IoT Design

Product Design

- Conceptual Design
- Industrial Design
- Engineering Design
- User Design
- Business Design
- Service Design
- Network Design
- Eco System Design
  - Create a larger pie
  - Blue Ocean

Product Questions

- What does it do
  - Value Proposition
- Why do I need it instead of other solution?
  - What is it that your product is doing better
- How does it work
  - Conceptual Design
- How do I use it?
  - Interaction Model

People Based Design

- User-centerd Design (Usability)
- Friendly Interaction
- Visually Attractive
- Without People, IoT is IoN (Internet of Nothing)
- Emotional Design
  - Human Behavior
  - Responsiveness
- Context Aware
  - Ambient Intelligence
- Connected Experience
- Ubiquitous Experience

UI – User Interface

- Graphical UI
  - Static (Desktop)
  - Mobile
  - Spatial (Projected)
  - VR
  - AR
- Natural UI
  - Voice
  - Brain
  - Gesture
  - Eye
- Tangible UI
  - Direct Manipulation
  - Embodiment
  - Metaphoric

Memory Budget

- Kernel/System
- Utility
- Driver
- Network
- App
- Swap/Temp
- Data/Database
- User
- Max/Min Work Load

mBed OS Code Size

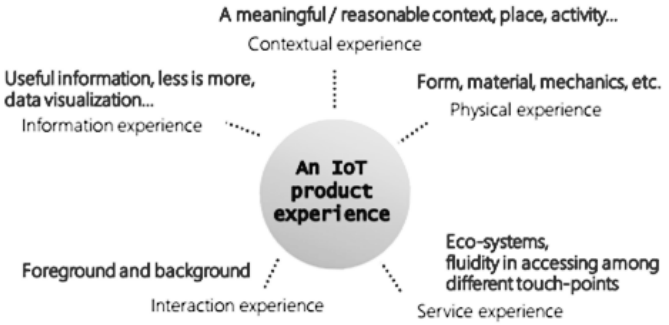
Name	Code Size	Target CPU
pOSEK	2K	Microcontrollers
pSOSystem		PII->ARM Thumb
VxWorks	286K	Pentium -> Strong ARM
QNX Nutrino	>100K	Pentium II -> NEC
QNX RealTime	100K	Pentium II -> SH4
OS-9		Pentium -> SH4
Chorus OS	10K	Pentium -> Strong ARM
ARIEL	19K	SH2, ARM Thumb
Creem	560 bytes	ATMEL 8051

Other Engineering Design

- Thermal Budget
  - Hazard Control
  - Air Flow
  - Heat Dissipation
  - Thermal Noise
- Form Factor
  - Where to install the device
  - Industrial Design
- Housing
  - Weather Proof
  - Fire Resistance
  - Keep Out Area

User Experiences

- Interusability
- Composition
  - **Subset of functions on device, full set in app.**
  - **Mirrored functions across device & app.**
- Consistency
- Continuity →

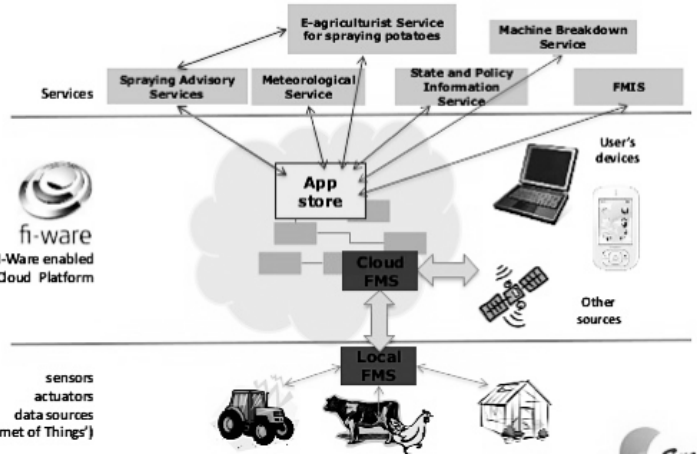


Abstract

A wireless sensor network-based automatic monitoring system is designed for monitoring the life conditions of greenhouse vegetatables. The complete system architecture includes a group of sensor nodes, a base station, and an internet data center. For the design of wireless sensor node, the JN5139 micro-processor is adopted as the core component and the Zigbee protocol is used for wireless communication between nodes. With an ARM7 microprocessor and embedded ZKOS operating system, a proprietary gateway node is developed to achieve data influx, screen display, system configuration and GPRS based remote data forwarding. Through a Client/Server mode the management software for remote data center achieves real-time data distribution and time-series analysis. Besides, a GSM-short-message-based interface is developed for sending real-time environmental measurements, and for alarming when a measurement is beyond some pre-defined threshold. The whole system has been tested for over one year and satisfactory results have been observed, which indicate that this system is very useful for greenhouse environment monitoring.

Keywords:
wireless sensor network; embedded operating system; environment monitoring; data center; base station

Smart Farming in the Cloud



2. **Concept Design**
   3. Functions
   4. System Model
   5. Network Architecture
1. Product Design
   1. Things /Components (ID/Form Factor)
   2. App / UI
   3. Web Dashboard / Portal
4. Business Design (Optional)
   1. Market
   2. Billing
   3. Maintenance
   4. Scaling Plan

IoT Protocols

- http
- CoAP
- MQTT
- XMPP
- AMQP

What's the Purpose of IoT Protocols

- Transport
- Session Management
- Network Routing
- Messaging

A Few Key Protocols

- HTTP
  - IETF standard (RFC 2616 is HTTP/1.1)
- CoAP
  - IETF draft 18 (December 2013)
- MQTT
  - soon (August 2014 ?) OASIS standard
- AMQP
  - OASIS and ISO 19464 standard (1.0)
- XMPP

CoAP vs HTTP

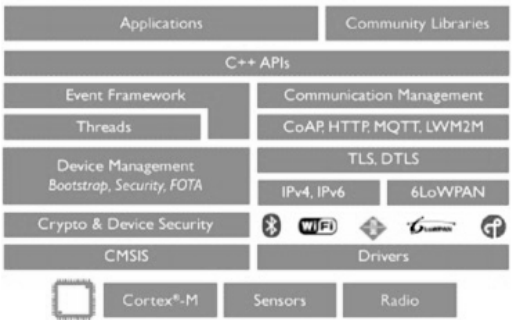
HTTP PERFORMANCE WITH COAP			
Internet Protocol suite (TCP/IP)		IP Smart Objects Protocol suite	
Application layer	HTTP/FTP/SMTP/etc.	Application layer	CoAP
Transport layer	TCP/UDP	Transport layer	UDP
Network layer	IPv4/IPv6	Network layer	6LoWPAN
Link layer	802.3, Ethernet/ 802.11, wireless LAN	Link layer	IEEE 802.15.4e

Key IoT Protocols Standardization

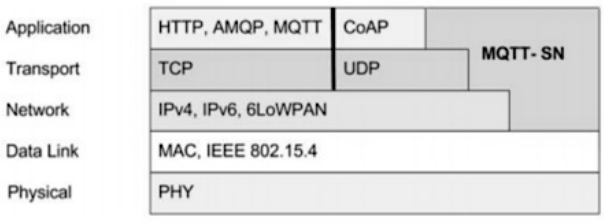
- **IETF**
  - **6LoWPAN** Working Group (IPv6 anywhere)
  - **ROLL** (Routing Over Low-power Lossy Networks) WG
  - CoRE WG (REST for IoT, **CoAP**, Resource Directory etc.)
  - TLS WG (**DTLS**)
- **OMA**
  - Lightweight M2M Enabler Standard (CoAP/DTLS based)
  - Device Management 2.0 Enabler Standard (HTTP/TLS based)
- **ETSI / OneM2M**
  - Ongoing work on M2M system standardization (CoAP, HTTP binding)
- **W3C**
  - Efficient XML Interchange (EXI) standardization
- **ZigBee IP**
  - An open-standard 6LoWPAN stack for e.g. Smart Energy 2.0

Protocol	CoAP	XMPP	RESTful HTTP	MQTT
Transport	UDP	TCP	TCP	TCP
Messaging	Request/Response	Publish/Subscribe Request/Response	Request/Response	Publish/Subscribe Request/Response
2G, 3G, 4G Suitability (1000s nodes)	Excellent	Excellent	Excellent	Excellent
LLN Suitability (1000s nodes)	Excellent	Fair	Fair	Fair
Compute Resources	10Ks RAM/Flash	10Ks RAM/Flash	10Ks RAM/Flash	10Ks RAM/Flash
Success Stories	Utility Field Area Networks	Remote management of consumer white goods	Smart Energy Profile 2 (premise energy management, home services)	Extending enterprise messaging into IoT applications

ARM OS Architecture Protocol Stack



Stack Comparison



HTTP – HyperText Transfer Protocol

- together with HTML forms the base of WWW
- is standardized by IETF (rfc 2616)
- is a request-response protocol
- it is stateless (does not maintain a state of a session) and asynchronous (an html document is loaded asynchronous by the browser, as soon as parts of it are available)
- latest version is HTTP/1.1
- runs on top of TCP on the standardized port 80

HTTP Request

- has the form:

  Request-Method SP Request-URL SP HTTP-Version <CR><LF>
  (generic-header | request-header | entity-header <CR><LF>)
  <CR><LF>
  [message body]

- Request-Method is:
  - **GET** – request whatever information is identified by the Request-URL
  - **POST** – request that server accepts the entity enclosed in the request
  - **OPTIONS** – request for information about communication options
  - **PUT** – request that the enclosed entity be stored under the Request-URL
  - **DELETE** – request that the server delete the resource identified by Request-URL
  - **TRACE** – invoke a remote, application-layer loopback of the request message
  - **CONNECT** – used by proxies in SSL connections
  - **HEAD** – identical to GET, but server must not return a message body in response

HTTP Response

- has the form:

  Http-Version SP Status-Code SP Reason-Phrase<CR><LF>
  (generic-header | response-header | entity-header <CR><LF>)
  <CR><LF>
  [message body]

- Response-header has the following fields (selection):
  - **Accept-Ranges** : server indicates its acceptance of range requests for resource
  - **Age** : sender's estimate of the amount of time since the response was generated by server
  - **Location** : redirect the client to a location other than Request-URL for completion of the request
  - **Retry-After** : indicate to client how long the service is expected to be unavailable
  - **Server** : information about software used by the server to handle the request

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

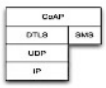
- an URL identifies a resource in the WWW
- URLs are a subset of URIs (Uniform Resource Identifiers); URL=URI that provides the location for a resource

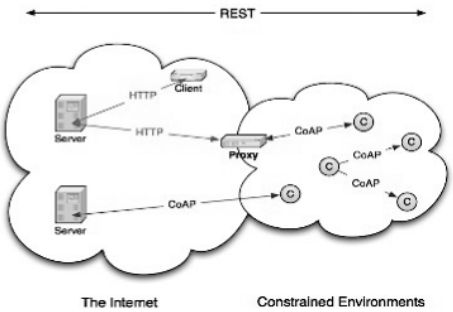
IoT Protocols - 2

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks.

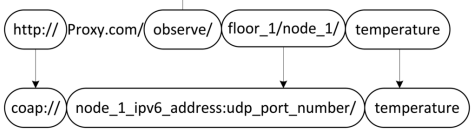
CoAP: The Web of Things Protocol

- Open IETF Standard
- Compact 4-byte Header
- UDP, SMS, (TCP) Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-in Discovery





Add the Observe Option to the CoAP Header



CoAP Overview – Characteristics

- Constrained machine-to-machine web protocol
- Representational State Transfer (REST) architecture
- Simple proxy and caching capabilities
- Asynchronous transaction support
- Low header overhead and parsing complexity
- URI and content-type support
- UDP binding (may use IPsec or DTLS)
- Reliable unicast and best-effort multicast support
- Built-in resource discovery

CoAP Layers in the Protocol Stack

- CoAP transactions provide reliable UDP messaging
- CoAP methods resemble HTTP method requests and responses
- CoAP method calls may involve multiple CoAP transactions
- Roles at the transaction layer may change during a method request / response execution

Application
CoAP Methods
CoAP Transactions
UDP
IPv6 / RPL
6LoWPAN
802.15.4

CoAP Transaction

Message	Description
CON	Confirmable requests that the receiving peer sends an acknowledgement or a reset
NON	Non-confirmable messages do not request any message being sent by the receiving peer
ACK	Acknowledges that a CON has been received, may carry payload
RST	Indicates that a CON has been received but some context is missing to process it

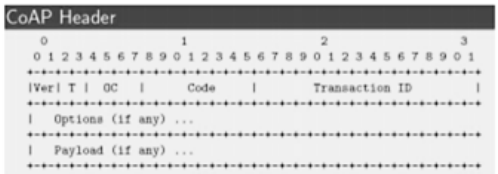
- Transactions are invoked peer to peer (not client/server)
- Transactions are identified by a Transaction ID (TID)

CoAP Method

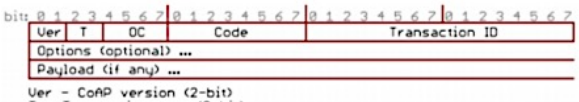
Method	Description
GET	Retrieves information of an identified resource
POST	Creates a new resource under the requested URI
PUT	Updates the resource identified by an URI
DELETE	Deletes the resource identified by an URI

- Resources are identified by URIs
- Methods are very similar to HTTP methods
- Response codes are a subset of HTTP response codes
- Options carry additional information (similar to HTTP header lines, but using a more compact encoding)

CoAP Message Header Format



- The Ver field contains the version number, the T field the message type, and the OC field the number of options.
- The Code field carries the method code / response code (methods are numbers not strings).
- The unique Transaction ID is changed for every new request but not during retransmissions.



Ver - CoAP version (2-bit)
T - Transaction type (2-bit)
OC - Option count (4-bit)
Code - Request method (1-10) or response code (40-255) (8-bit). E.g:
- GET: 1
- POST: 2
- PUT: 3
- DELETE: 4

Transaction ID - Unique identifier for matching response (16-bit)

Message types

Type	Name
0	CONfirmable
1	NON-confirmable
2	ACKnowledgement
3	ReSeT

Method codes

Code	Name
0.00	EMPTY
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

Response codes

Class	
2.xx	Success
4.xx	Client Error
5.xx	Server Error

CoAP RFCs & Testbed

- Constrained Application Protocol (CoAP)
- IETF **Constrained RESTful Environments (core)**
  - http://datatracker.ietf.org/wg/core/documents/
- CoAP Me
  - http://coap.me/
- ETSI CoAP Plugfest
- Other Implementations
  - Firefox Plugin
  - Contiki
  - Eclipse

Options

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	(see note 1)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	(see note 1)
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
17	x				Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)
60			x		Size1	uint	0-4	(none)

C=Critical, U=Unsafe, N=No-Cache-Key, R=Repeatable

Note 1: taken from destination address/port of request message

Content-Formats

Media type	Id.
text/plain;charset=utf-8	0
application/link-format	40
application/xml	41
application/octet-stream	42
application/exi	47
application/json	50
application/cbor	60

URI schemes

coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]
coaps-URI = "coaps:" "//" host [":" port] path-abempty ["?" query]

Code	Description
2.01 (65)	Created
2.02 (66)	Deleted
2.03 (67)	Valid
2.04 (68)	Changed
2.05 (69)	Content
4.00 (128)	Bad Request
4.01 (129)	Unauthorized
4.02 (130)	Bad Option
4.03 (131)	Forbidden
4.04 (132)	Not Found
4.05 (133)	Method Not Allowed
4.06 (134)	Not Acceptable
4.12 (140)	Precondition Failed
4.13 (141)	Request Entity Too Large
4.15 (143)	Unsupported Content-Format
5.00 (160)	Internal Server Error
5.01 (161)	Not Implemented
5.02 (162)	Bad Gateway
5.03 (163)	Service Unavailable
5.04 (164)	Gateway Timeout
5.05 (165)	Proxying Not Supported

IoT Protocols - 3

MQTT (MQ Telemetry Transport)

- **open source protocol** for:
  - constrained devices
  - low-bandwidth
  - high-latency networks.
- **publish/subscribe messaging transport**
  - extremely **lightweight**
  - ideal for **connecting small devices to constrained networks.**
- **bandwidth efficient**
- **data agnostic**
- **continuous session awareness.**
- MQTT is extremely **simple**, offering **few control options**
- Created by IBM /Eurotech (donated to Eclipse)

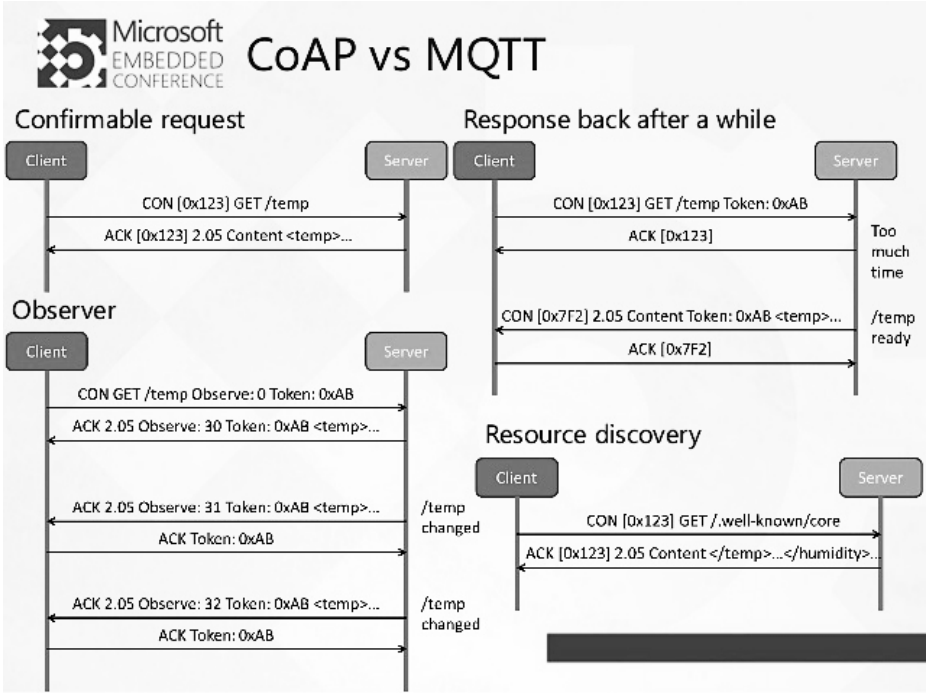
- **Minimize the resource** requirements for IoT device
- Sufficient **reliability** and **assurance** of delivery with grades of service.
  - three QoS levels
    - at most once
    - at least once
    - exactly once
- **For Large networks of small devices**
  - that need to be monitored or controlled from a back-end server on the Internet.
- **NOT designed** for
  - device-to-device transfer.
  - "**multicast**" data to many receivers.

Web Socket

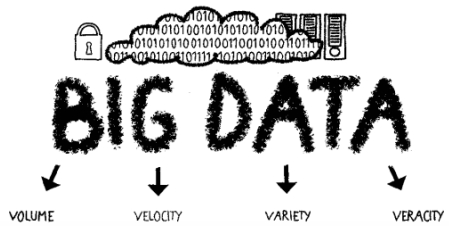
- A protocol that provides full-duplex communication over a single TCP connecti between client and server.
- Part of the HTML 5 specification.
- WebSocket standard simplifies much of th complexity around bi-directional Web communication and connection managem

XMPP (Extensible Messaging and Presence Protocol)

- Roots in instant messaging and presence information.
- Expanded into
  - signaling for VoIP
  - Collaboration
  - lightweight middleware
  - content syndication
  - generalized routing of XML data.
- Contender for mass scale management of consumer white goods such as washers, dryers, refrigerators, and so on.



Big Data



Big Data

- The ability to analyze large, complex and rapidly changing datasets comprised of structured, semi-structured or unstructured data to gain valuable insights
  - Clustering (structure/commonalities across data)
  - Associations (relationships between actions/items)
  - Classification of data
  - Regression (relationships between input and output data from a process)

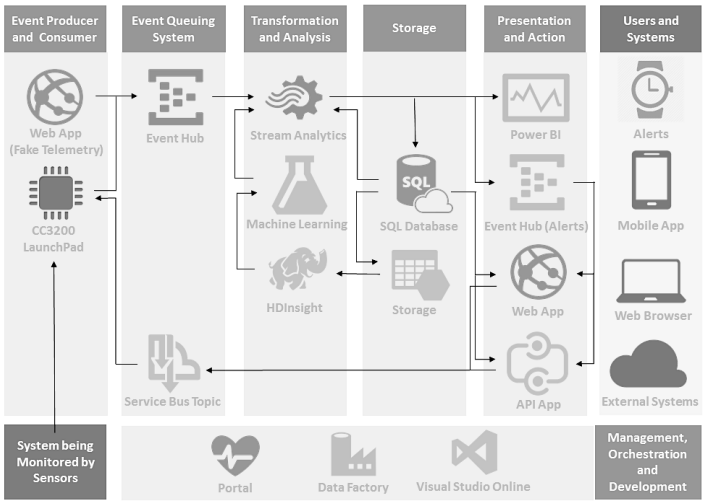
Key Data Functions required by IoT

- **Device and Infrastructure Management Platform**
  - Automated provisioning of **edge devices** to ease deployment
  - IoT requires operators to operate software on devices remotely, without taking the network of sensors out of service
- **New Platform-as-a-Service (PAAS)** offerings emerging for Big Data
  - Data storage (*e.g., supported versions of Hadoop); data warehousing and database tools*
  - Data normalization techniques to improve interoperability
  - Data analytics tools and applications: real-time and predictive
- **Data Filtering**
  - Vast amounts of data can be generated by sensors
  - Need to filter data to that which is necessary so that systems aren't overwhelmed
- **Analytics Platform**
  - Diversity of devices producing data from different locations need to be configured so that the data can be leveraged
  - Integration with the big data analytics platforms

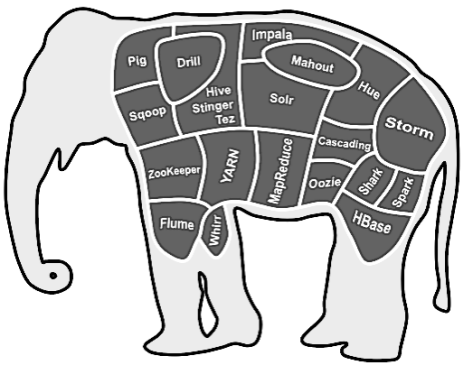
Security by Design

Security Threats in IoT

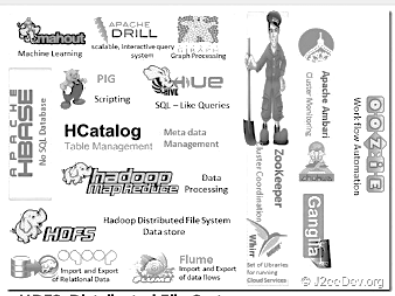
	Manufacturing	Installation/ Commissioning	Operation
Thing Itself	Device cloning	Substitution	Privacy threat Extraction of security parameters
Application Layer			Firmware replacement
Transport Layer		Eavesdropping/ man-in-the-middle attack	Eavesdropping/ man-in-the-middle attack
Network Layer		Eavesdropping/ man-in-the-middle attack	DDOS/Routing Attack
Physical Layer			DDOS



Apache Hadoop Ecosystem



Apache Hadoop Ecosystem



- **HDFS: Distributed File System**
- **MapReduce:** parallel data processing
- **Pig latin:** data flow scripting language
- **HBase:** distributed, columnar, non-relational database
- **Hive:** data warehouse infrastructure + HQL
- **ZooKeeper:** centralized service providing distributed synchronization
- **Ganglia:** monitoring system for clusters and grids
- **Sqoop:** tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores (RDBMS)
- **Hama:** distributed engine for massive scientific computations such as matrix, graph and network algorithm (BSP)
- **HCatalog:** table mgmt layer for Hive metadata to other Hadoop applications
- **Mahout:** scalable machine learning library.
- **Ambari:** software for provisioning, managing, and monitoring Apache Hadoop clusters
- **Flume:** distributed service for efficiently collecting, aggregating, and moving large amounts of log data
- **Giraph:** iterative graph processing system
- **DRILL:** low latency SQL query engine for Hadoop
- **Oozie or TEZ:** workflow automation

Philosophical Differences

Traditional Methods	Big Data
More power	More machines
Summarize data	Keep all data
Transform and store	Transform on demand
Pre-defined schema	Flexible / no schema
Move data -> compute	Move compute -> data
Less data / more complex algorithms	Mode data / simple algorithms

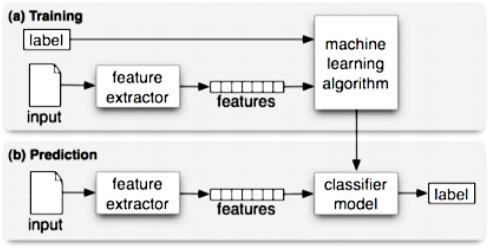
MPP Analytic Databases or Hadoop

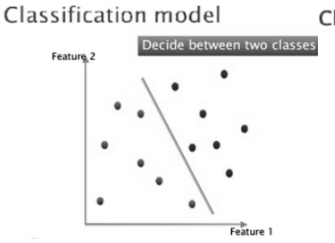
TERADATA. NETEZZA an IBM Company

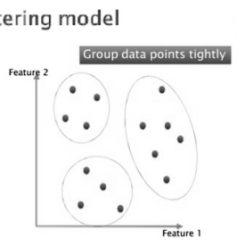
VERTICA An HP Company hadoop Greenplum

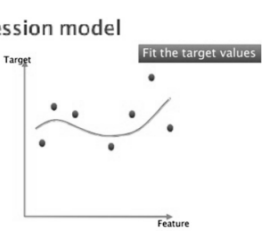
cloudera MAPR Hortonworks

Machine Learning & Data Mining









ML Categories

- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

Supervised learning
- labelled data are available

Unsupervised learning
- No labelled data is available

Semi-supervised learning
- mix of Supervised and Unsupervised learning
- usually small part of data is labelled

Reinforcement learning
- model is continuously learn and relearn based on the actions effects/rewards from that actions.
- reward feedback

	Unsupervised	Supervised
Continuous	• Clustering & Dimensionality Reduction ◦ SVD ◦ PCA ◦ K-means	• Regression ◦ Linear ◦ Polynomial • Decision Trees • Random Forests
Categorical	• Association Analysis ◦ Apriori ◦ FP-Growth • Hidden Markov Model	• Classification ◦ KNN ◦ Trees ◦ Logistic Regression ◦ Naive-Bayes ◦ SVM

Prediction Techniques

- Classification-Based Approaches
  - Nearest Neighbor
  - Neural Networks
  - Bayesian Classifiers
  - Decision Trees
- Sequential Behavior Modeling
  - Hidden Markov Models
  - Temporal Belief Networks

ML Models Applications

- Classification
  Predict amongst a discrete set of classes
  - Face Recognition, Spam Filter, Text Classification
- Regression
  Predict real/numeric values
  - Stock Marekt
- Clustering
  Grouping similar items
  - Clustering Web Search Results...
- Similarity
  Find things like this
  - Similar Products (people are buying)
  - Similar Images
- Recommender systems
  Learn what I want before I know it
  - Playlist Recommendation
  - Friends Recommendation

ML Tools/Libs

- Weka ToolKit (linked to Hadoop)
  - ■ Preprocessing data
  - ■ Clustering
  - ■ Classification
  - ■ Regression
  - ■ Association rules
- Mahout
  - ■ Naive Bayes Classifier
  - ■ K Means Clustering
  - ■ Recommendation Engines
  - ■ Random Forest Decision Trees
  - ■ Logistic Regression Classifier

Objectives of Data Mining and Prediction

- Large amounts of sensor data have to be "interpreted" to acquire knowledge about tasks that occur in the environment
- Patterns in the data can be used to predict future events
- Knowledge of tasks facilitates the automation of task components to improve the inhabitants' experience

Example Patterns

- Associative pattern
  When Bob is in the living room he likes to watch TV and eat popcorn with the light turned off.
- Classification
  Action movie fans like to watch Terminator, drink beer, and have pizza.
- Sequential patterns
  After coming out of the bedroom in the morning, Bob turns off the bedroom lights, then goes to the kitchen where he makes coffee, and then leaves the house.

Data Mining and Prediction

Prediction attempts to form patterns that permit it to predict the next event(s) given the available input data.

- **Deterministic predictions**
  If Bob leaves the bedroom before 7:00 am on a workday, then he will make coffee in the kitchen.
- **Probabilistic sequence models**
  If Bob turns on the TV in the evening then he will 80% of the time go to the kitchen to make popcorn.

Decision Trees

- Advantages
  - Understandable rules
  - Fast learning and prediction
  - Lower memory requirements
- Problems
  - Replication problem (each category requires multiple branches)
  - Limited rule representation (attributes are assumed to be locally independent)
  - Numeric attributes can lead to large branching factors

Neural Networks

- Advantages
  - General purpose learner (can learn arbitrary categories)
  - Fast prediction
- Problems
  - All inputs have to be translated into numeric inputs
  - Slow training
  - Learning might result in a local optimum

Bayes Classifier

- Advantages
  - Yields optimal prediction (given the assumptions)
  - Can handle discrete or numeric attribute values
  - Naive Bayes classifier easy to compute
- Problems
  - Optimal Bayes classifier computationally intractable
  - Naive Bayes assumption usually violated

Bayesian Networks

- Advantages
  - Efficient inference mechanism
  - Readable structure
  - For many problems relatively easy to design by hand
  - Mechanisms for learning network structure exist
- Problems
  - Building network automatically is complex
  - Does not handle sequence information

Sequence Prediction Techniques

String matching algorithms
- Deterministic best match
- Probabilistic matching

Markov Models
- Markov Chains
- Hidden Markov Models

Dynamic Belief Networks

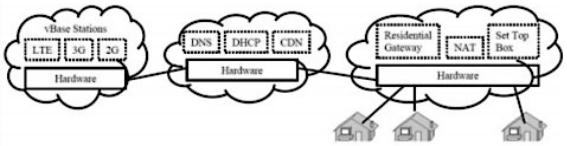
Markov Models

- Advantages
  - Permits probabilistic predictions
  - Transition probabilities are easy to learn
  - Representation is easy to interpret
- Problems
  - State space has to have Markov property
  - State space selection is not automatic
  - States might have to include previous information
  - State attributes might not be observable

SDN / NFV

Network Function Virtualization (NFV)

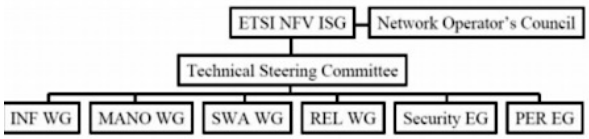
1. Fast standard hardware ⇒ **Software based Devices**
   Routers, Firewalls, Broadband Remote Access Server (BRAS)
   ⇒ A.k.a. *white box* implementation
2. **Function Modules** (Both data plane and control plane)
   ⇒ DHCP (Dynamic Host control Protocol), NAT (Network Address Translation), Rate Limiting,



3. **Virtual Machine implementation**
   ⇒ Virtual appliances
   ⇒ All advantages of virtualization (quick provisioning, scalability, mobility, Reduced CapEx, Reduced OpEx, …)

4. **Standard APIs**: New ISG (Industry Specification Group) in ETSI (European Telecom Standards Institute) set up in November 2012

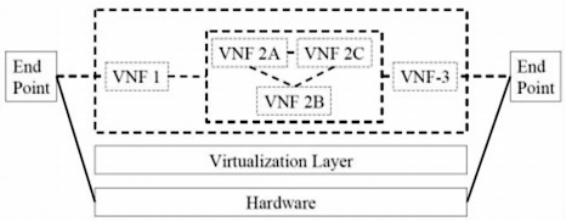
ETSI NFV ISG



- Industry Specification Group (ISG)'s goal is to define the requirements.
- Four Working Groups:
  - **INF**: Architecture for the virtualization Infrastructure
  - **MANO**: Management and orchestration
  - **SWA**: Software architecture
  - **REL**: Reliability and Availability, resilience and fault tolerance

Network Forwarding Graph

- An end-to-end service may include nested forwarding graphs



NFV Use Cases

- **Cloud**:
  1. NFV infrastructure as a service (NFVIaaS) like IaaS
  2. Virtual Network Functions (VNFs) as a service (VNFaaS) like SaaS
  3. VNF forwarding graphs (Service Chains)
  4. Virtual Network Platform as a Service (VNPaaS) like PaaS
- **Mobile**:
  5. Virtualization of the Mobile Core Network and IMS
  6. Virtualization of Mobile Base Station
- **Data Center**:
  7. Virtualization of CDNs
- **Access/Residential**:
  8. Virtualization of the Home environment
  9. Fixed Access NFV

Why We need NFV?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Manage thousands of devices
3. **Programmable**: Should be able to change behavior on the fly.
4. **Dynamic Scaling**: Should be able to change size, quantity
5. **Automation**
6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
8. **Multi-tenancy**
9. **Service Integration**
10. **Openness**: Full choice of Modular plug-ins

Note: These are exactly the same reasons why we need SDN.

NFV and SDN Relationship

- Concept of NFV originated from SDN
  ⇒ First ETSI white paper showed overlapping Venn diagram
  ⇒ It was removed in the second version of the white paper
- NFV and SDN are complementary.
  One does not depend upon the other.
  You can do SDN only, NFV only, or SDN and NFV.
- Both have similar goals but approaches are very different.
- SDN needs new interfaces, control modules, applications.
  NFV requires moving network applications from dedicated hardware to virtual containers on commercial-off-the-shelf (COTS) hardware
- NFV is present. SDN is the future.
- Virtualization alone provides many of the required features
- Not much debate about NFV.
- Two Expert Groups:
  - **Security** Expert Group: Security
  - **Performance and Portability** Expert Group: Scalability, efficiency, and performance VNFs relative to current dedicated hardware

NFV Concepts

- **Network Function (NF):** Functional building block with a well defined interfaces and well defined functional behavior
- **Virtualized Network Function (VNF)**: Software implementation of NF that can be deployed in a virtualized infrastructure
- **VNF Set**: Connectivity between VNFs is not specified, e.g., residential gateways
- **VNF Forwarding Graph**: Service chain when network connectivity order is important, e.g., firewall, NAT, load balancer
- **NFV Infrastructure (NFVI)**: Hardware and software required to deploy, mange and execute VNFs including computation, networking, and storage.
- **NFVI Point of Presence (PoP)**: Location of NFVI
- **NFVI-PoP Network**: Internal network
- **Transport Network**: Network connecting a PoP to other PoPs or external networks
- **VNF Manager**: VNF lifecycle management e.g., instantiation, update, scaling, query, monitoring, fault diagnosis, healing, termination
- **Virtualized Infrastructure Manager**: Management of computing, storage, network, software resources
- **Network Service**: A composition of network functions and defined by its functional and behavioral specification
- **NFV Service**: A network services using NFs with at least one VNF.

SDN - 2

Addressing

Some Special Addresses

Type	Binary	Hex
Aggregatable Global Unicast Address	001	2 or 3
Link-Local Unicast Address	1111 1110 10	FE80::/10
Unique Local Unicast Address	1111 1100 1111 1101	FC00::/7 FC00::/8(Registry) FD00::/8 (No Registry)
Multicast Address	1111 1111	FF00::/8

IPv4-IPv6 Transition/Coexistence

- A wide range of techniques have been identified and implemented, basically falling into three categories:
  1. Dual-stack techniques, to allow IPv4 and IPv6 to co-exist in the same devices and networks
  2. Tunneling techniques, to avoid order dependencies when upgrading hosts, routers, or regions
  3. Translation techniques, to allow IPv6-only devices to communicate with IPv4-only devices
- Expect all of these to be used, in combination

Tunneling

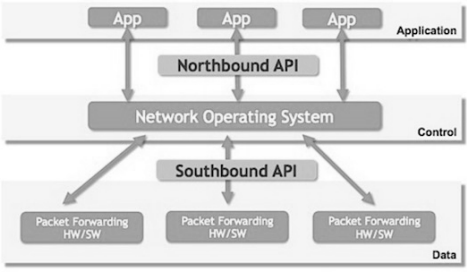
Many Ways to Do Tunneling

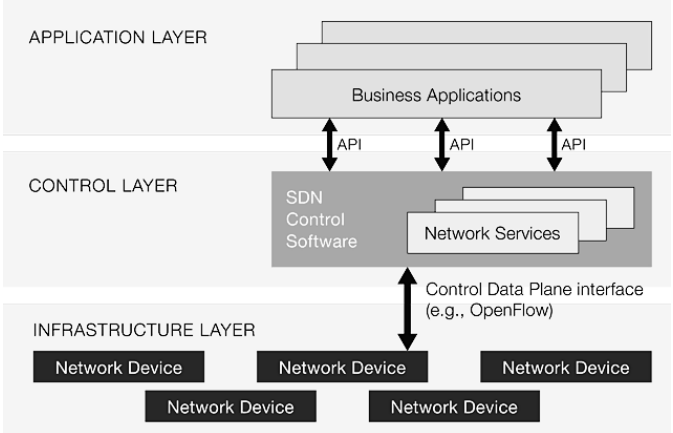
- Some ideas same as before
  - GRE, MPLS, IP
- Native IP over data link layers
  - ATM PVC, dWDM Lambda, Frame Relay PVC, serial, Sonet/SDH, Ethernet
- Some new techniques
  - Automatic tunnels using IPv4, compatible IPv6 address 6to4, ISATAP

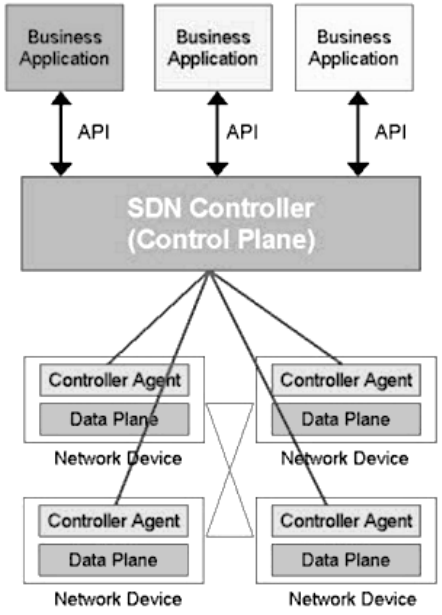
What is SDN?

- Recent trends in communications networking have made it possible to control the behavior of entire networks from a single, high-level software program.
- This trend, called software-defined networking (**SDN**), is reshaping the way networks are designed, managed, and secured.
- This new field of networking is still evolving for OpenFlow Switches/Controllers (NOX, FloodLight, and OpenDayLight).
- Cloud (OpenStack) and SDN (OpenFlow) integration is: "Network Connectivity as a Service - **NaaS**" (Quantum/Neutron)

SDN Reference Model



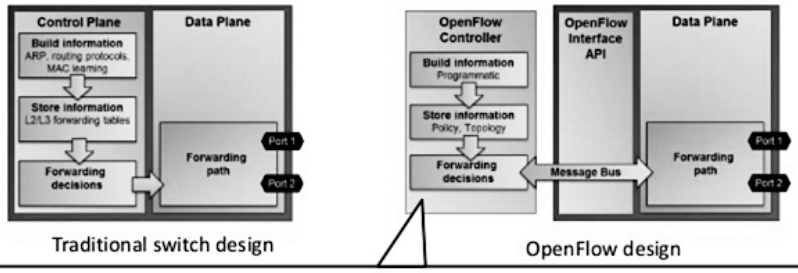




Software-Defined Networking

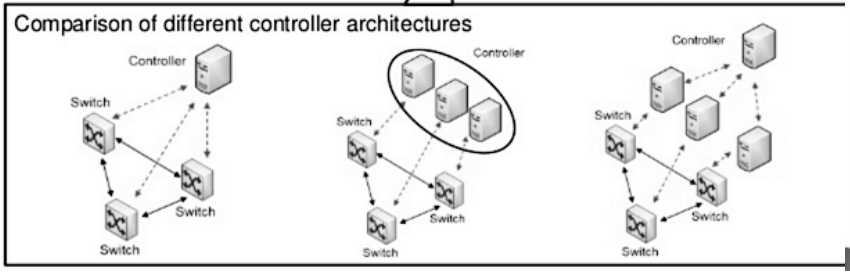
SDN: Software Defined Networking

Technology that enables data center team to use software to efficiently control network resources



Traditional switch design

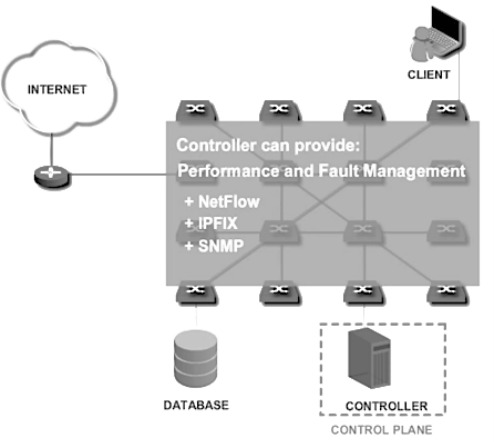
OpenFlow design

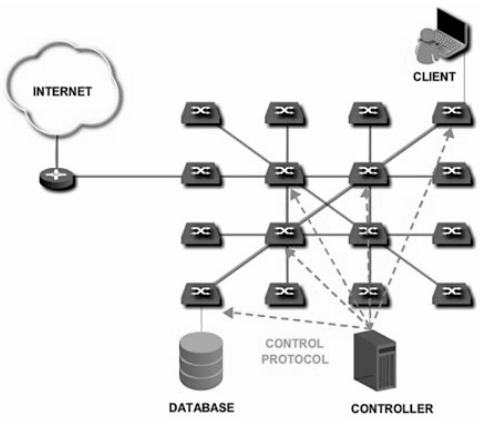


SDN Lab SW Tools

1. **OpenStack** Framework
2. **OpenDayLighyt** - SDN Controller
3. **FloodLight** - SDN Controller
4. **Open vSwitch** - Virtual Switch
5. **MiniNet** - Virtual Network: OpenFlow Switches, SDN Controllers, and Servers/Hosts
6. **OMNet++** Network Simulator
7. **Avlor** - Sample FloodLight Java Application
8. **NOX/POX** - C++/ Python OpenFlow API for building network control applications
9. **Pyretic** = Python + Frenetic - Enables network programmers and operators to write modular network applications by providing powerful abstractions
10. **Resonance** - Event-Driven Control for Software-Defined Networks (written in Pyretic)
11. **Trema** - Full-Stack OpenFlow Framework in Ruby and C
12. **FlowScale** - Project to divide and distribute traffic over multiple physical switch ports.
13. **SNAC** - Open source OpenFlow controller for LANs with a graphical user interface.

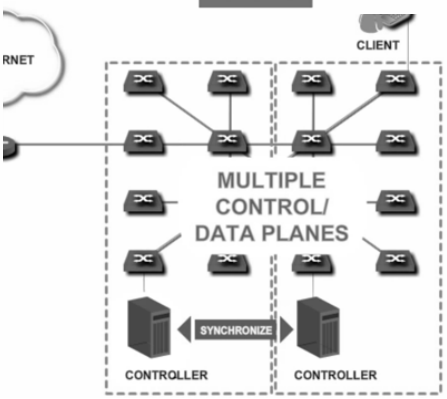
SDN - 3

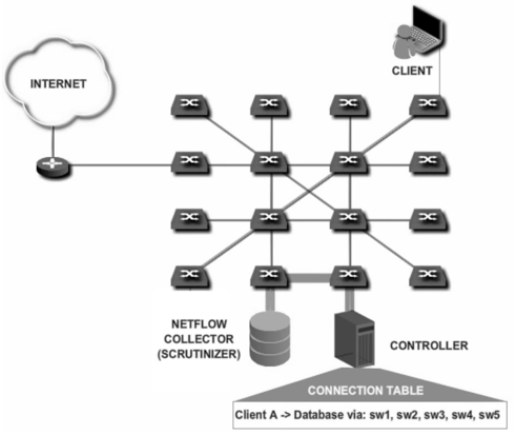


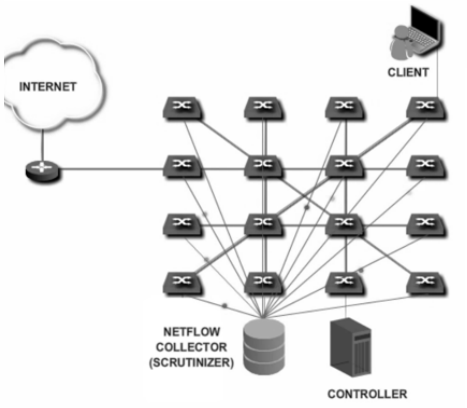


Big Companies Support SDN

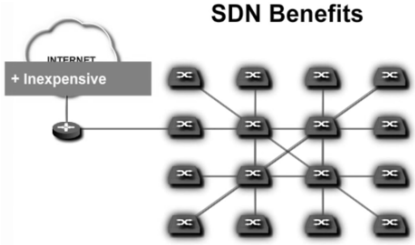
Deutsche Telekom Google verizon YAHOO! facebook Microsoft



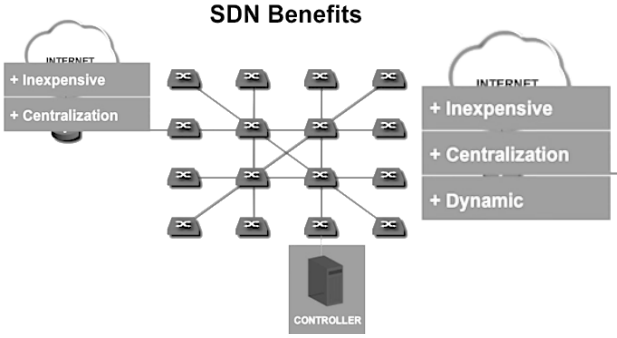




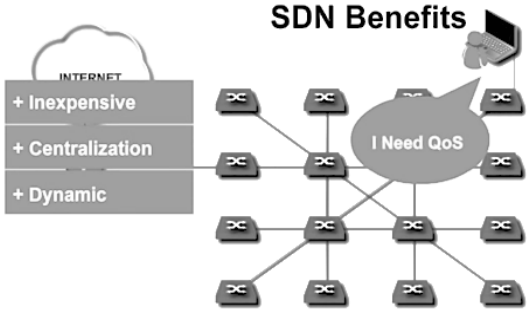
SDN Benefits



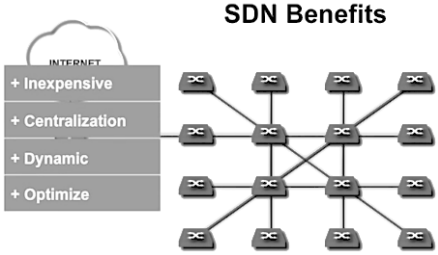
SDN Benefits



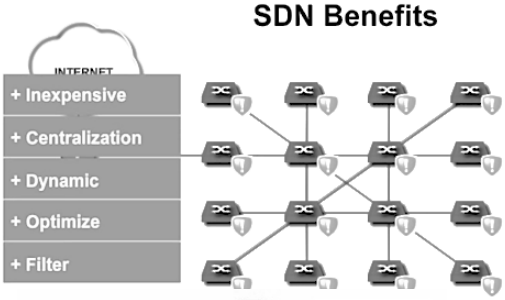
SDN Benefits



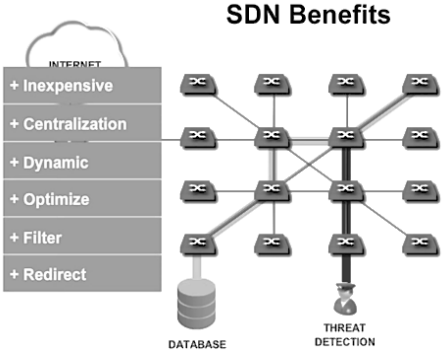
SDN Benefits



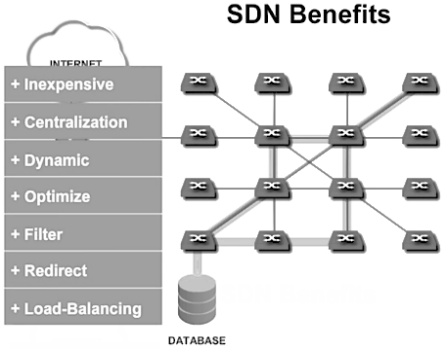
SDN Benefits

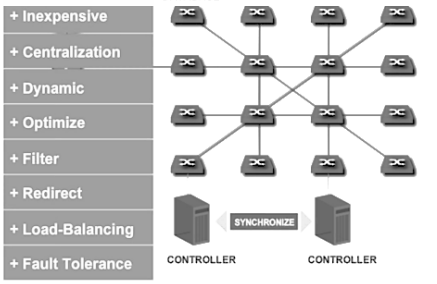


SDN Benefits



SDN Benefits

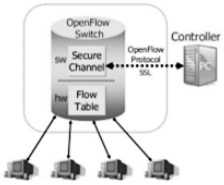


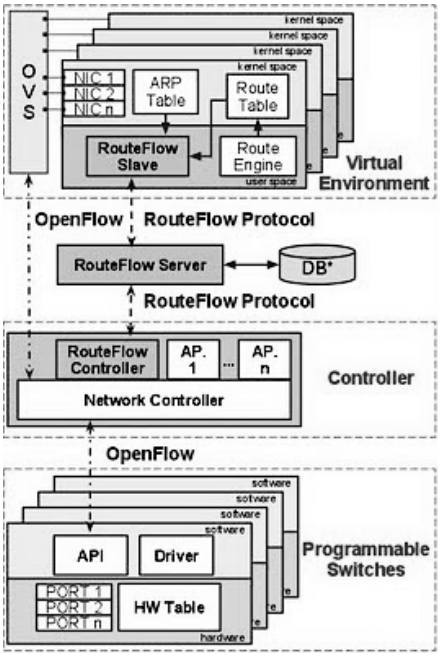


OpenFlow

What is OpenFlow?

- OpenFlow is a communication protocol that enables SDN(Software-defined networking) by getting access to the control plane of the switch or router.





- Virtual network environment
  - Each VM (virtual machine) maps to one OpenFlow hardware.
  - Each VM runs one instance of the IP routing engine (Any linux based routing engine: e.g. Quagga).
- OVS (Open vSwitch)
  - A software switch that supports OpenFlow.
  - Depends on the RouteFlow controller to make the decisions.
- RouteFlow Controller (RF-C)
  - Controller : NOX controller .
  - Sends packets and events its receives from OVS and OpenFlow hardware to RouteFlow Server.
- RouteFlow Server (RF-Server)
  - Keeps the core logic of the system.
  - Receives the registered events from the RF-C and takes decisions over those events (e.g. packet-in, datapath- join).
  - Also receives information about route changes from the rf-slave running in the quagga vms, which will trigger a flow install/modification in the corresponding OpenFlow switch.
- RouteFlow Slave (RF-Slave)
  - FIB(forwarding information base) : IP and ARP tables are collected (using the Linux Netlink API) and then translated into OpenFlow tuples.

What is OpenStack?

OpenStack Cloud Platform

- Allows anyone to build and deploy their own cloud
  - Cloud Providers, Enterprise Private Clouds, Service Providers
- Compute Service, Storage Service, and Networking Service

User Applications

openstack

What is OpenDaylight

OpenDaylight is an **Open Source Software** project under the **Linux Foundation** with the goal of furthering the adoption and innovation of **Software Defined Networking (SDN)** through the creation of a common industry supported platform

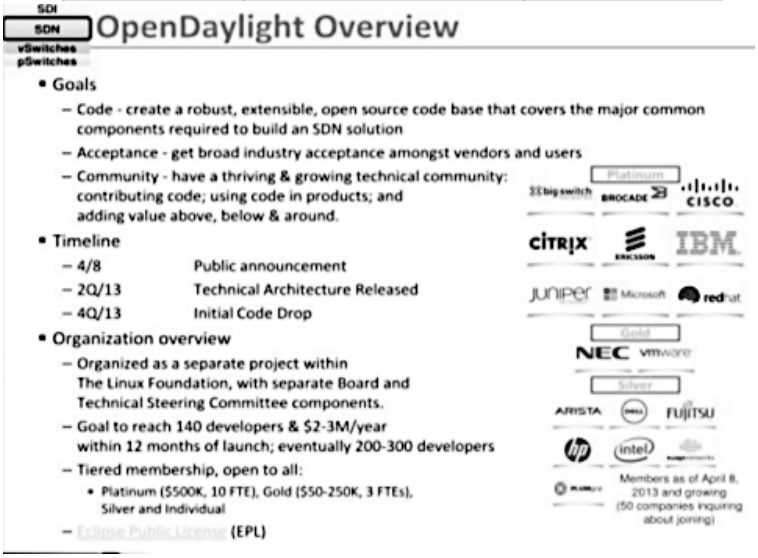
Code	Acceptance	Community
To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution	To get broad industry acceptance amongst vendors and users • Using OpenDaylight code directly or through vendor products • Vendors using OpenDaylight code as part of commercial products	To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.

What is OpenDaylight building?

An evolvable SDN platform capable of handling diverse use cases and implementation approaches

- Common abstractions for people to program
  - "Northbound" Interfaces
- Southbound "drivers", e.g., OpenFlow, OVSDB, BGP-LS
- Intermediation between north and south

- Programmable Network services
- Network Applications
- Whatever else we need to make it work



SDN – NetFPGA Misc

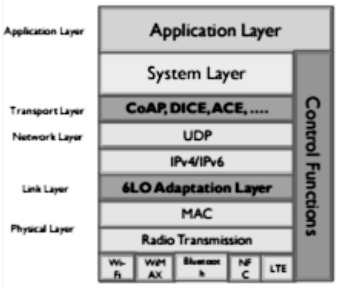
What is NetFPGA?

- Originally designed as a tool for education, the 1G platform consisted of a PCI board with a Xilinx Virtex-II pro FPGA and 4 x 1GigE interfaces feeding into it, along with a downloadable code repository containing an IP library and a few example designs.
- A board with 10GigE is also available
- It costs about $1000 per 1GigE unit and $3000 per 10GigE unit
- Sample software code for Ethernet and OpenFlow Switches is Available.

6Lo Functions

- IPv6 over Networks of Resource-constrained Nodes (6Lo WG)
- IPv6-over-foo adaptation layer specifications using 6LoWPAN technologies (RFC4944, RFC6282, RFC6775 ….)
  - →Transmission of IPv6 Packets over BLUETOOTH(R) Low Energy
  - →Transmission of IPv6 Packets over DECT Ultra Low Energy
  - →Transmission of IPv6 over MS/TP Networks
  - →Transmission of IPv6 packets over ITU-T G.9959 Networks
  - →Transmission of IPv6 Packets over IEEE 1901.2 Narrowband Powerline Communication Networks
  - →Transmission of IPv6 Packets over Near Field Communication (NFC)

Variety of IoT Protocols

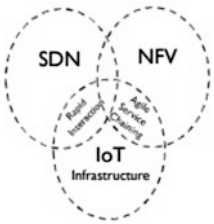


- Various Physical Layers
  - WiFi, WiMAX, BLE, NFC, LTE, …
- Various 6LO Functions
  - IPv6-over-foo adaptation layer using 6LoWPAN technologies (RFC4944, RFC6282, RFC6775 ..)
- Constrained Application Protocol
  - RFC 7252 CoAP and related mapping protocols
- Constrained Security Protocols
  - DTLS In Constrained Environments (DICE, draft-ietf-dice-profile-05)
  - Authentication and Authorization for Constrained Environments (ACE, Work-in-Progress)

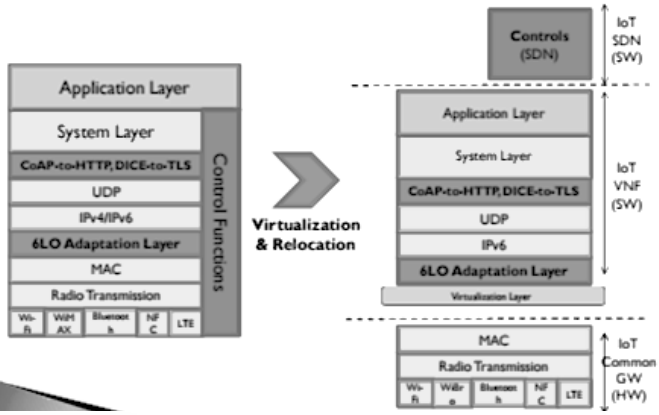
Note that we will mainly focus on end-to-end networking to resource-constrained nodes using 6LO, CoAP, DICE, RIOT protocols, etc.

Our Approach

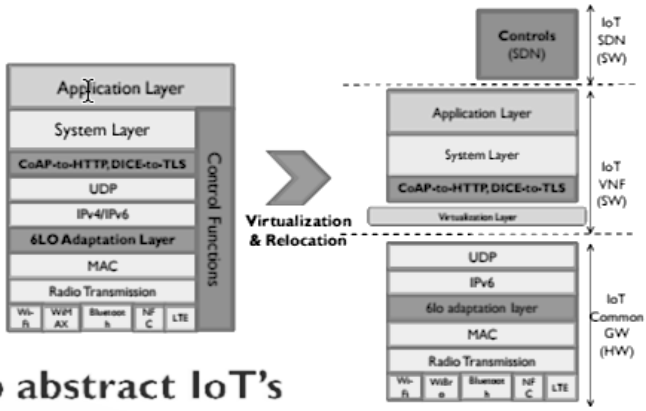
- SDN and NFV can solve those problems and challenges.
  - SDIoT : A Software-defined end-to-end IoT infrastructure (including aka, IoT service chaining support)
- IoT Infrastructure could be built by means of NFV with integration of SDN which makes it more agile.
- SDN and NFV will be enablers for new IoT Infrastructure.



1) How to relocate IoT GW functions ?



1) How to relocate IoT GW functions ?



NFV IoT GW Functions Candidates

- IoT DPI functions
- L2~L3
  - IP mapping function for non-IP nodes
  - 6LO functions (IPv6 Packets over WPAN, BT, Low Power Wi-Fi, NFC, etc.)
    - RFC4944, RFC6282, RFC6775, and
    - Many other WG I-Ds (work-in-progresses)
- L4~L7
  - CoAP-HTTP protocols mapping <draft-ietf-
  - DICE-TLS protocols mapping …

2) How to abstract IoT's behaviors by SDN Concept ?