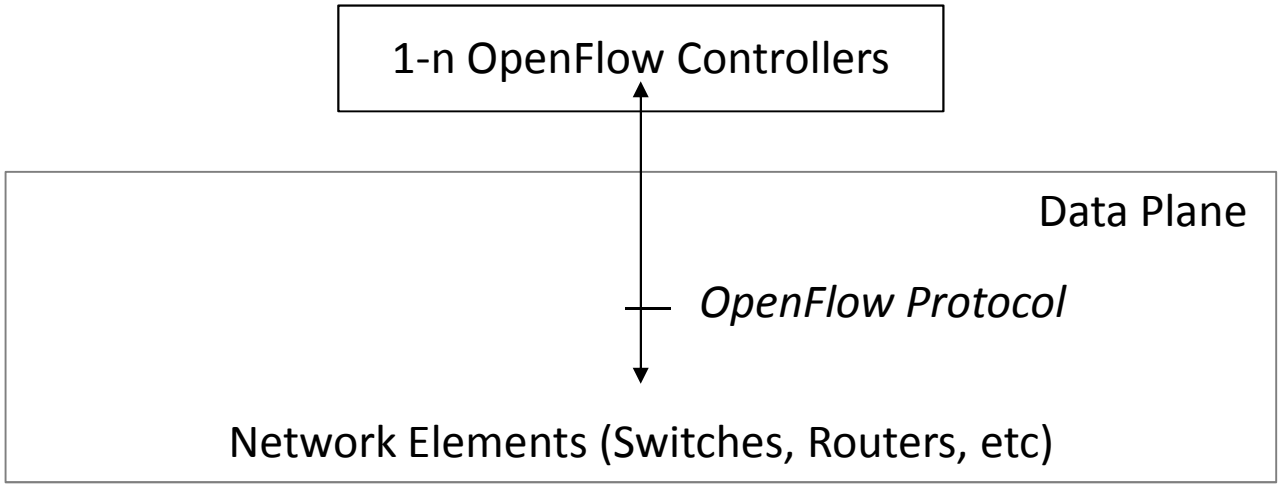
Imaginary SDN Environment

- **Physically separate control plane**
- **Unique, closed, non-extensible controller**



Current SDN Environment

- **Control path leads, in big parts, over the controlled data plane**
- **Several independent, closed controllers**

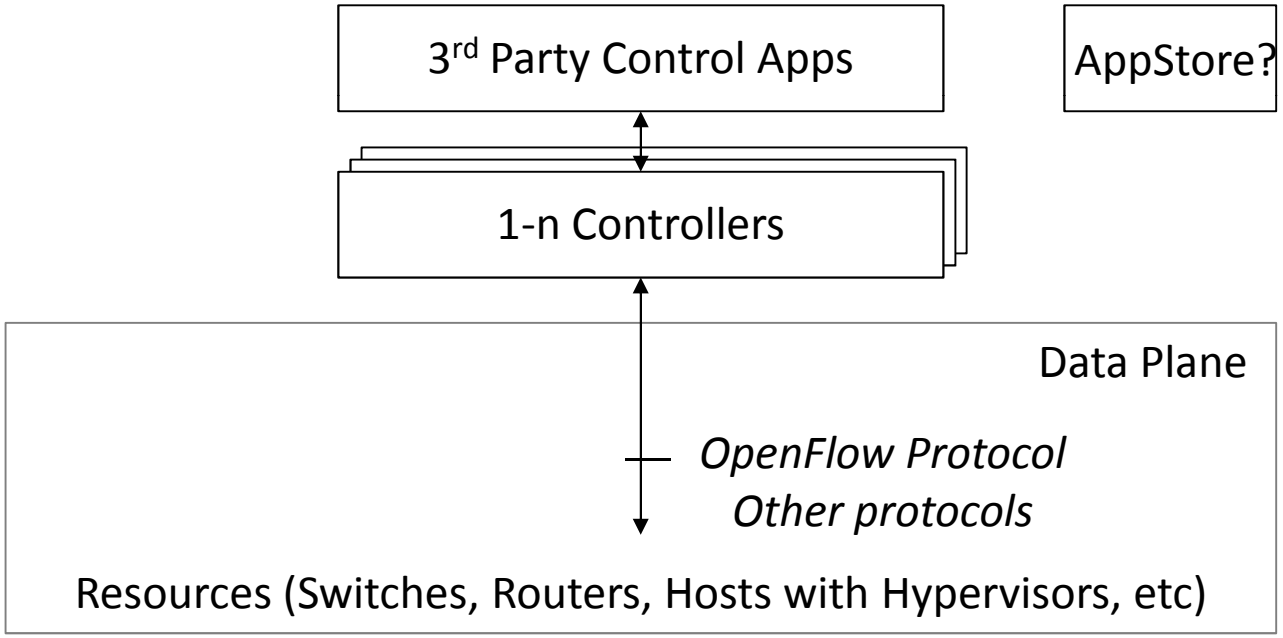


Current situation: new problems

- **How to ensure synchronized state of controllers?**
  - "Split brain problem"
    - See Onix, HyperFlow for some proposals
    - Are these secure? Are they resilient?
  - Protocols to achieve that, traffic to be preserved, failures to be avoided
- **How to virtually isolate control plane from data plane?**
  - How to reach elements behind a failed element?
  - How to make sure that no command sequence…
    - … Leads to a self-lockout?
    - … Leads to unexpected behaviour? (race conditions betweens control apps)

CG-SDN ecosystem

- **Several multi-protocol controllers rule infrastructure elements**
- **Controlled and controlling entities are mixed in the data plane**
- **Extensible controllers**
  - Open API to control applications



SDN: the control applications

- **Can these be generally trusted?**
- **Developed by third parties**
  - Not the controller vendor
  - Not the local admin
- **Approaches:**
  - FlowVisor: prevent cross-slice attacks. How to prevent intra-slice attacks?
  - FortNOX, FRESCO: detect rule conflicts violating security policies
- **Two models**
  - Benign but buggy
  - Malicious
- **Isolation of apps**
  - Controller is more important and should detain the rights
  - Control path should be isolated from apps as well
  - Control external app communications

CG-SDN ecosystem – new problems

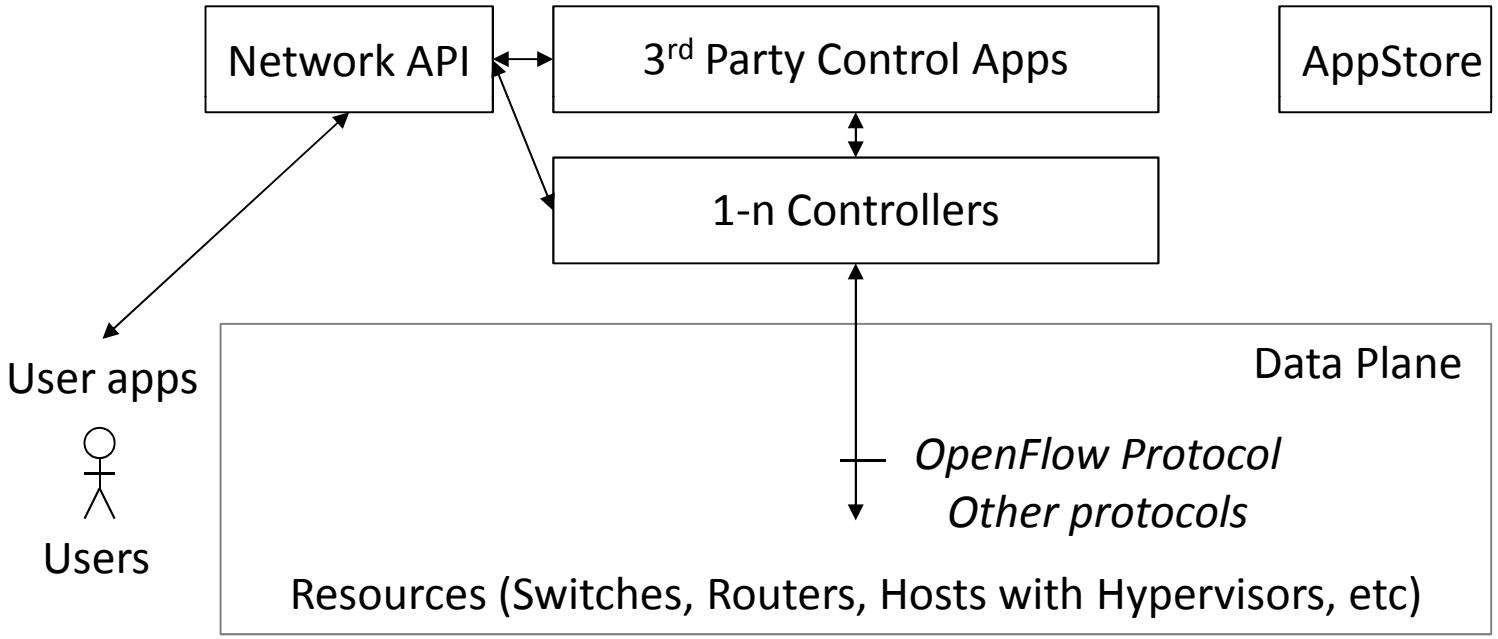
- **How to ensure that an app behaves correctly?**
- **How to vet apps in the AppStore?**
- **Flexible control plane:**
  - How to dynamically deploy additional controllers?
  - Where to provision control paths?
  - How to intelligently orchestrate entities? (where when which one)
    - Joint path/node optimization, see FARO

Sec SDN: the control applications

- **Can these be generally trusted?**
- **Developed by third parties**
  - Not the controller vendor
  - Not the local admin
- **Two potential models**
  - Benign but buggy
  - Malicious
- **Approaches:**
  - FlowVisor: prevent cross-slice attacks. How to prevent intra-slice attacks?
  - FortNOX, FRESCO: detect rule conflicts violating security policies
  - Secure Controller Platform - Isolation of apps
    - Controller is more important and should detain the rights
    - Control path should be isolated from apps as well
    - Control external app communications

Future possible CG-SDN

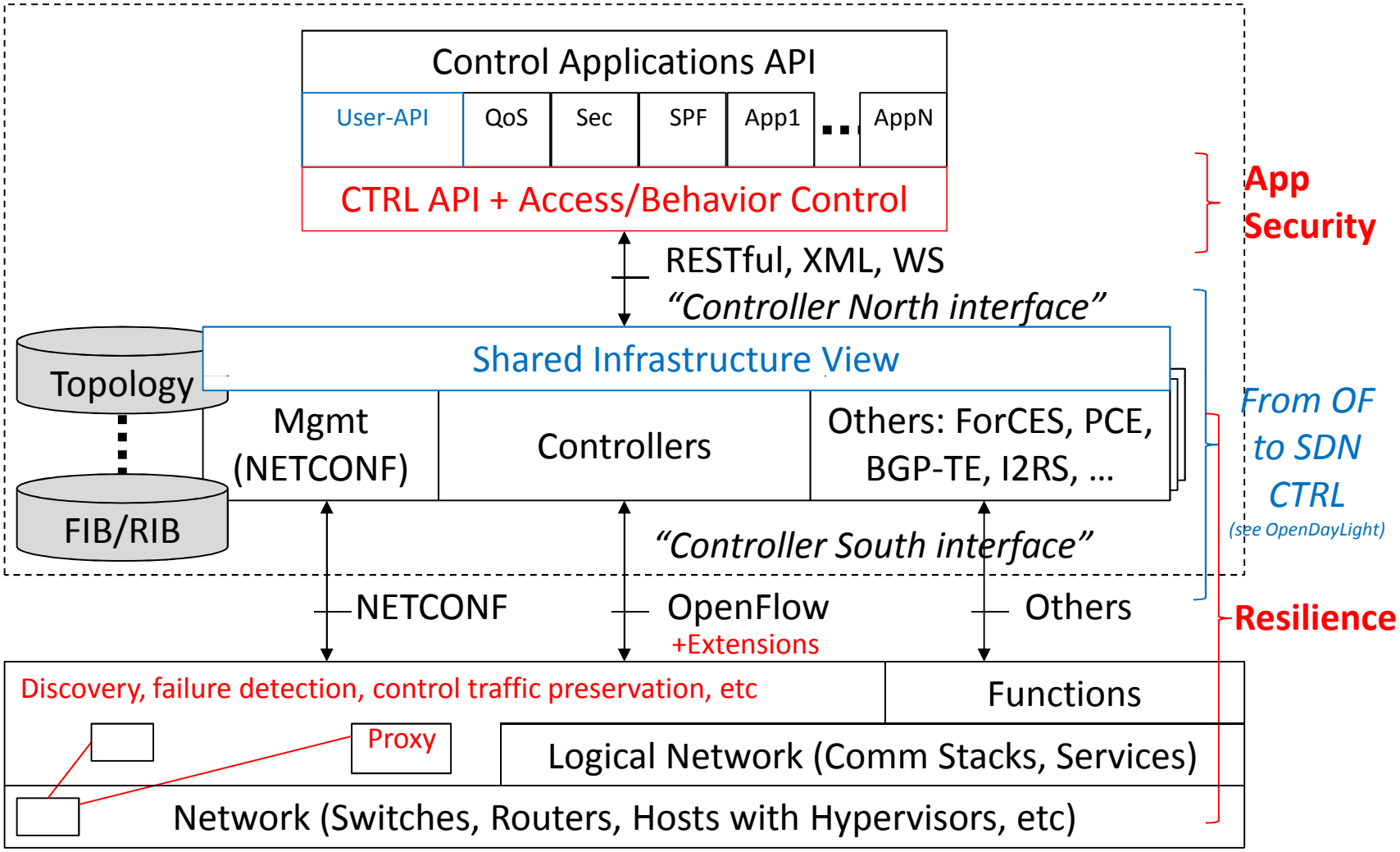
- **Network API enables development of network aware apps**
  - Towards explicit exchanges with a programmable network
    - From the pipe to semantic networking, support for innovative services
      - Criticality, locality of treatment, extreme QoS constraints



Network API

- How to identify and authorize user apps?
- How to verify their origin, correctness, etc.
- What are the necessary, recommended, advisable limitations on the Network API?

Towards full CG-SDN



Conclusions

- **Deployment of OpenFlow very limited until now**
  - Only academic attacks so far
  - Bad guys have had no incentives to look into it so far
- **A real CG-SDN will be considerably more complex**
  - Several controllers sharing states
  - Complex switches with autonomous capabilities
  - Open control application interface, mix of different applications
    - Resulting in a mix of control apps from unclear sources
- **Research has only started on these matters**
  - The findings already are worrisome
  - Yet, most attacks are not even characteristic to SDN!
- **SDN: General programmability = major vulnerability**
  - **The very feature bears the main risk**
  - *What will happen when thousands of programmable entities will be waiting in a large-scale environment for close-to-realtime programmatic control by other virtual entities?*

Thank you

www.huawei.com

References (1/2)

- **Google WAN SDN**
  - Google, “Inter-Datacenter WAN with centralized TE using SDN and OpenFlow”, white paper.
  - U. Hoelzle, “OpenFlow @ Google”, Open Networking Summit, April 2012.
  - A. Vahdat, “A Software Defined WAN Architecture”, Open Networking Summit, April 2012.
- **Practical Take on OpenFlow**
  - I. Pepelnjak’s blog, http://blog.ipspace.net/
  - Open Networking Foundation, https://www.opennetworking.org
- **OpenFlow Controllers**
  - NOX, Open Source SDN Controller, http://www.noxrepo.org/
  - FloodLight, Open SDN Controller, http://floodlight.openflowhub.org/
  - Beacon, OpenFlow Controller in Java, https://openflow.stanford.edu/display/Beacon/Home
  - OpenDayLight, http://www.opendaylight.org/
- **Carrier Grade SDN**
  - Research project FP7 SPARC, www.fp7-sparc.eu
  - D. Staessens et al., “Software Defined Networking: Meeting Carrier Grade Requirements”, LANMAN 2011.
  - Y. Ito, “Expectations for OpenFlow/SDN as Carrier’s Network”, Open Networking Summit, April 2012.
- ***Programmability, verification***
  - *Frenetic Language Project, http://www.frenetic-lang.org*
  - *Nate Foster, Michael J. Freedman, Arjun Guha, Rob Harrison, Naga Praveen Katta, Christopher Monsanto, Joshua Reich, Mark Reitblatt, Jennifer Rexford, Cole Schlesinger, Alec Story, and David Walker. Languages for software-defined networks. IEEE Communications Magazine, vol 51, issue 2, 2013.*
  - *Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, David Walker, “Abstractions for Network Update”, in proc. ACM SIGCOMM 2012, Helsinki, Finland.*
  - *Arjun Guha, Mark Reitblatt, and Nate Foster. Machine-Verified Network Controllers. In ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Seattle, WA, June 2013.*

References (2/2)

- ***Scalability of SDN***
  - *T. Koponen et al., "Onix: A Distributed Control Platform for Large-Scale Production Networks", USENIX OSDI 2010.*
  - *A. Tootoonchian, Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow", INM/WREN'10 in conjunction with USENIX NSDI, April 2010.*
  - *Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, "DevoFlow: Scaling Flow Management for High-Performance Networks", in proc. ACM SIGCOMM 2011, Toronto, Canada.*
  - *M. Yu, J. Rexford, M. J. Freedman, J. Wang, „Scalable Flow-Based Networking with DIFANE", ACM SIGCOMM 2010.*
  - *A. Voellmy, J. Wang, "Scalable Software Defined Network Controllers", ACM SIGCOMM 2012 Poster.*
  - *S. H. Yeganeh, A. Tootoonchian, Y. Ganjali, "On Scalability of Software-Defined Networking", IEEE Communications Magazine, vol 51, issue 2, 2013.*
  - *Soheil Hassas Yeganeh, Yashar Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications", in proc. ACM HotSDN 2012, Helsinki, Finland.*
- ***Reliability/Security/Resilience***
  - *L. MacVittie, Cyclomatic Complexity of OpenFlow-based SDN, DevCentral, f5.com*
  - *A. Khurshid, W. Zhou, M. Caesar, P. Brighten Godfrey, "VeriFlow: Verifying Network-wide Invariants in Real Time", ACM SIGCOMM 2010.*
  - *D. Kreutz, F. M.V. Ramos, P. Verissimo, „Towards Secure and Dependable Software-Defined Networks", HotSDN 2013.*
  - *K. Benton, L. J. Camp, C. Small, "OpenFlow Vulnerability Assessment", HotSDN 2013.*
  - *S. Shin, G. Gu, "Attacking Software Defined Networks: A first Feasibility Study", HotSDN 2013.*
  - *X. Wen, Y. Chen, C. Hu, C. Shi, Y. Wang, "Towards a Secure Controller Platform for OpenFlow Applications", HotSDN 2013.*