NETWORKING ARCHITECTURES FOR SMART CITIES

CHALLENGES AND PERSPECTIVES

Prof. Stênio Fernandes, PhD

Centro de Informática U·F·P·E

CIMS³ 29 Nov. - 7 Dic. 2016 BUENOS AIRES, ARGENTINA

WORKSHOP Ciudades Inteligentes: Modelado y Simulación de Sociedades Sustentables

VIRTUS IMPAVIDA

UNIVERSIDADE FEDERAL DE PERNAMBUCO

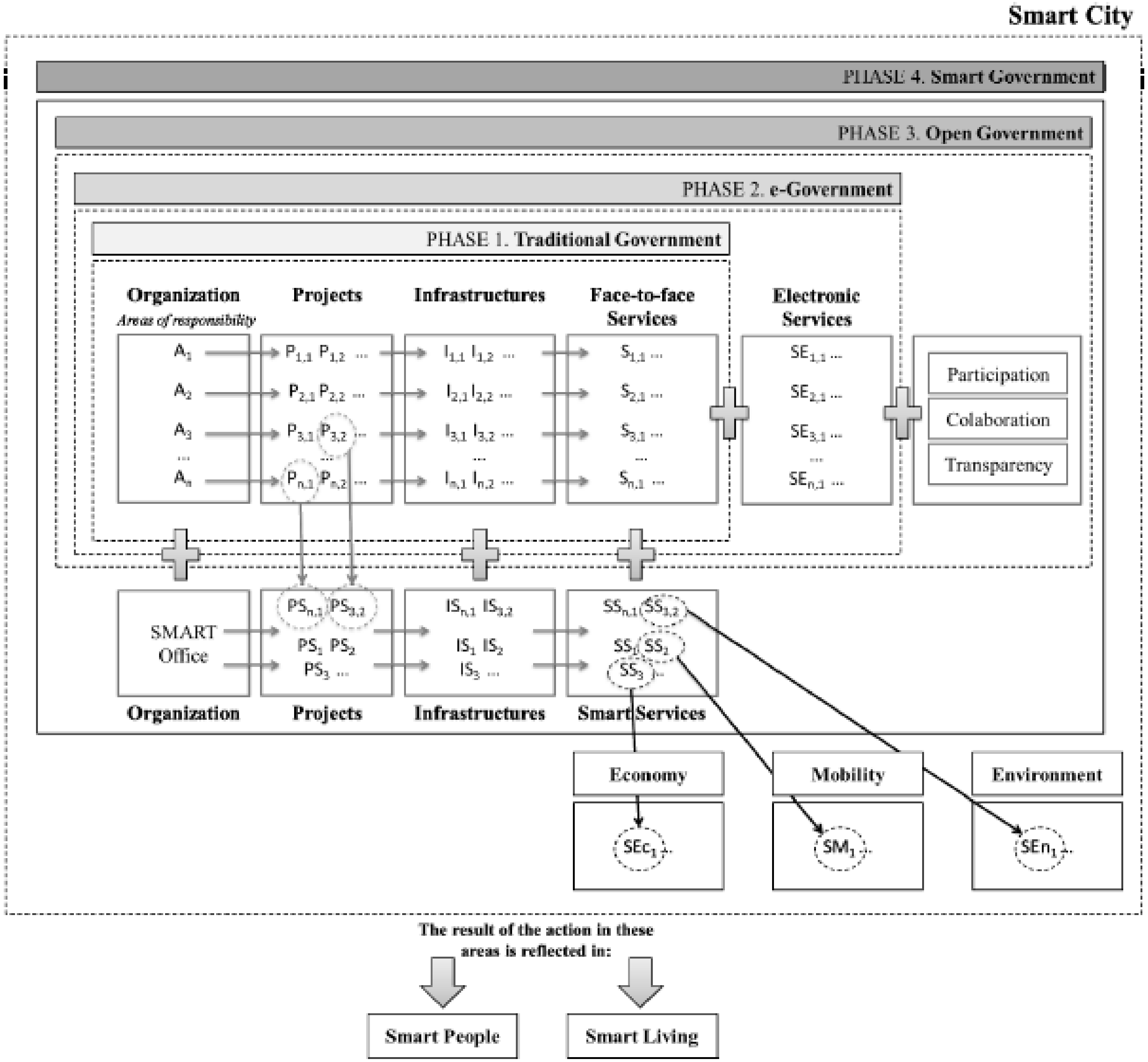
THE SMART CITY REVOLUTION

SUPPORT FROM THE NETWORKING AND TELECOMMUNICATIONS INFRASTRUCTURES

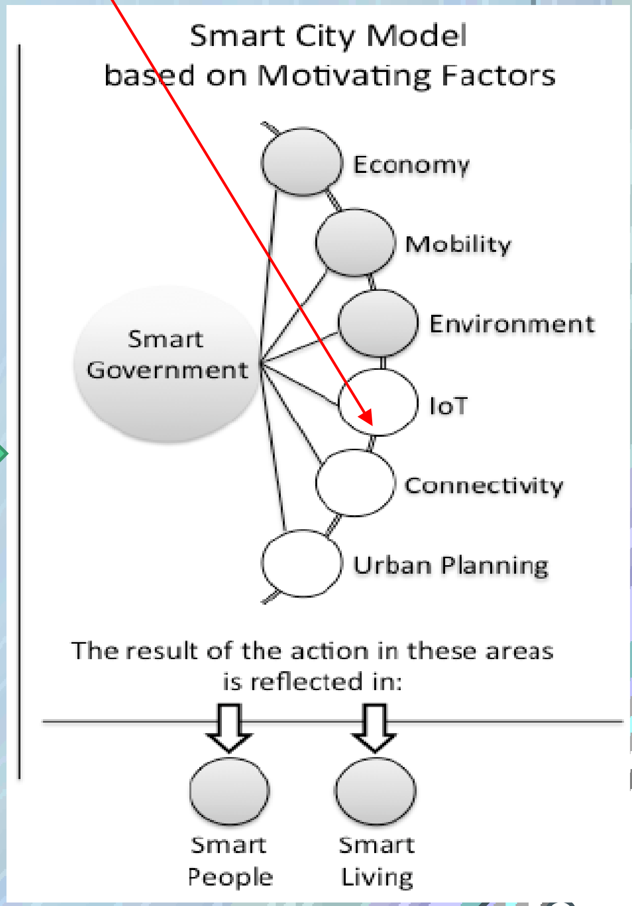
CITY TO SMART CITY

- What really drives a city to become smart?
- There is a need
  - To analyze and systematize the change process of smart cities
    - Based on a set of elements called transformation factors
  - To develop an approach to measure smartness
    - by considering transformations, not a measurement of actions

Source: Fully Smart Cities: Analysis of the City to Smart



Our Context



Source: Fully Smart Cities: Analysis of the City to Smart

CITY TO SMART CITY

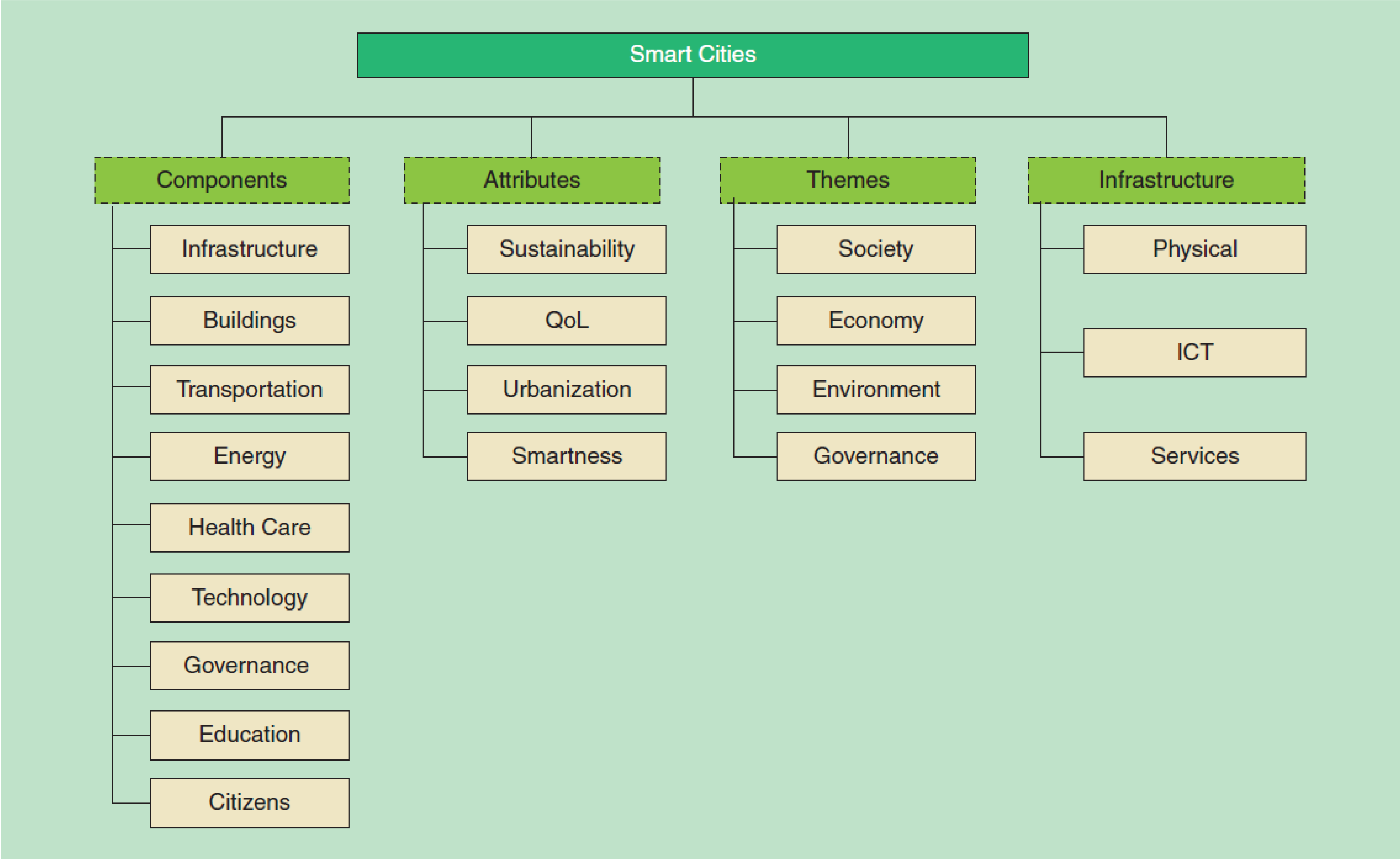


FIGURE 2. The components and characteristics of smart cities.

[illegible] ...nted to Know About Smart Cities: *The Internet of Things is the backbone.*

COMMON SMART CITY LOOP

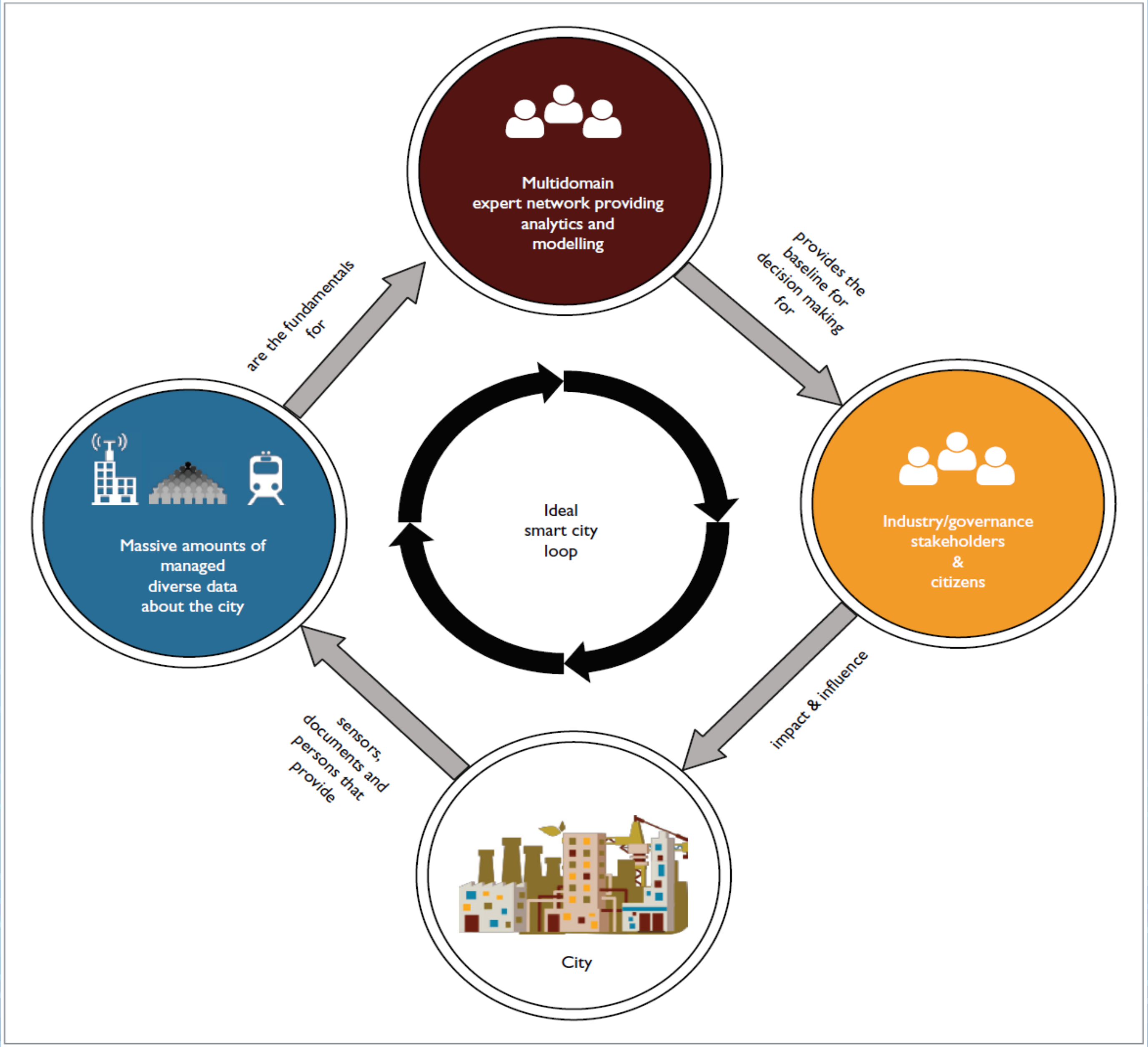


Figure 1. The ideal smart city loop. A conceptual model for a reactive system that addresses the challenges of today's smart city applications.

Source: Enabling a Smart City Application Ecosystem: *Requiremen*

SUPPORTING NETWORK TECHNOLOGIES FOR SMART CITIES

SENSORS, WIRELESS SENSORS NETWORKS, IOT, MIDDLEWARES, PROTOCOLS

A LAYERED VIEW

- From research to implementation and real mass-scale deployment
- Three different domains/layers: link/network, transport/application, and users
- Network level
  - A number of wireless technologies available to connect the city sensors and actuators to the Internet
    - LoRa, WiFi, ZigBee
    - Strengths and weaknesses: range, data rates and energy consumption.
  - It is unclear if and which technology is going to win the race for the SC wireless technology
  - SC will likely to be characterized by a multi-wireless technology environment

Source: City of Things: an Integrated and Multi-Technology Testbed for

A LAYERED VIEW

- Data level
  - Lots of sensors and technologies available
  - Lots of big data tools and implementations
    - Spark/Hadoop, NoSQL databases
  - Lack of interoperability between sensors and tools is often a limiting factor
  - There is therefore a need for more automation and flexibility for SC data experiments
- User level
  - Assess new smart city business models
    - Users' willingness to pay for smart city services
    - How cities can incentive users in a SC model

Source: City of Things: an Integrated and Multi-Technology Testbed fo[illegible]

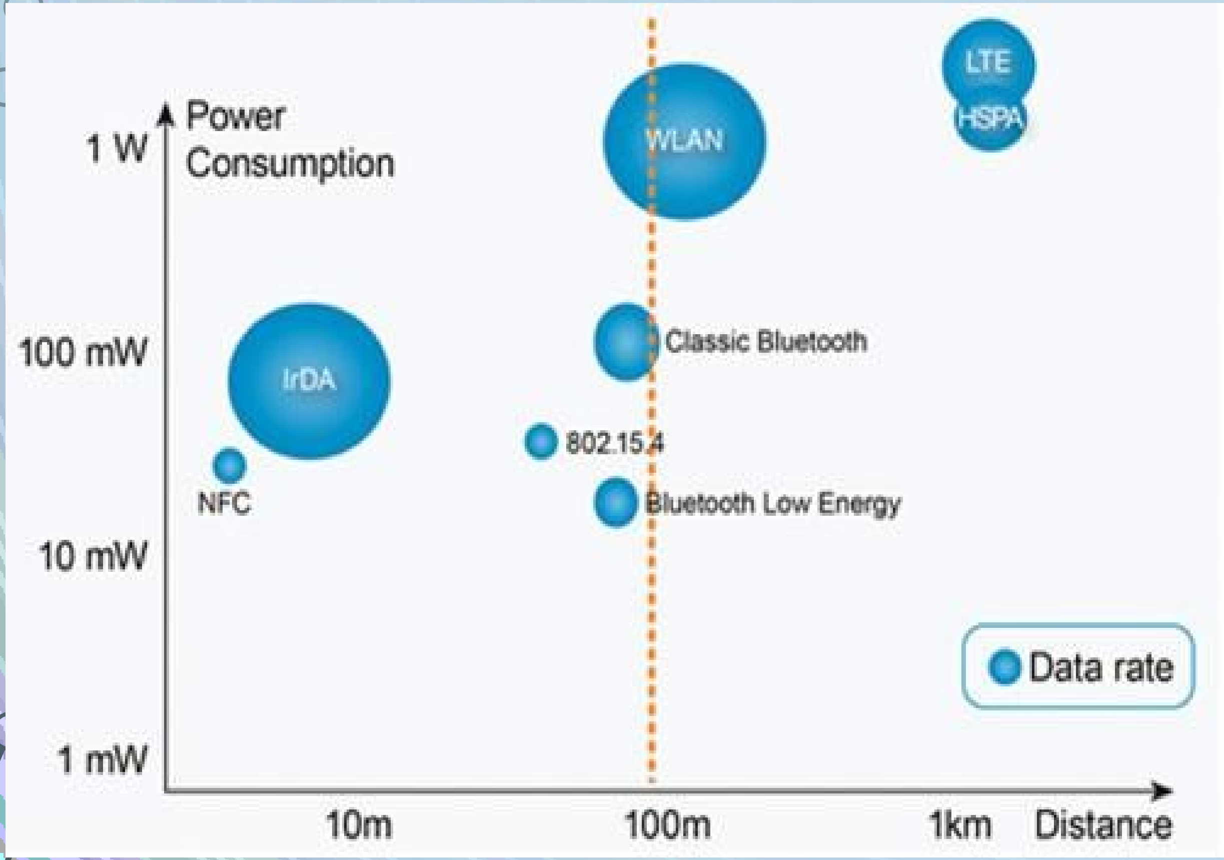
- Emerging communication technologies
  - different technologies from which the smart city environment can benefit from
    - RFID (active, passive, or battery-assisted passive)
      - transmits only in the presence of RFID reader
    - WSN: a network of distributed autonomous sensing nodes
      - low-power integrated circuits and wireless communication technology
      - can cope with large-scale deployment
      - faces the challenges of energy consumption

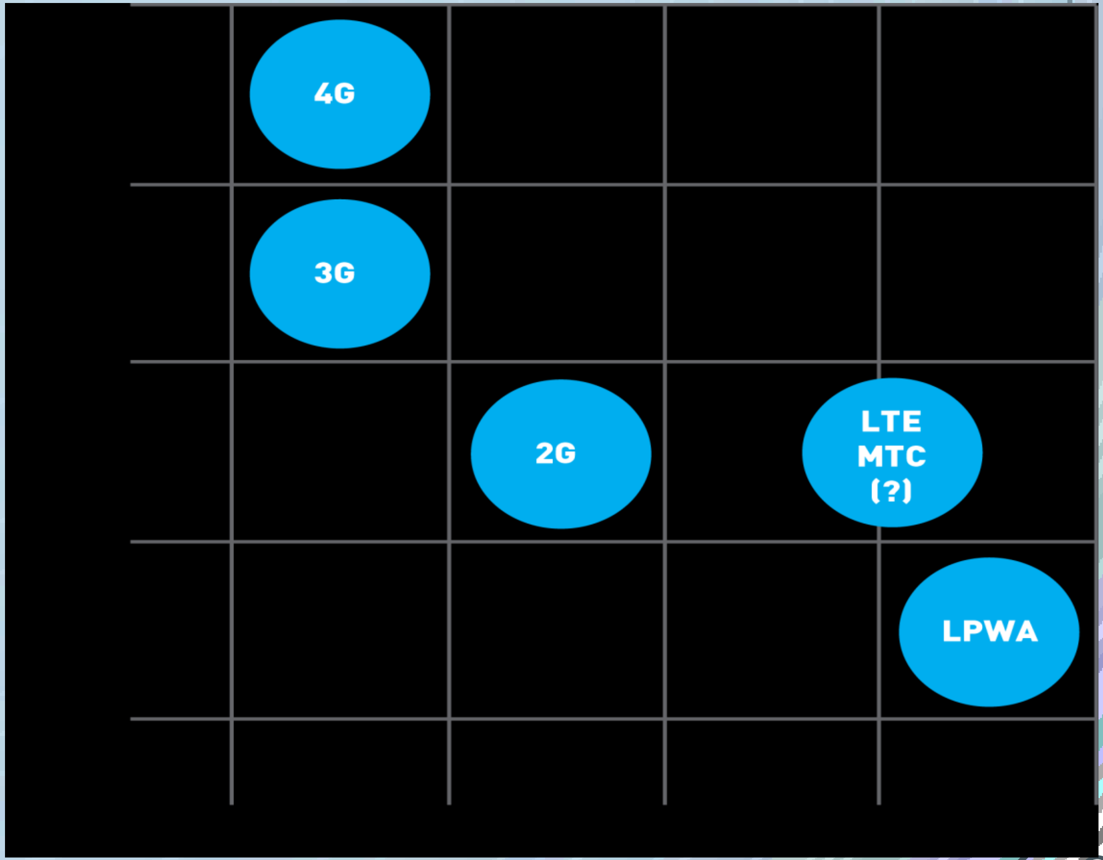
- Emerging communication technologies
  - WiFi, Ultra-wideband, ZigBee, and Bluetooth
    - WiFi – no need to explain
    - Ultra-wideband: high-bandwidth indoor short-range wireless networks over multimedia links
    - ZigBee: short-range wireless communication with provision for long lifetime battery usage capability
    - Bluetooth: standard based on a wireless radio system designed for short-range and non-expensive devices

- Emerging communication technologies
  - 4G LTE, LTE-A, and 5G
    - 4G: multiple-input multiple-output (MIMO) and orthogonal frequency divisionmultiplex (OFDM) to acquire more data throughput than 3G
    - LTE-A
      - bridges the gap between 4G and 5G
      - high bandwidths (3x the basic LTE), carrier aggregation, increased MIMO, coordinated multipoint, relay station, and HetNets
    - 5G: promise - 10 Gbit/sec with low latency

IOT WIRELESS TECHNOLOGIES							
Technologies	Standards & Organizations	Network Type	Frequency (US)	Max Range	Max Data Rate	Max Power	Encryption
WiFi	IEEE 802.11 (a,b,g,n,ac,ad, and etc)	WLAN	2.4,3.6,5,60 GHz	100 m	"6-780 Mb/s 6.75 Gb/s @ 60 GHz"	1 W	WEP, WPA, WPA2
Z-Wave	Z-Wave	Mesh	908.42 MHz	30 m	100 kb/s	1 mW	Triple DES
Bluetooth	Bluetooth (formerly IEEE 802.15.1)	WPAN	2400-2483.5 MHz	100 m	1-3 Mb/s	1 W	56/128-bit
Bluetooth Smart (BLE)	IoT Interconnect	WPAN	2400-2483.5 MHz	35 m	1 Mb/s	10 mW	128-bit AES
Zigbee	IEEE 802.15.4	Mesh	2400-2483.5 MHz	160 m	250 kb/s	100 mW	128-bit AES
THREAD	IEEE 802.15.4 + 6LoWPAN	Mesh	2400-2483.5 MHz	160 m	250 kb/s	100 mW	128-bit AES
RFID	Many	P2P	13.56 MHz, etc.	1 m	423 kb/s	~1 mW	possible
NFC	ISO/IEC 13157 & etc	P2P	13.56 MHz	0.1 m	424 kb/s	1-2 mW	possible
GPRS (2G)	3GPP	GERAN	GSM 850/1900 MHz	25 km / 10 km	171 kb/s	2 W / 1 W	GEA2/GEA3/GEA4
EDGE (2G)	3GPP	GERAN	GSM 850/1900 MHz	26 km / 10 km	384 kb/s	3 W / 1 W	A5/4, A5/3
UMTS (3G) HSDPA/HSUPA	3GPP	UTRAN	850/1700/1900 MHz	27 km / 10 km	0.73-56 Mb/s	4 W / 1 W	USIM
LTE (4G)	3GPP	GERAN/UTRAN	700-2600 MHz	28 km / 10 km	0.1-1 Gb/s	5 W / 1 W	SNOW 3G Stream Cipher
ANT+	ANT+ Alliance	WSN	2.4 GHz	100 m	1 Mb/s	1 mW	AES-128
Cognitive Radio	IEEE 802.22 WG	WRAN	54-862 MHz	100 km	24 Mb/s	1 W	AES-GCM
Weightless-N/W	Weightless SIG	LPWAN	700/900 MHz	5 km	0.001-10 Mb/s	40 mW / 4 W	128-bit

http://mwrf.com/systems/4-major-m2m-and-io





http://www.analysysmason.com/About-Us/News/Insight/For-IoT-CSPs-may-need-multiple-networks-each-optimised-for-a-different-use-case-/
http://www.digikey.com/en/articles/techzone/2014/jan/short-range-low-power-wireless-devices-and-internet-of-things-iot

URBAN IOT SYSTEM

- Web Service Approach for IoT Service Architecture
  - IETF standards for IoT
  - Designed in accordance with the ReST (Representational State Transfer)

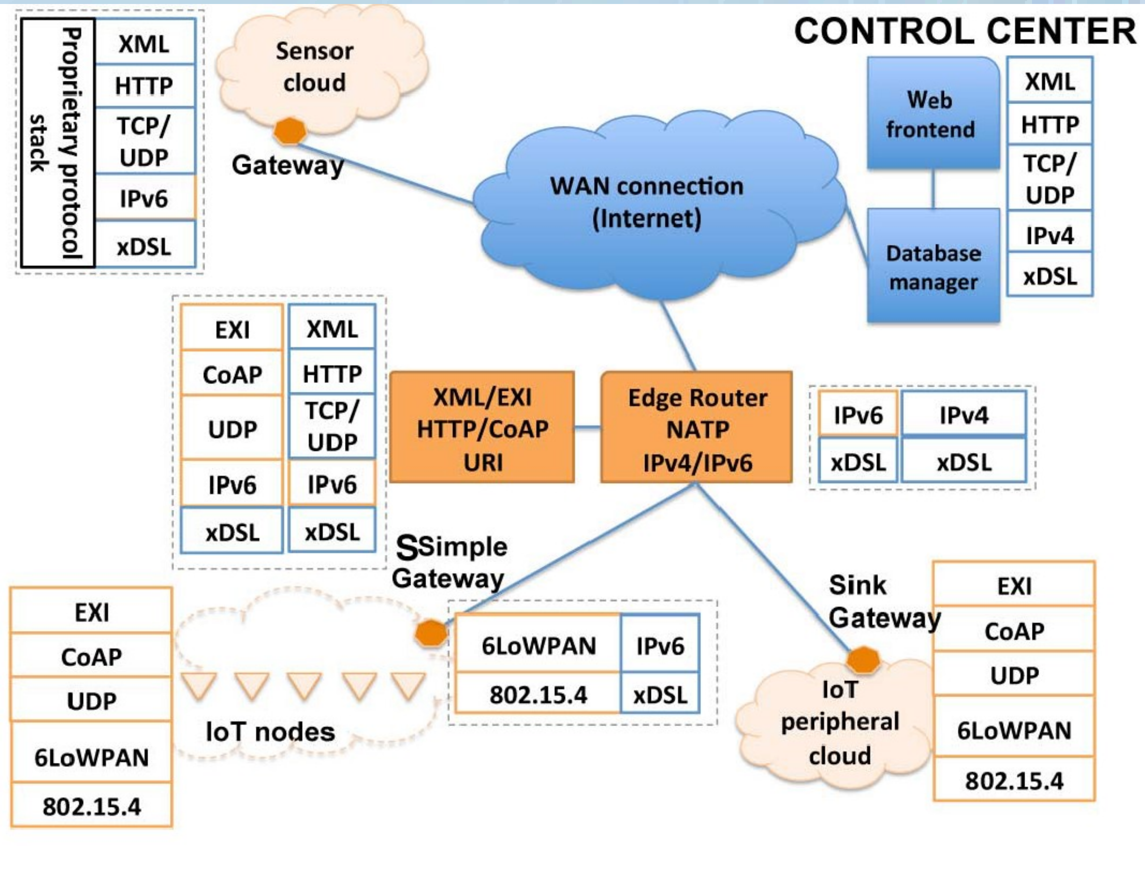


Fig. 1. Conceptual representation of an urban IoT network based on the web service approach.

URBAN IOT SYSTEM

- Reference protocol architecture for the urban IoT system
  - Unconstrained
    - de-facto standards for Internet communications
      - HTML/XML, HTTP/TCP, IPv4/IPv6
  - Constrained
    - low-complexity counterparts
      - Efficient Extensible Interchange (EXI) – transmitting of a highly compressed sequence of parse events
      - Constrained Application Protocol (CoAP) - a binary format transported over UDP
      - 6LoWPAN - compression format for IPv6 and UDP headers over low-power constrained networks

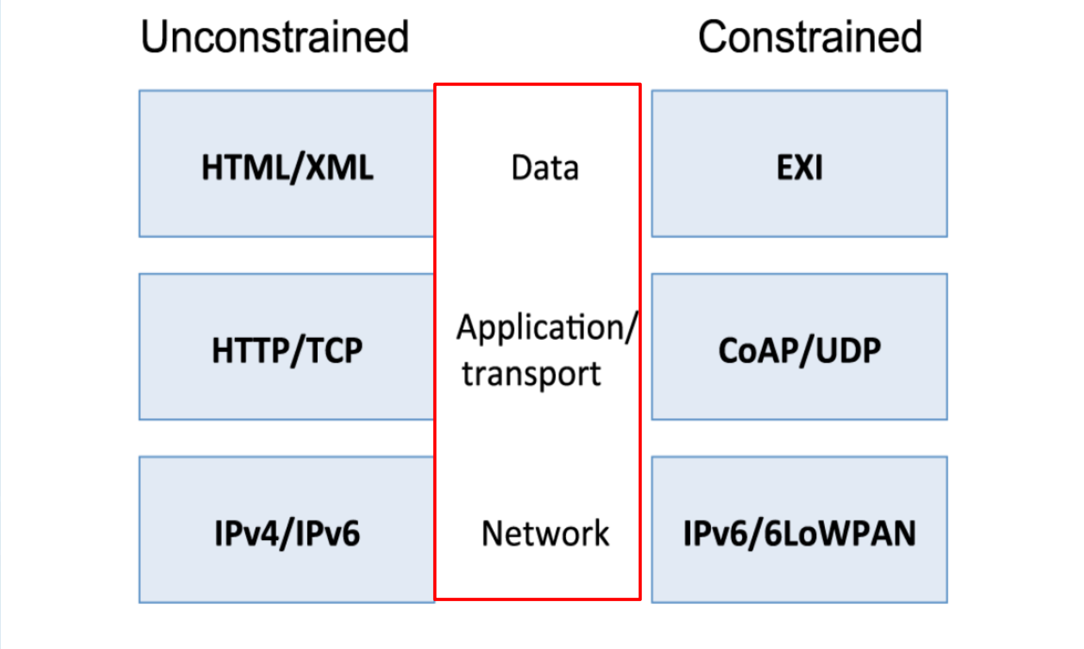


Fig. 2. Protocol stacks for unconstrained (left) and constrained (right) IoT nodes.

MIDDLEWARE FOR IOT

Proposals for IoT middleware

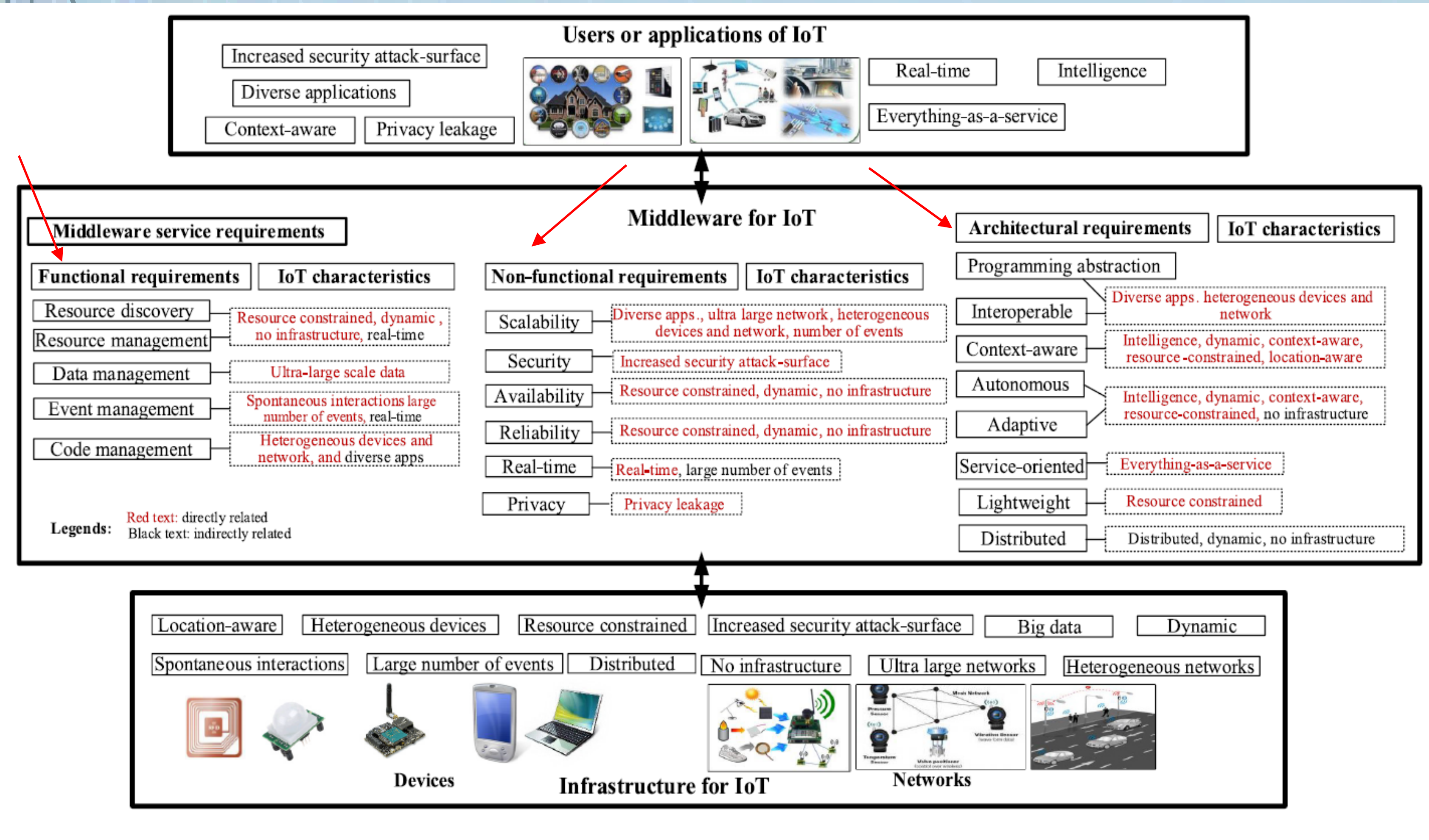


Fig. 4. Relationships between the IoT applications and infrastructure and its middleware requirements.

Huge list of middleware solutions

Functional requirements					
	Resource discovery	Resource management	Data management	Event management	Code managemen
Event-based approach					
Hermes [79]	CD-DD	RM	DPF	LS	NI
EMMA [27]	CD-DeD, CD-SD	RM	DS	SS	CA
GREEN [81]	DD-ND	RM	DS, DPF	LS	CA
RUNES [82]	CD-DeD, CD-SD	RM, RCA	DPA	SS	CA
PRISMA [29]	CD-DD	RM	DS, DPA	SS	CA
SensorBus [87]	CD-DeD	RM	NS	SS	NI
Mires [88]	CD-DeD, CD-SD	RM	DPA	SS	CA
Service-oriented approach					
Hydra [101]	DD-DeD, DD-SD	RA, RM, RCP	DS	SS	NS
Sensewrap [103]	DD-DeD, DD-SD	NI	NI	SS	NS
MUSIC [70]	DD-SD	RA, RM, RCA	NS	NS	NS
TinySOA [105]	DD-DeD, DD-SD	RA	DS	NS	NS
SOCRADES [93]	DD-DeD, DD-SD	RA, RM, RCP	NI	LS	NS
SENSEI [109]	DD-DeD	RA, RM, RCA	DS, DPA	NI	NS
ubiSOAP [94]	DD-DeD, DD-ND	RA, RM, RCA, RCL	NI	SS	NS
Servilla [95]	DD-SD	RA, RM, RCA	DPC	SS	CA, CM
KASOM [110]	DD-SD	RA, RM, RCA	NS	LS	NS
CHOReOS [112]	DD-DeD, DD-SD	RA, RM, RCA	DPA	SS	NS
MOSDEN [46]	DD-DeD, DD-SD	RA, RM, RCP	DS, DPA	NI	NS
Xively [99]	DD-DeD, DD-SD	RA, RM	DS, DPA	NI	NI
CarrIoT [98]	DD-DeD, DD-SD	RA, RM	DS, DPA	NI	NI
Echelon [118]	DD-DeD, DD-SD	RA, RM	DS, DPA	NI	NI
VM approach					
Maté [125]	DD-DeD	RA, RM	DS, DPA	LS	CA
VM* [128]	DD-DeD	RM	DS, DPA	LS	CA
Melete [130]	DD-DeD	RA, RM, RCA	DS, DPA	LS	CA, CM
MagnetOS [132]	DD-DeD	RA, RM, RCA	DS, DPA	LS	CA
Squawk [133]	DD-DeD	RA, RM	DS, DPA	NI	CA
Sensorware [129]	DD-DeD	RA, RM, RCA	DS, DPA	LS	CA, CM
Extended Maté [137]	DD-DeD	RA, RM	DS, DPA	LS	CA
DVM [138]	DD-DeD	RA, RM, RCL	DS, DPA	SS	CA
DAViM [139]	CD-DD	RA, RM	DS, DPA	SS	CA
SwissQM [140]	DD-DeD, DD-SD	RA, RM, RCA	DS, DPA, DPF	LS	CA
TinyVM [141]	NI	NI	DPA	NI	NI
TinyReef [123]	NI	NS	DS, DPA	SS	CA
Agent-based approach					
Impala [149]	DD-DeD	RA, RM	DPA	LS	CA, CM
Smart messages [150]	DD-ND	RA, RM	DPA	SS	CA, CM
ActorNet [151]	DD-DeD	RA	DPA	SS	CA, CM
Agilla [28]	DD-DeD	RA, RM, RCA	DPA	LS	CA, CM
Ubiware [152]	DD-DeD, DD-SD	RA, RM, RCA	DPA	LS	CA, CM
UbiROAD [153]	CD-SD	RM	DS	LS	CA, CM
AFME [154]	CD-DD	RA	DPA	SS	CM
MAPS [155]	DD-DeD	RA, RM	DPA	LS	CA, CM
MASPOT [147]	DD-DeD	RA, RM, RCA	DPA	LS	CA, CM
TinyMAPS [156]	DD-DeD	RCA	DPA, DPF	LS	CA, CM
Tuple-space approach					
LIME [160]	CD-DeD, CD-SD	RM	DS	SS	CM
TeenyLIME [162]	CD-DeD, CD-SD	RM	DS, DPA	SS	CM
TinyLIME [161]	CD-DeD, CD-SD	RM	DPA	SS	CM
TS-Mid [164]	CD-DeD, CD-SD	RM	DS, DPA	SS	NS
A3-TAG [165]	CD-DeD, CD-SD	RM	DS, DPA	SS	NS
Database approach					
SINA [166]	DD-DeD	RM	DS, DPA	SS	NS
COUGAR [168]	DD-ND	RM	DS, DPA	LS	NS
IrisNet [169]	DD-SD	RA, RM, RCL	DS, DPA, DPF	SS	CA, CM
Sensation [170]	CD-DeD	RM	DS, DPA	SS	NS
TinyDB [69], [171]	DD-DeD	RM	DS, DPA, DPF	SS	NI
GSN [172]	DD-Ded, DD-SD	RA, RM	DS, DPF	LS	CA
KSpot+ [173]	DD-SD	RM	DS, DPA	NS	NS
HyCache [174]	DD-DeD	RM	DS, DPA	NS	NS
Application-specific approach					
AutoSec [175]	CD-SD	RA, RM, RCA, RCL	DS, DPA, DPF	LS	CA, CM
Adaptive middleware [176]	CD-SD	RA, RM	DS, DPA	LS	CA
MiLAN [177]	CD-SD	RA, RM, RCA	DS, DPA	LS	CA, CM
TinyCubus [178]	CD-SD	RA, RM	DS, DPA	LS	CA
MidFusion [179]	CD-SD	RA, RM, RCA	DS, DPA, DPC, DPF	LS	CA
Legend Not supported (NS) No information (NI)	Centralised discovery (CD) Distributed discovery (DD) Device discovery (DeD) Network discovery (ND) Service discovery (SD)	Resource allocation (RA) Resource monitor (RM) Resource composition (RC) - Adaptive (A) - Predefined (P) Resource conflict (RCL)	Data storage (DS) Data preprocessing (DP) - Aggregation (A) - Compression(C) - Filtering (F)	Supported - Large scale (LS) - Small scale (SS)	Code allocation (CA) Code migration (CM)

Middleware for Internet of Things: A Survey, IEEE INTERNET OF THINGS JOURNAL, VOL.

MIDDLEWARE FOR IOT

- Existing middleware solutions (based on their design approaches)
  - Event-based
  - Service-oriented
  - VM-based
  - Agent-based
  - Tuple-spaces
  - Database-oriented
  - Application-specific

Middleware for Internet of Things: A Survey, IEEE INTERNET OF THINGS JOURNAL, VOL.

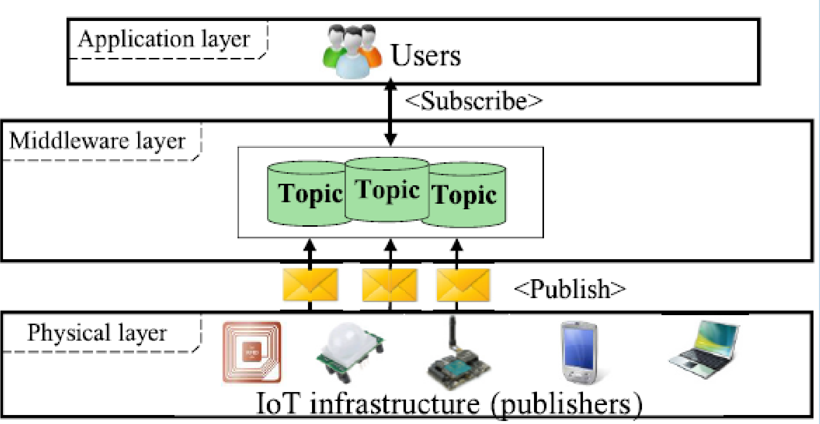


Fig. 5. General design model for an event-based middleware.

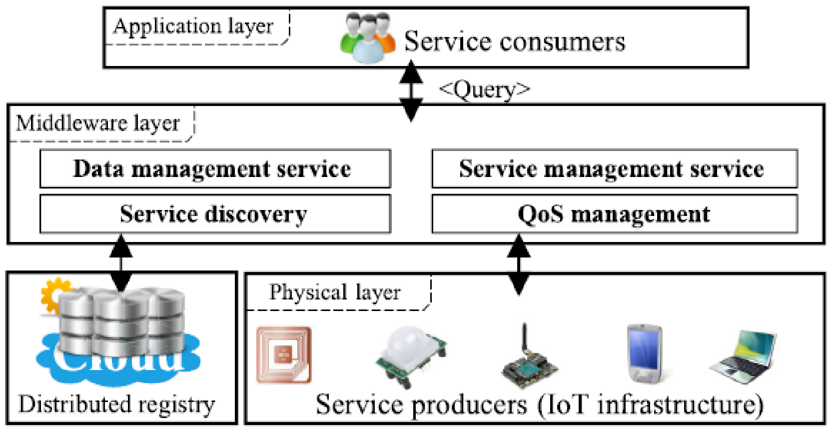


Fig. 6. General design model for an SOM.

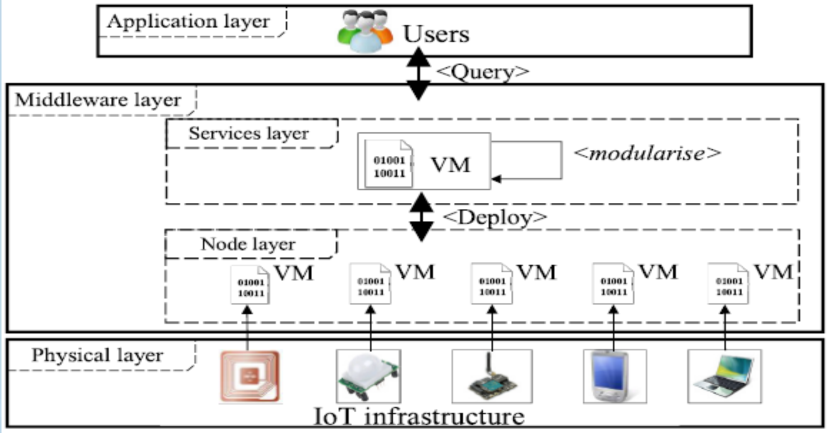


Fig. 7. General design model for a VM-based middleware.

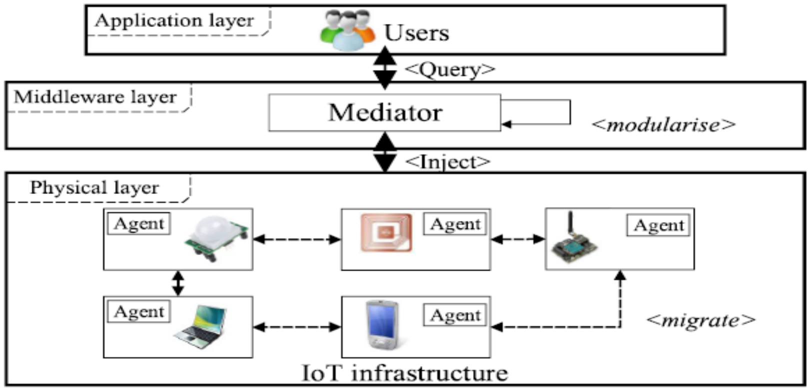


Fig. 8. General design model for an agent-based middleware.

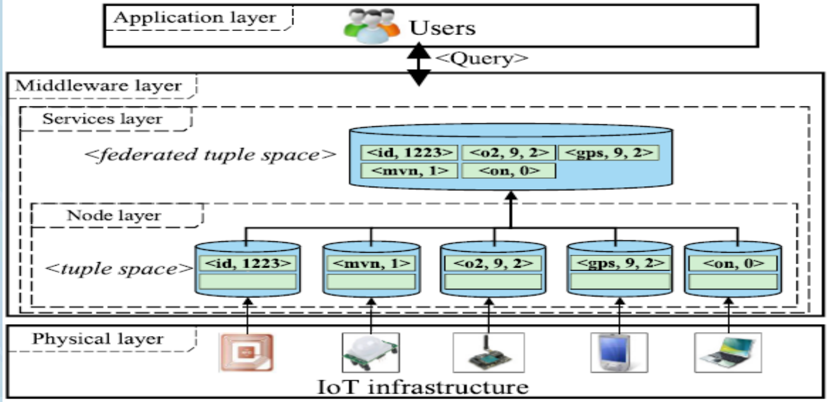


Fig. 9. General design model for a tuple-space middleware.

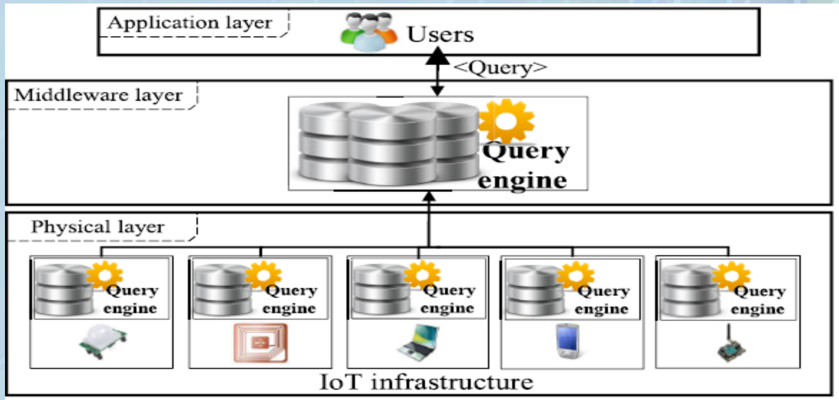


Fig. 10. General design model for a database-oriented middleware.

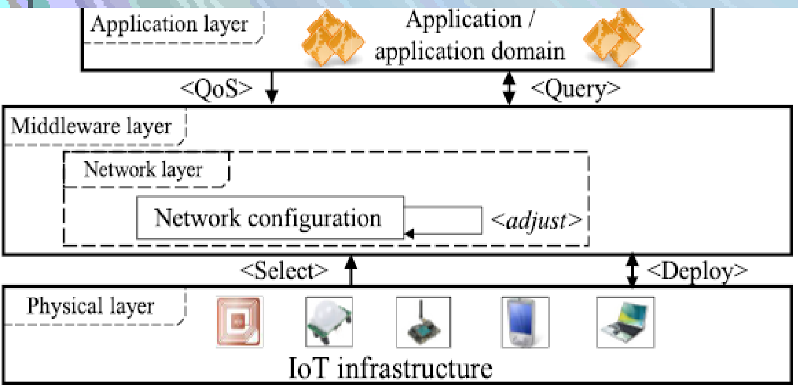


Fig. 11. General design model for an application-specific middleware.

MIDDLEWARES FOR IOT

ADVANCED NETWORKING TECHNOLOGIES FOR SMART CITIES

NETWORK VIRTUALIZATION, SOFTWARE-DEFINED NETWORKING, NETWORK FUNCTIONS VIRTUALIZATION, SOFTWARE-DEFINED NETWORK FUNCTION VIRTUALIZATION

STATUS OF CURRENT NETWORKS

- Computer networks can be divided in three planes of functionality:
  - The data plane: responsible for forwarding data
  - The control plane: routing protocols to define the forwarding tables
  - The management plane: remotely monitor and configure the control functionality
- IP networks are complex and hard to manage
  - To express high-level policies, operators need to configure individual network elements separately
    - low-level and/or vendor-specific commands
  - Automatic reconfiguration and response mechanisms are virtually non-existent

STATUS OF CURRENT NETWORKS

- Current networks are vertically integrated
  - Two abstract elements:
    - Control plane (that decides **HOW** to handle network traffic)
    - Data plane (that **FORWARD** traffic according to the control plane policies)
  - They are integrated inside the networking elements
    - Lack of flexibility of the networking infrastructure.
  - A new protocol can take 5-10 years to be fully designed, evaluated and deployed
    - IPv4 to IPv6 (still at 15% adoption)
  - A clean-slate approach is not feasible

STATUS OF CURRENT NETWORKS

- The Internet is a very complex and relatively static architecture
- Network misconfigurations and related errors are extremely common in today’s networks
- Network management: proprietary solutions of specialized hardware, operating systems, and network applications
- A myriad of specialized components and middleboxes to configure, deploy, and manage
  - E.g., firewalls, IDS, and DPI engines

ADVANCED NETWORKING TECHNOLOGIES FOR SMART CITIES

NETWORK VIRTUALIZATION, SOFTWARE-DEFINED NETWORKING, NETWORK FUNCTIONS VIRTUALIZATION, SOFTWARE-DEFINED NETWORK FUNCTION VIRTUALIZATION

THE BEGINNING: NETWORK VIRTUALIZATION

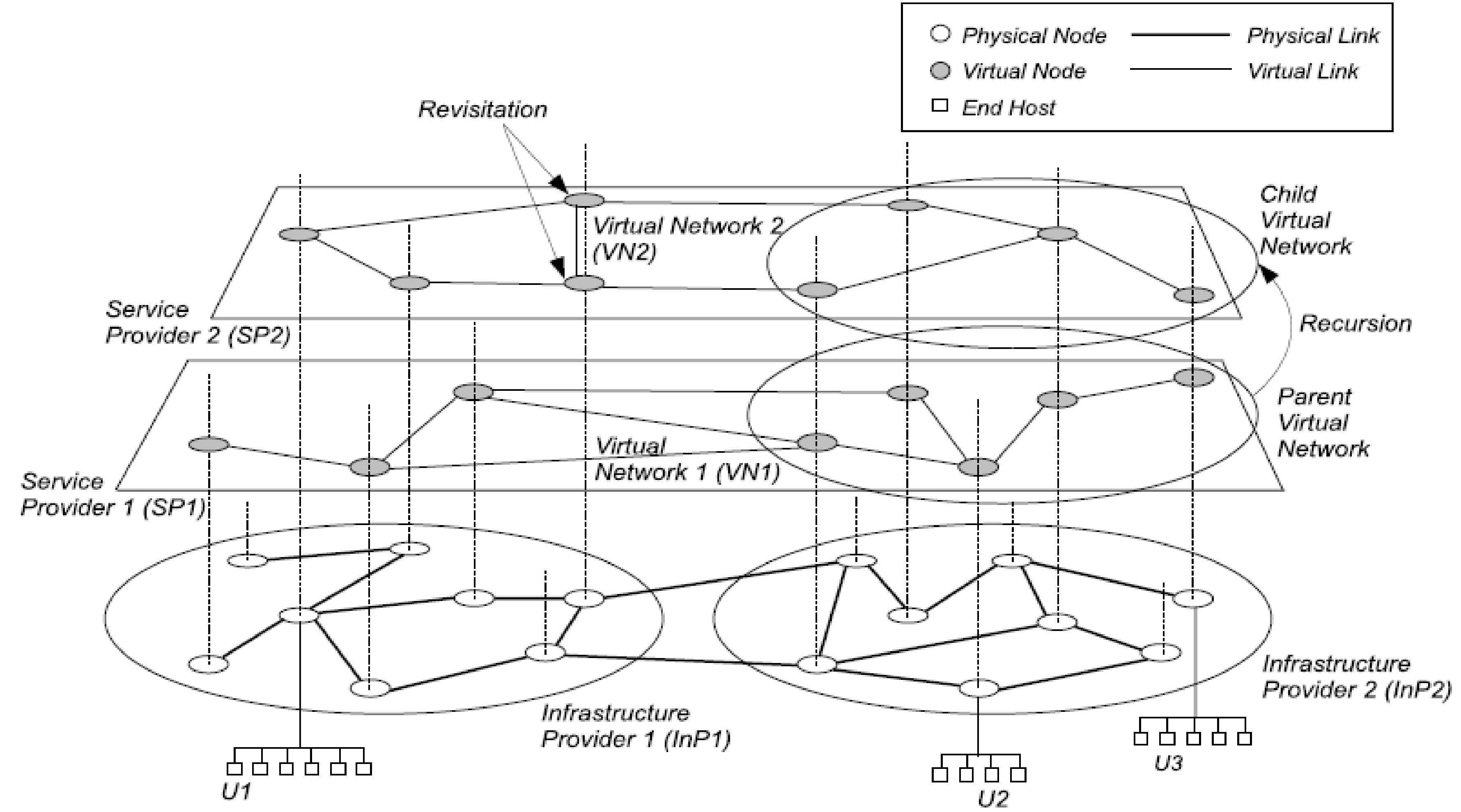


Figure 2: Network Virtualization Architecture

SOFTWARE-DEFINED NETWORKING

- An emerging networking paradigm
- it breaks the vertical integration
  - Separates the network's control logic (the control plane) from the underlying routers and switches that forward the traffic (the data plane).
  - Network switches become simple forwarding devices
- The control logic is implemented in a logically centralized controller (or Network Operating System)

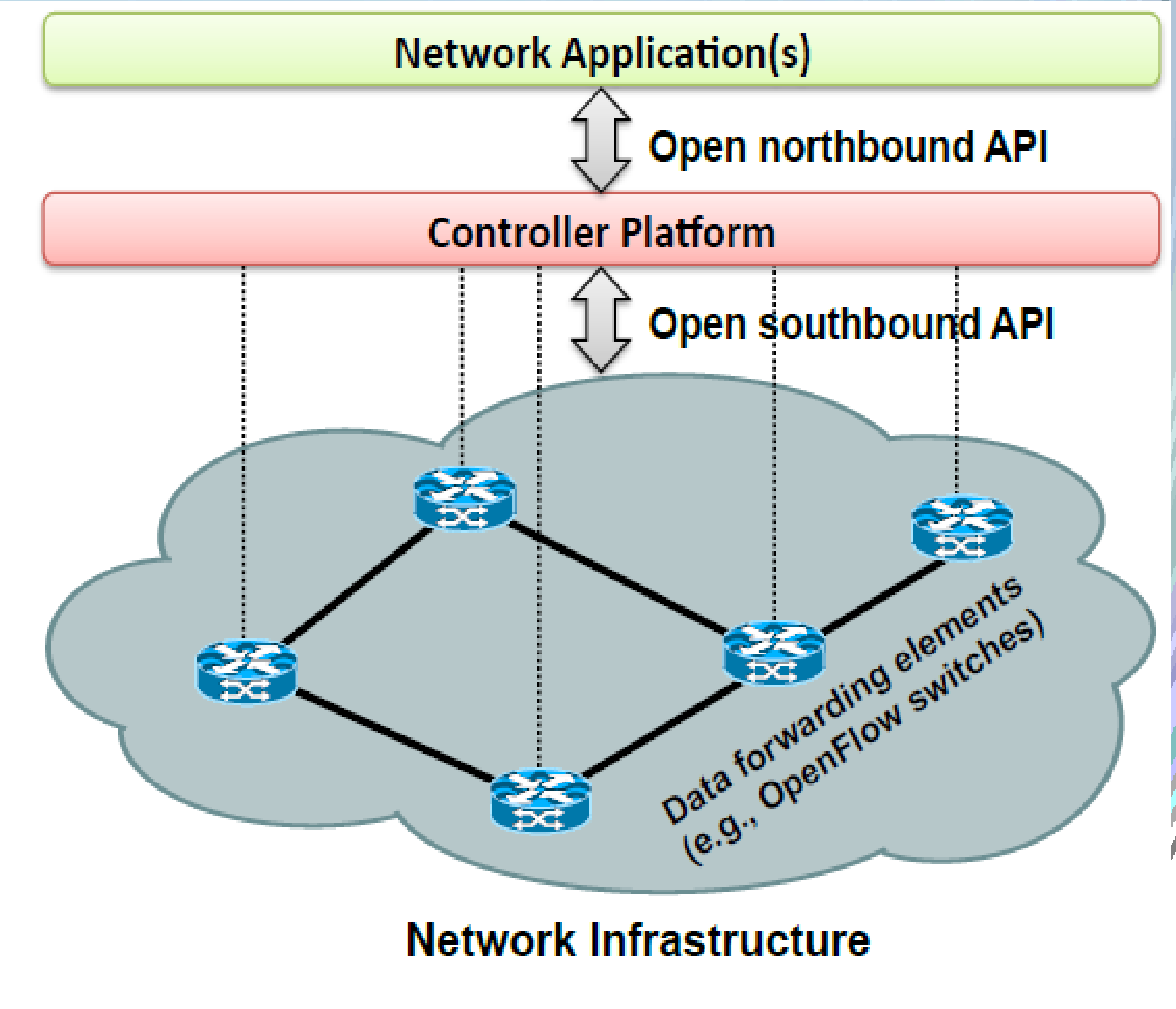


Fig. 1. Simplified view of an SDN architecture.

SOFTWARE-DEFINED NETWORKING

- Simplifies policy enforcement and network (re)configuration and evolution
- Logically centralized programmatic model does not mean a physically centralized system
  - For adequate levels of performance, scalability and reliability,
    - production-level SDN network designs resort to physically distributed control planes

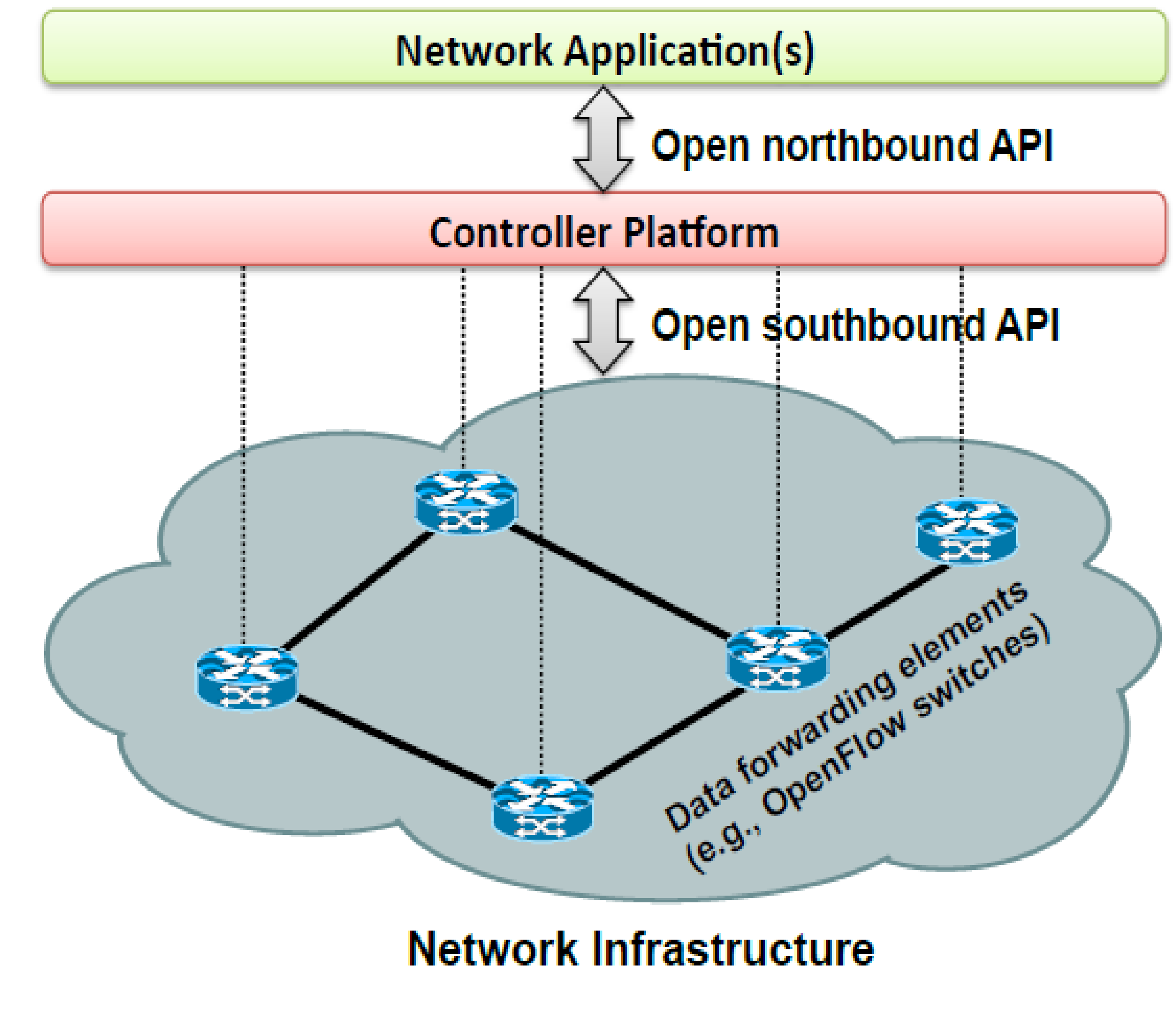


Fig. 1. Simplified view of an SDN architecture.

SOFTWARE-DEFINED NETWORKING

- In a nutshell (the four pillars):
  1. control and data planes are decoupled
  2. Forwarding decisions are flow-based, instead of destination-based
  3. Control logic is moved to an external entity
     - SDN controller or NOS
  4. The network is programmable

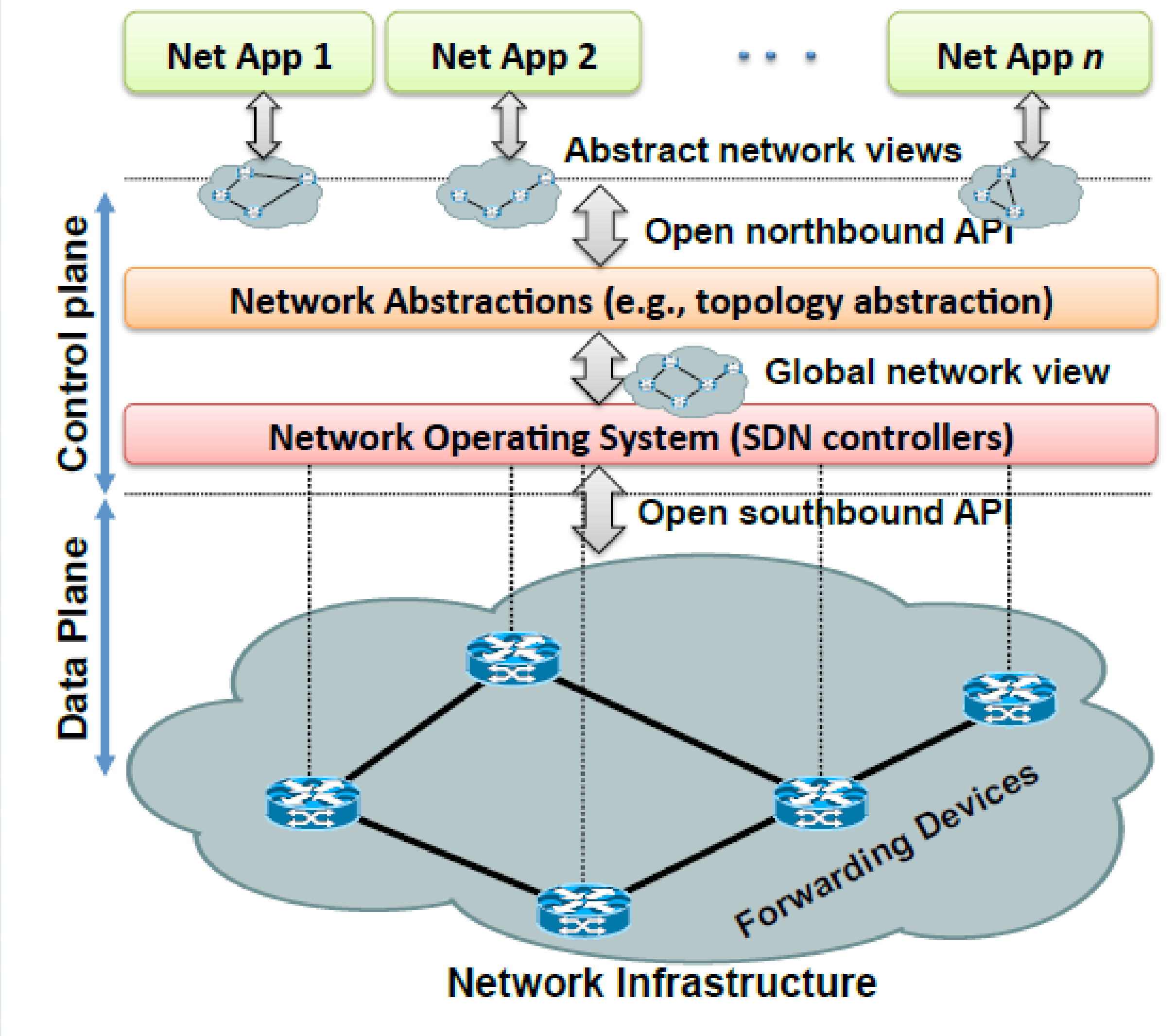


Fig. 4. SDN architecture and its fundamental abstractions.

TRADITIONAL NETWORKING VS SDN

- Terminology:
  - Forwarding Devices (FD):
    - Hardware- or software-based devices
    - Implements southbound protocols:
      - OpenFlow, ForCES, POF, P4
  - Data Plane (DP): interconnected FDs
  - Control Plane (CP): control logic rests in the applications and controllers
  - Southbound Interface (SI): defines the communication protocol between FD and CP
  - Northbound Interface (NI): API to APP developers
  - Management Plane (MP): applications that implement network control and operation logic
    - Routing, firewalls, load balancers, monitoring

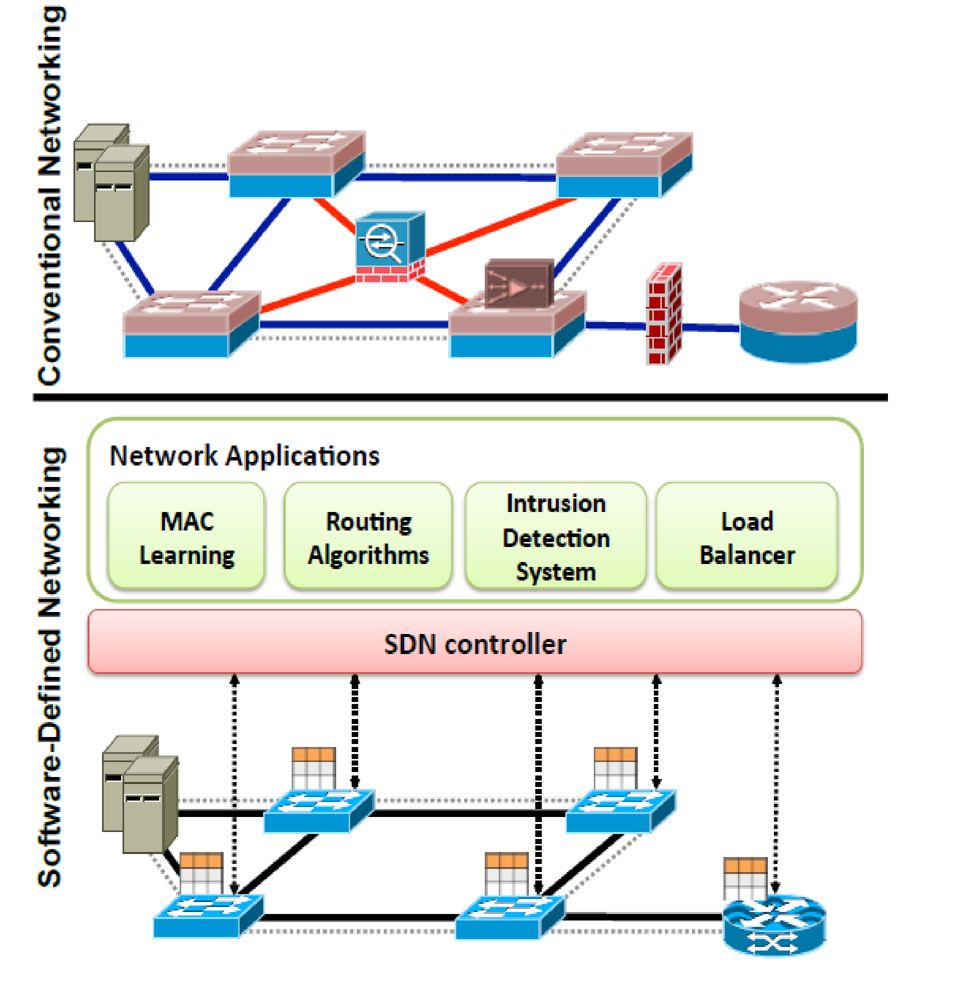


Fig. 5. Traditional networking versus Software-Defined Networking (SDN). With SDN, management becomes simpler and middleboxes services can be delivered as SDN controller applications.

A CLOSER LOOK AT SDN

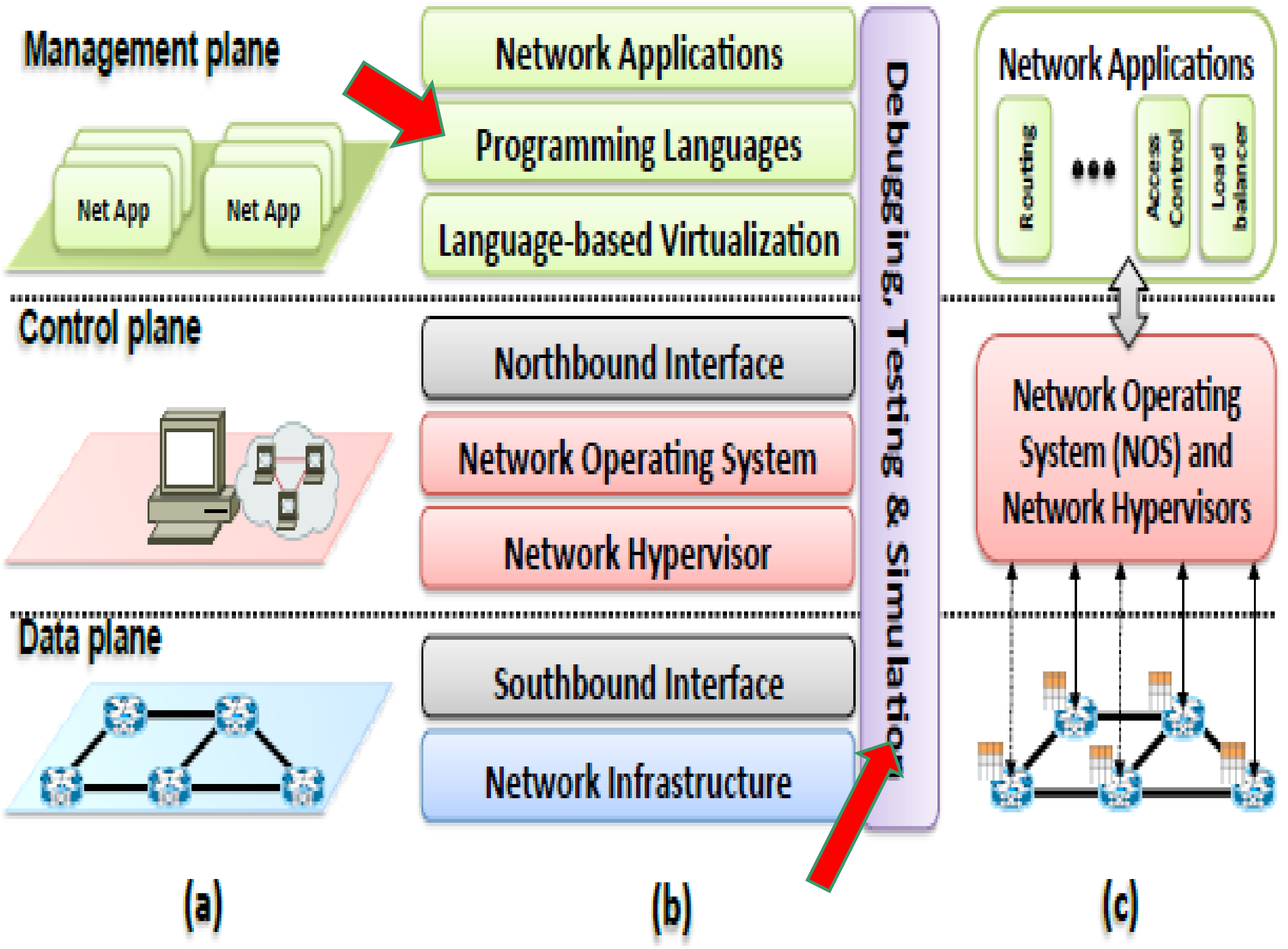


Fig. 6. Software-Defined Networks in (a) planes, (b) layers, and (c) system design architecture

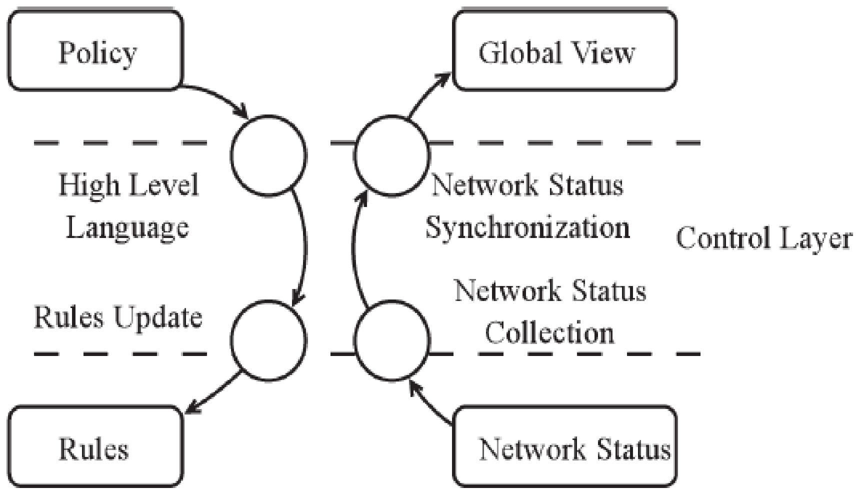


Fig. 4. Controller Logical Design: a high level language for SDN applications to define their network operation policies; a rule update process to install rules generated from those policies; a network status collection process to gather network infrastructure information; a network status synchronization process to build a global network view using network status collected by each individual controller.

A CLOSER LOOK AT NETWORK VIRTUALIZATION AND SDN

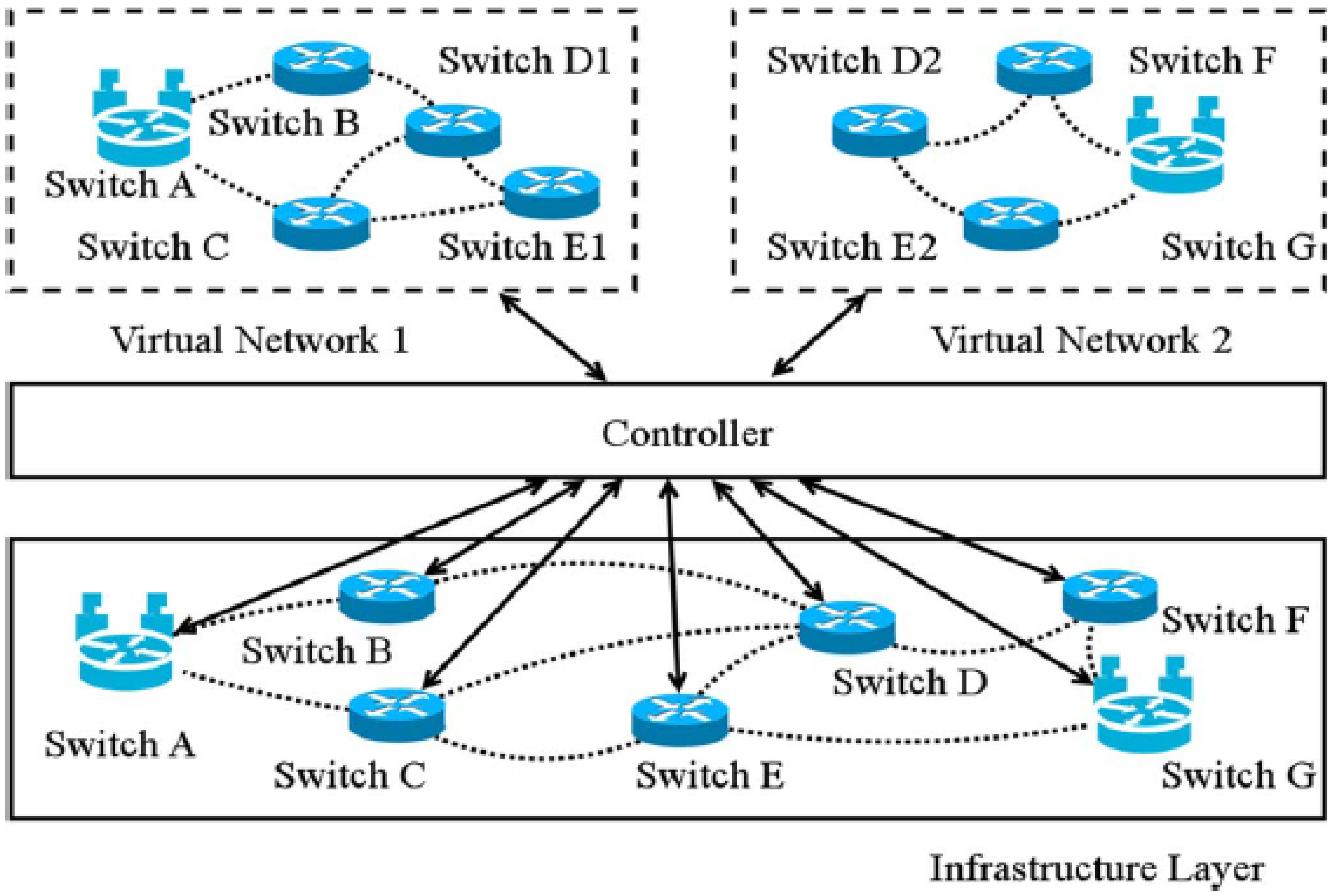


Fig. 5. Network virtualization: Multiple virtual networks can be created on the same physical network, sharing infrastructure resources. An SDN application can only oversee and user resources of its own virtual network.

A CLOSER LOOK AT OPENFLOW

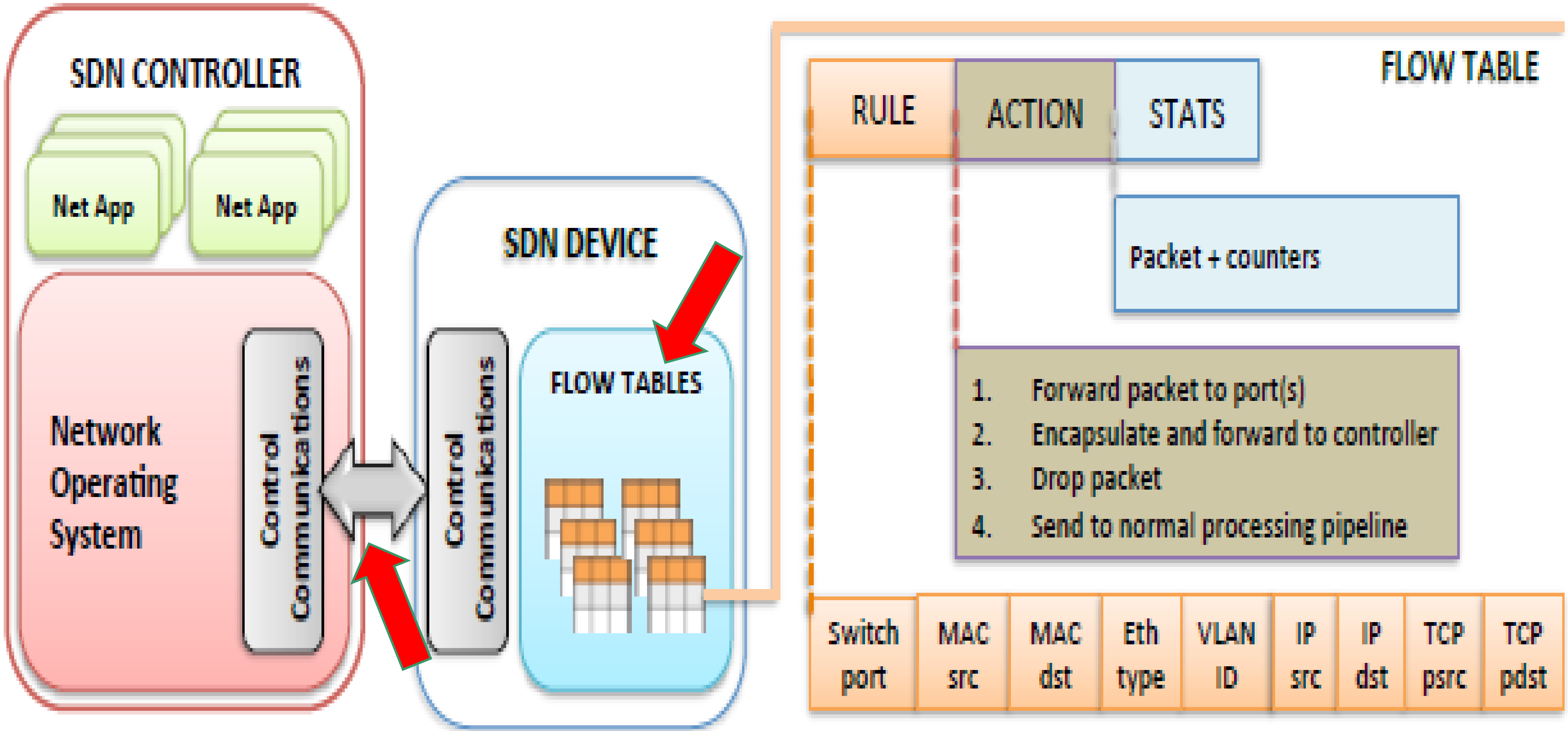


Fig. 7. OpenFlow-enabled SDN devices

A CLOSER LOOK AT SDN CONTROLLERS

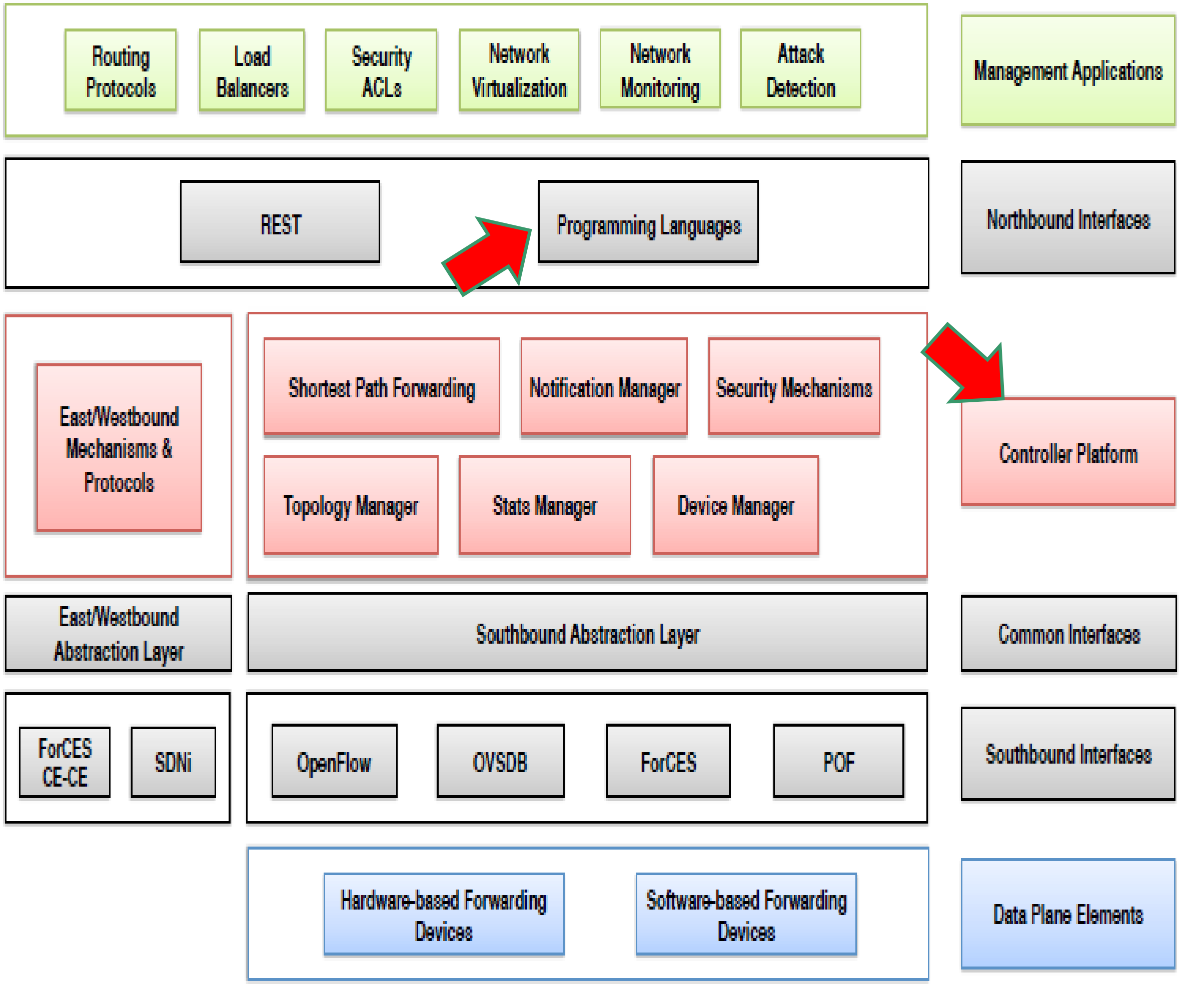


Fig. 8. SDN control platforms: elements, services and interfaces

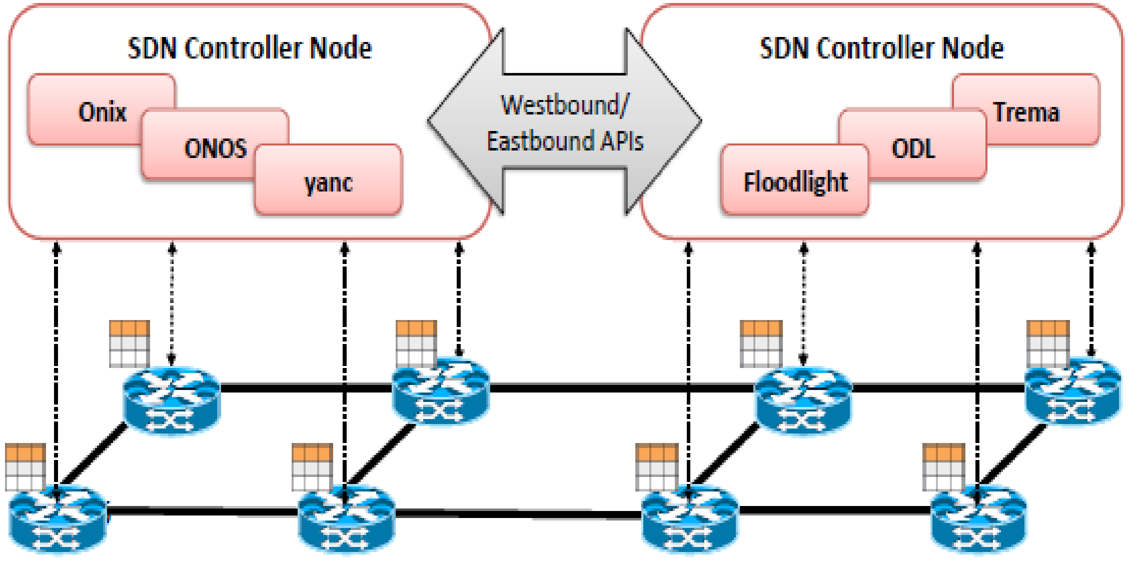


Fig. 9. Distributed controllers: east/westbound APIs.

A CLOSER LOOK AT SDN PROGRAMMING LANGUAGES

TABLE III
CURRENT CONTROLLER IMPLEMENTATIONS COMPLIANT WITH THE OPENFLOW STANDARD.

Controller	Implementation	Open Source	Developer	Overview
POX [59]	Python	Yes	Nicira	General, open-source SDN controller written in Python.
NOX [17]	Python/C++	Yes	Nicira	The first OpenFlow controller written in Python and C++.
MUL [60]	C	Yes	Kulcloud	OpenFlow controller that has a C-based multi-threaded infrastructure at its core. It supports a multi-level north-bound interface (see Section III-E) for application development.
Maestro [21]	Java	Yes	Rice University	A network operating system based on Java; it provides interfaces for implementing modular network control applications and for them to access and modify network state.
Trema [61]	Ruby/C	Yes	NEC	A framework for developing OpenFlow controllers written in Ruby and C.
Beacon [22]	Java	Yes	Stanford	A cross-platform, modular, Java-based OpenFlow controller that supports event-based and threaded operations.
Jaxon [62]	Java	Yes	Independent Developers	a Java-based OpenFlow controller based on NOX.
Helios [24]	C	No	NEC	An extensible C-based OpenFlow controller that provides a programmatic shell for performing integrated experiments.
Floodlight [38]	Java	Yes	BigSwitch	A Java-based OpenFlow controller (supports v1.3), based on the Beacon implementation, that works with physical- and virtual- OpenFlow switches.
SNAC [23]	C++	No	Nicira	An OpenFlow controller based on NOX-0.4, which uses a web-based, user-friendly policy manager to manage the network, configure devices, and monitor events.
Ryu [63]	Python	Yes	NTT, OSRG group	An SDN operating system that aims to provide logically centralized control and APIs to create new network management and control applications. Ryu fully supports OpenFlow v1.0, v1.2, v1.3, and the Nicira Extensions.
NodeFlow [64]	JavaScript	Yes	Independent Developers	An OpenFlow controller written in JavaScript for Node.JS [65].
ovs-controller [55]	C	Yes	Independent Developers	A simple OpenFlow controller reference implementation with Open vSwitch for managing any number of remote switches through the OpenFlow protocol; as a result the switches function as L2 MAC-learning switches or hubs.
Flowvisor [48]	C	Yes	Stanford/Nicira	Special purpose controller implementation.
RouteFlow [66]	C++	Yes	CPqD	Special purpose controller implementation.

A CLOSER LOOK AT SDN PROGRAMMING LANGUAGES

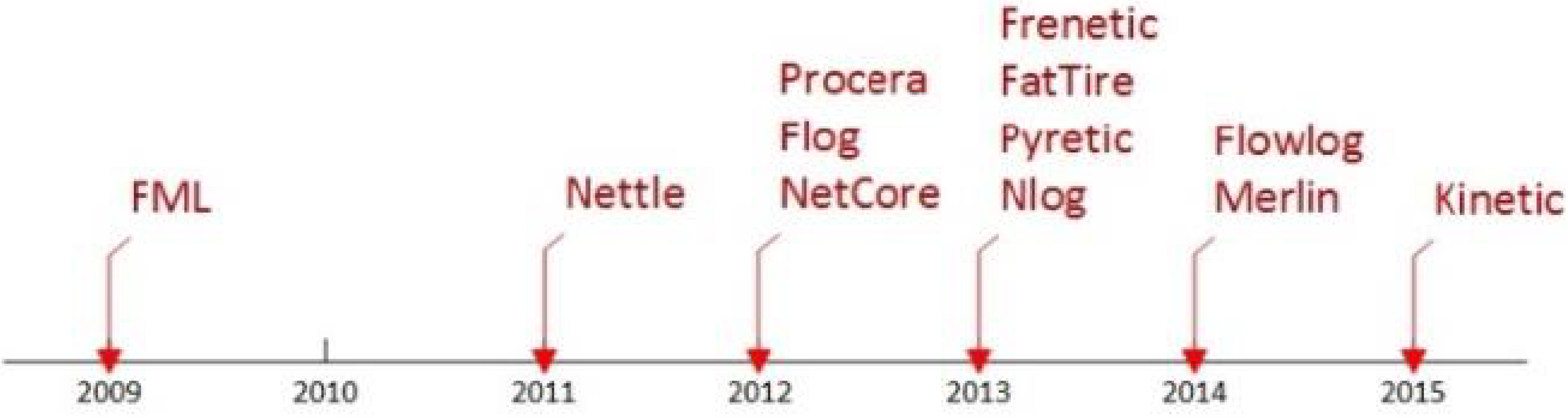


Figure 6. The SDN programming languages timeline.

ADVANCED NETWORKING TECHNOLOGIES FOR SMART CITIES

NETWORK VIRTUALIZATION, SOFTWARE-DEFINED NETWORKING, **NETWORK FUNCTIONS VIRTUALIZATION, SOFTWARE-DEFINED NETWORK FUNCTION VIRTUALIZATION**

NETWORK FUNCTIONS VIRTUALIZATION (NFV)

- Current network services rely on proprietary appliances
- Service provision is deployed at physical proprietary devices
  - **One equipment for each function or a few functions**
    - **E.g.: Firewall, NAT, DPI**
- Service Providers must continuously purchase, store, and operate new physical equipment
  - High CAPEX and OPEX

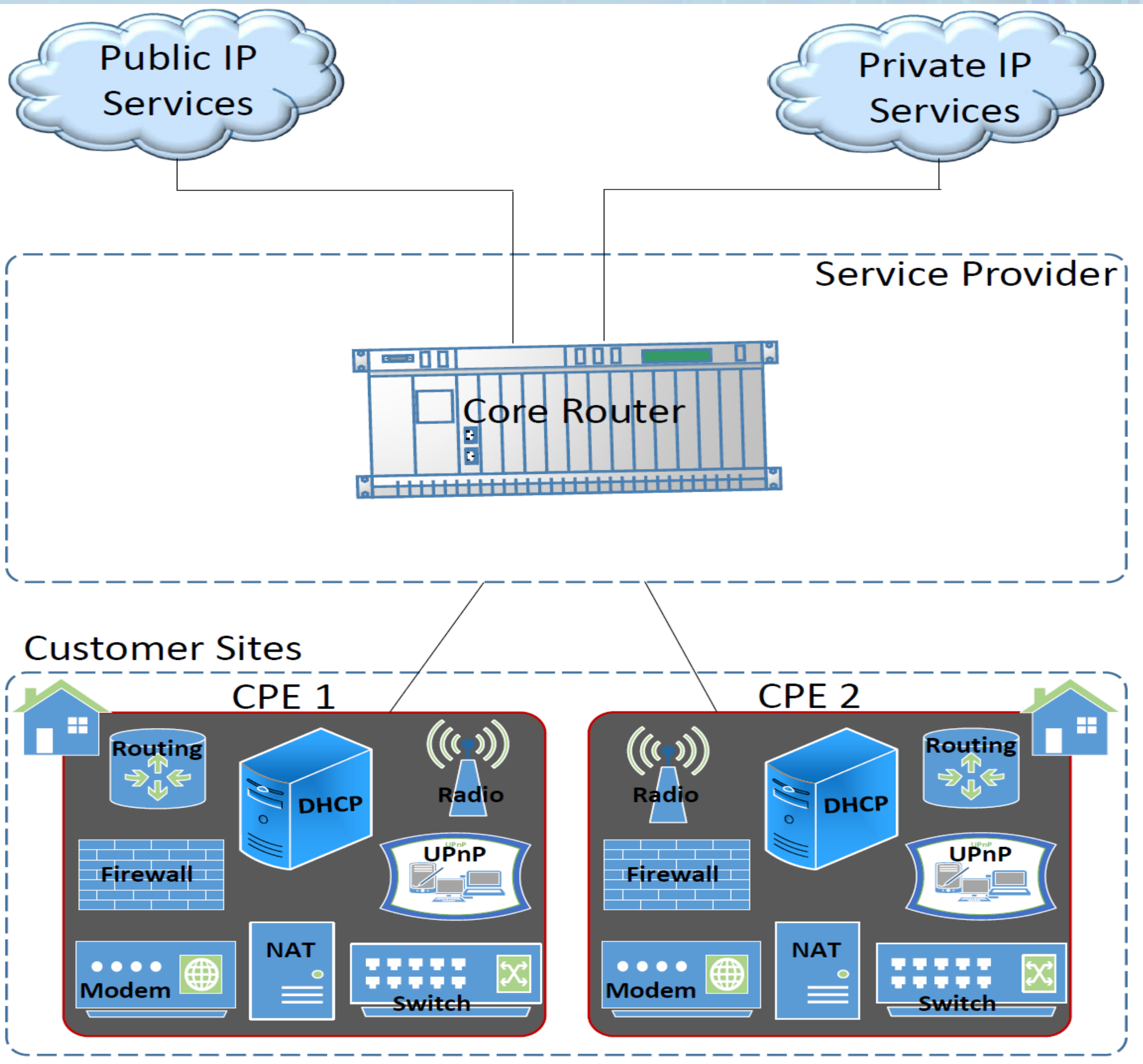


Fig. 1. Traditional CPE Implementations

Source: Network Function Virtualization: State-of-the-

NETWORK FUNCTION VIRTUALIZATION

- NFV: address these challenges by leveraging virtualization technology
  - new way to design, deploy and manage networking services
- Decouples physical network equipment from the functions that run on them
- A network function can be deployed as an instance of software
  - consolidation of many equipment types
  - Located in data centers, distributed network nodes and at end user premises

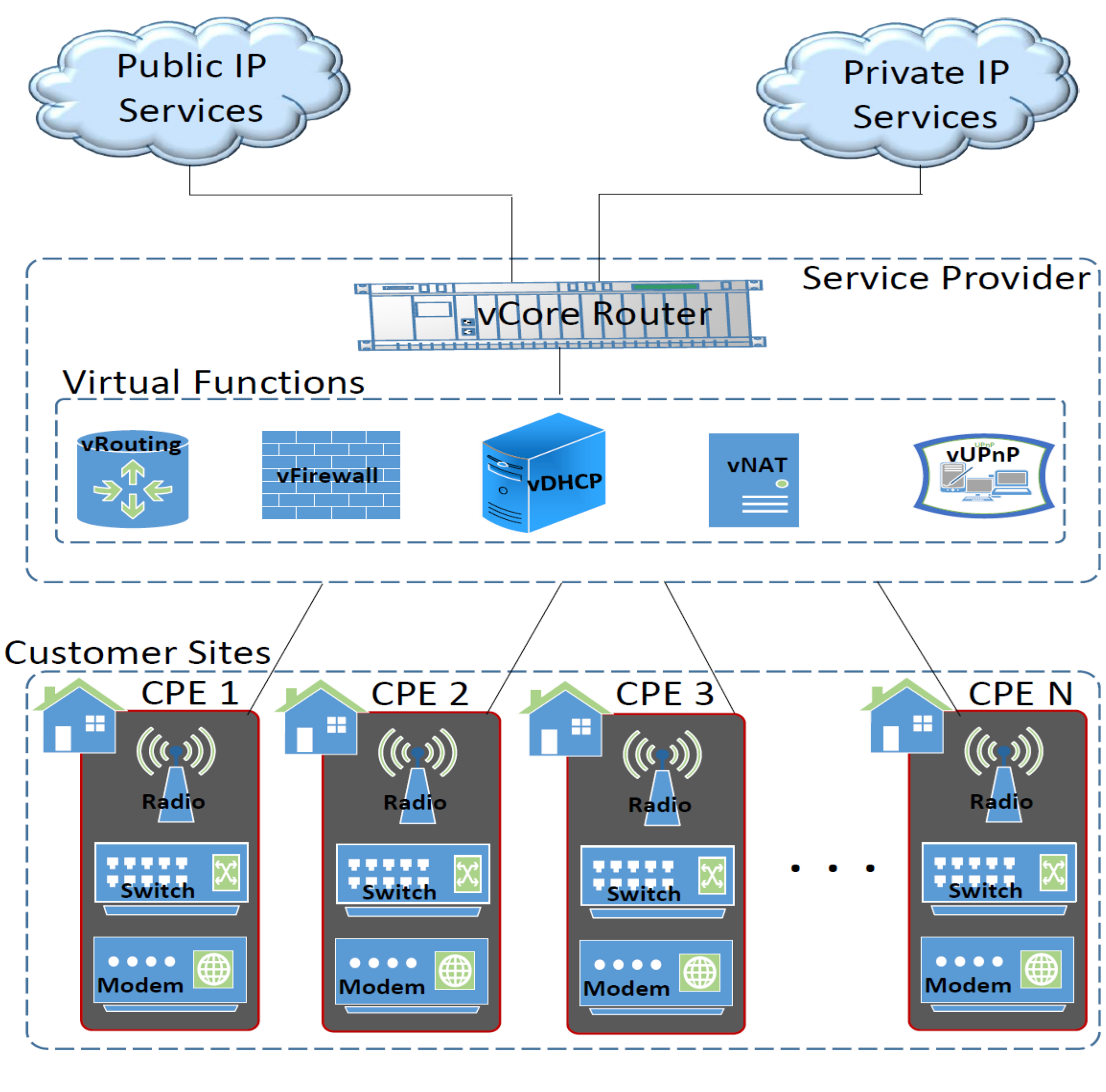


Fig. 2. Possible CPE Implementation with NFV

Source: Network Function Virtualization: State-of-the

NETWORK FUNCTION VIRTUALIZATION

- In a nutshell (the 3 key elements):
  1. NFV Infrastructure (NFVI)
     - combination of COTS HW and SW
  2. Virtual Network Functions / Services
     - NF is a functional block within a NFVI
       - It has well-defined interfaces and functional behavior
  3. NFV Management and Orchestration (NFV MANO)
     - Provisioning and configuration of VNFs

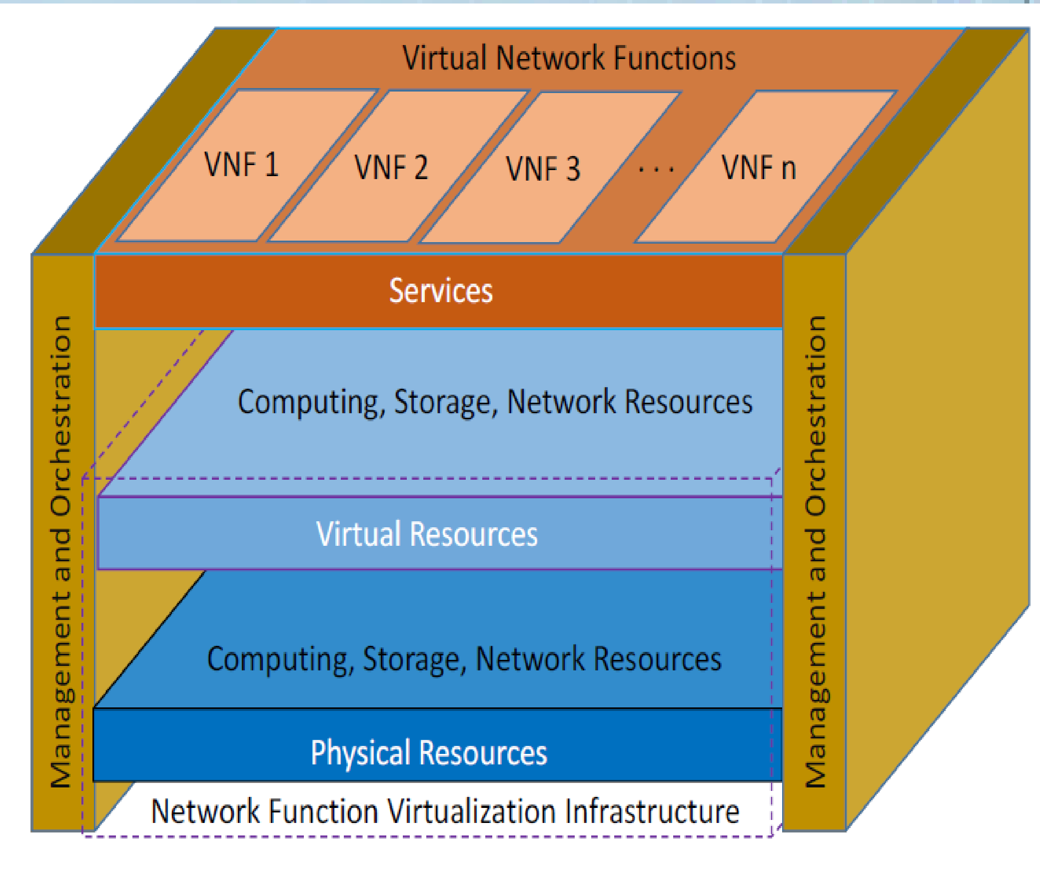


Fig. 4. Network Function Virtualization Architecture

Source: Network Function Virtualization: State-of-the-

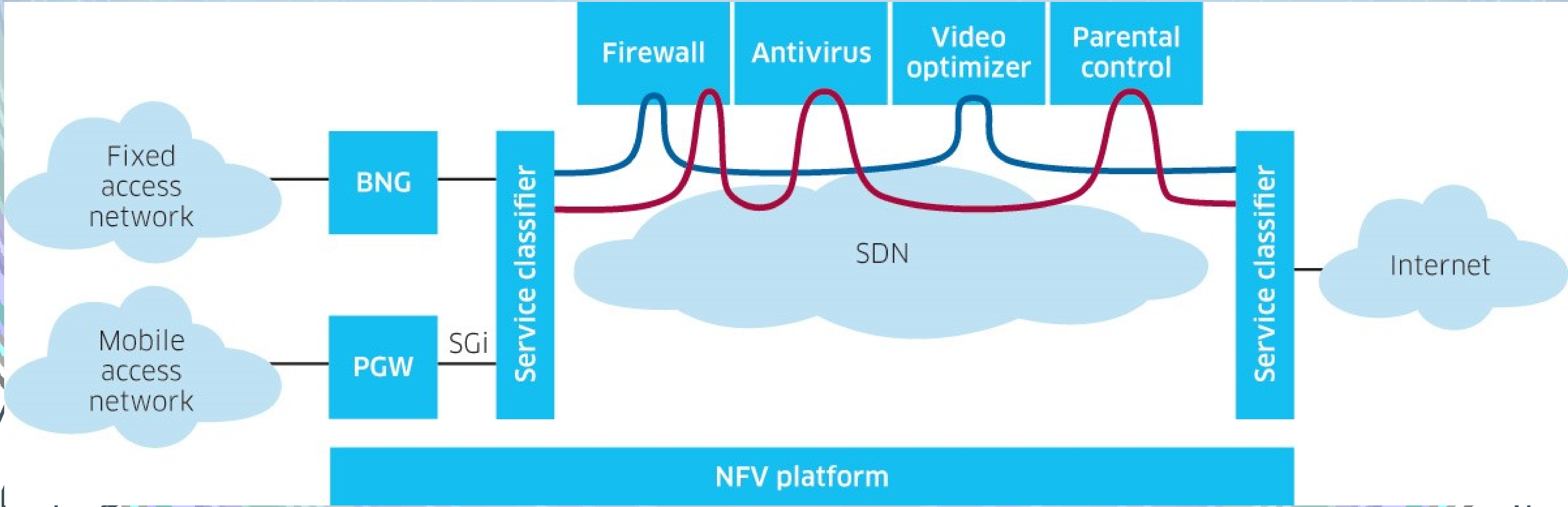
COMMON NETWORK FUNCTIONS

- Broadband Network Gateway / Carrier grade NAT
- Broadband remote access server (BRAS) and Routers
- Home Location Register/ Home Subscriber Server (HLR/HSS)
- Serving GPRS Support Node Mobility Management Entity (SGSNMME)
- Gateway Support Node / Packet Data Network Gateway (GGSN/PDN-GW),
- RNC / NodeB / and Evolved Node B (eNodeB).
- IPSec / SSL VPN gateways
- Deep Packet Inspection (DPI) / Firewall / NAT
- Service Assurance / Service Level Agreement (SLA) monitoring, Test and Diagnostics.
- IP Multimedia Sub-system (IMS)
- Video Optimizers / Transcoding

Source: Software-Defined Network Function

SERVICE FUNCTION CHAINING

- SFC (RCF 7498): definition and instantiation of an ordered list of service functions
  - "steering" of traffic flows through the SFs



Source: Problem Statement for Ser

SDN VS NFV

- Concepts
  - NFV implements network functions in software
  - SDN provides better network control through centralized and programmable network architecture
- Goals
  - NFV aims at reducing CapEx, OpEx, and space and power consumption
  - SDN aims at providing network abstractions to enable exible network control, conguration and fast innovation
- Approach
  - NFV decouples the network functions from the proprietary hardware to achieve agile provisioning and deployment
  - SDN decouples the network control plane from the data plane forwarding to provide a centralized controller via enabling programmability

Source: Software-Defined Networking

SOFTWARE-DEFINED NETWORK FUNCTION VIRTUALIZATION (SDNFV)

- integrating SDN with NFV aims at achieving various network control and management goals
  - Dynamic resource management
  - Virtual service environment for SFCs
  - Enables real-time and dynamic function provisioning
- NFV serves SDN by virtualizing the SDN controller
- serves NFV by providing programmable connectivity between VNFs

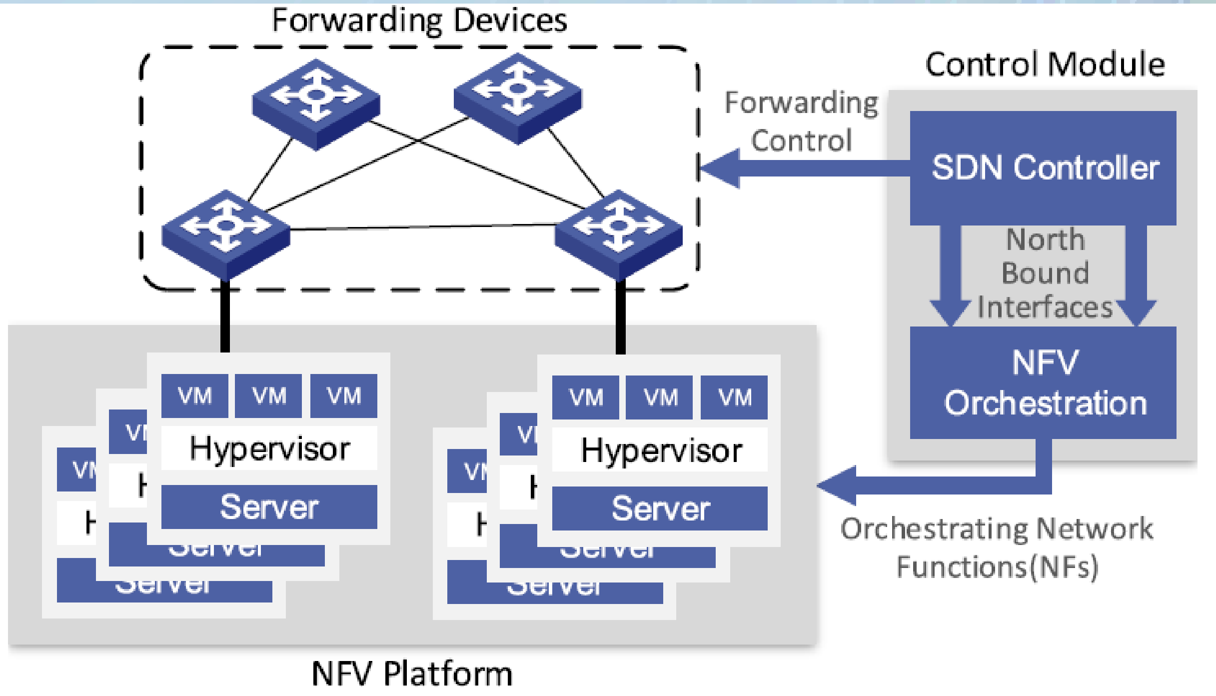


FIGURE 3. Software-defined NFV system.

Source: Network Function Virtualization: State-of-the-

SFC IN SDNFV

- NFV moves network functions to software on a general HW platform
- SDN moves control functions to the software controller
- Service deployment and chains can be provided and configured in the SDN controller

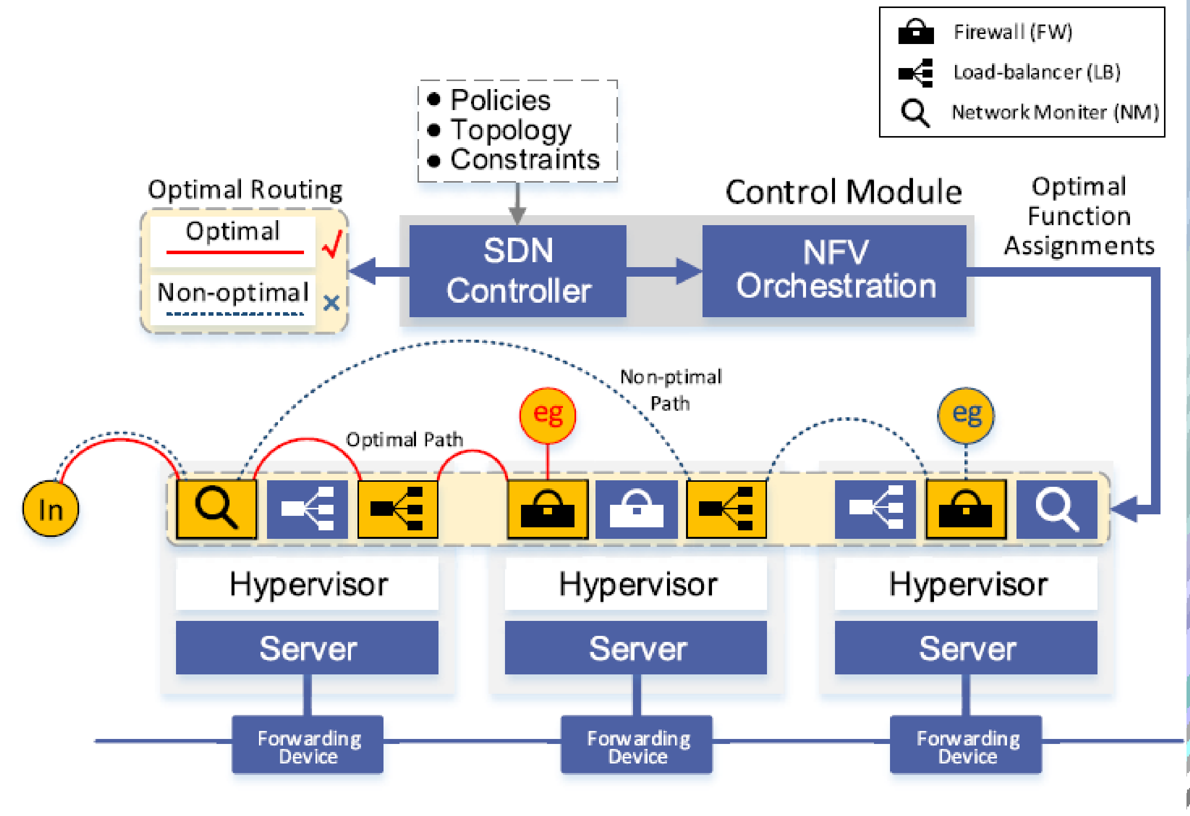
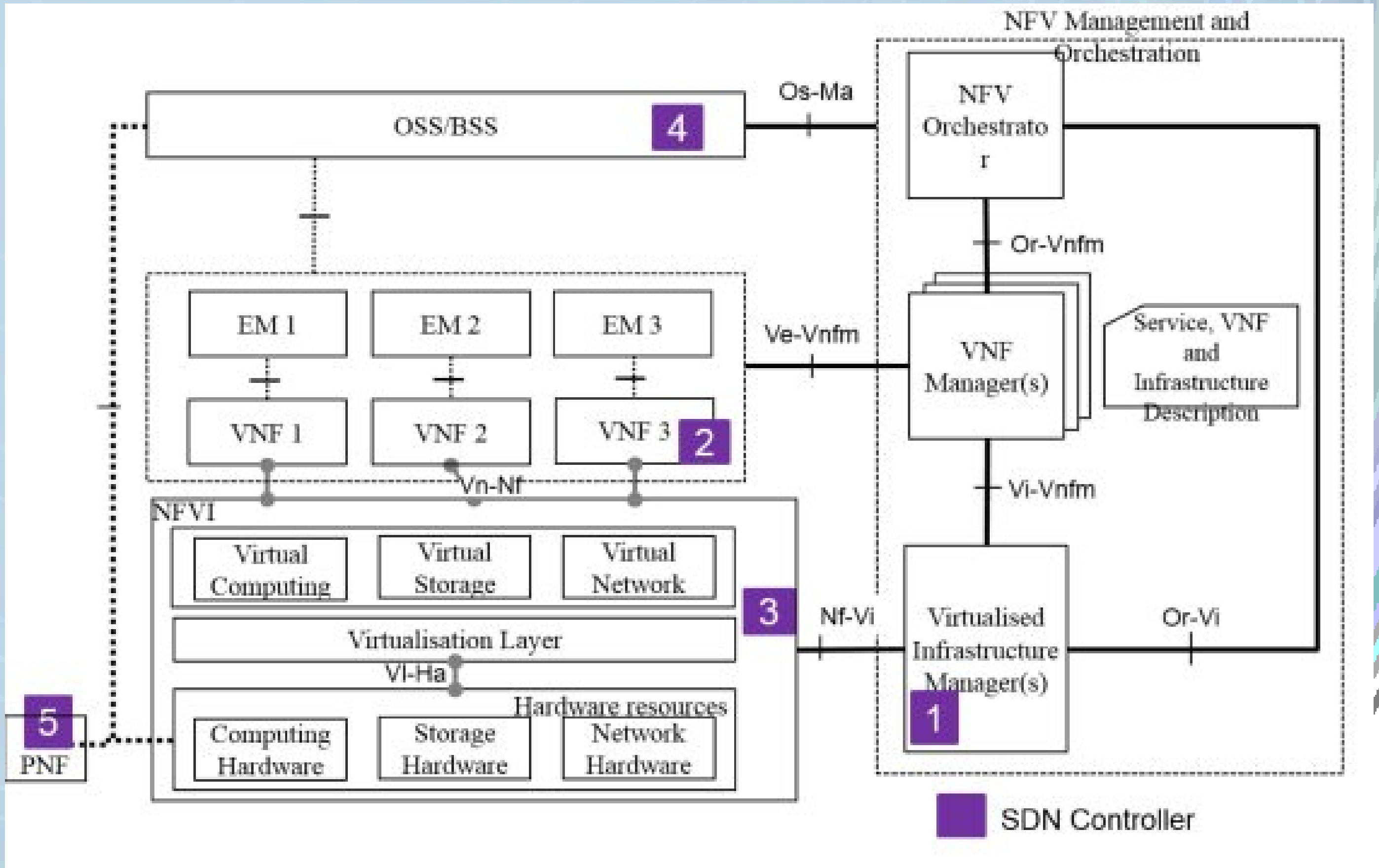


FIGURE 4. Service chaining in the software-defined NFV architecture.

Source: Software-Defined Networking

SDN CONTROLLER IN A SDNFV ARCHITECTURE

- Positioning of SDN controller in ETSI NFV architectural framework:
  1. SDN controller as a VIM, Virtualized Infrastructure Manager
  2. SDN controller as a VNF
  3. SDN controller in the NFVI
  4. SDN controller in the OSS/BSS
  5. SDN controller as a PNF



http://sdn.ieee.org/newsletter/may-2016/sdn

NETWORKING ARCHITECTURES AND PLATFORMS FOR SMART CITIES

ARCHITECTURES AND PLATFORMS: OVERVIEW

- IoT Service Experimentation Platforms (IoT SEPs) in urban contexts attract numerous stakeholders
  - city, research institutions, technology providers and users
- Definition of platform: "*as a reuse of sharing of common elements across complex products or systems of production*"

Source: A Framework for IoT Service Experiment Platforms

ARCHITECTURES AND PLATFORMS: OVERVIEW

- *Special Platforms in the SC context*
  - *Prototyping*: closed in-house design and development facilities
  - *Testbeds*: standardized environment for testing yet immature new technologies, products, services
  - *Field trials: agile platforms for specific small-scale tests*
  - *Living Labs*: for technology experimentation in real-life context, and their users are integrated to technology innovation process
  - *Market pilots*: when a product or a service is close to maturity and ready for commercialization
  - *Societal pilots*: mature new product or services in a real-life

A Framework for IoT Service Experiment Platforms in
Ballon, P., Pierson, J., & Delaere, S. (2005). Test and experimentation platforms for broadband innovation

ARCHITECTURES AND PLATFORMS: OVERVIEW

- Requirements for a SC application platform
  - Most existing smart city applications are concerned with the resources pertaining to one domain
    - energy, transportation, water, waste, etc.
  - However, smart city applications usually span multiple domains

Source: Smart City Platforms on Multitier Software-Defined Infra...

ARCHITECTURES AND PLATFORMS: OVERVIEW

- Architectural Requirements (1/2):
  - To support applications ranging from simple single-domain applications to complex multi-domain applications.
  - To scale down and be economically viable in small city settings,
  - To scale up to provide performance and economies of scale in large cities.
  - To enable a progressive deployment
    - According to the investment that is available,
    - and it should be progressively extensible as the
  - To allow the portfolio of applications and their scope to increase

Source: Smart City Platforms on Multitier Software-Defined Infra

ARCHITECTURES AND PLATFORMS: OVERVIEW

- Architectural Requirements (2/2):
  - operate the massive number of devices originating from the IoT domain
    - to manage and operate the massive amount of devices
  - perform analytics and manage data and the analytics' results
    - enable data processing by allocating the necessary resources where and when they're needed
      - Deploy elastic services
    - to handle high-volume data streams and large batches of data
      - in structured and unstructured formats.
  - need an effective way to plan based on this gathered information
    - enable the analytical models that provide the essential baseline for informed planning

Source: Enabling a Smart City Application Ecosystem: *Requirem*

ARCHITECTURES AND PLATFORMS: OVERVIEW (EXAMPLE)

- 4-Layered Smart and Connected Community (SCC)
  - Sensing: to realize ubiquitous sensing
    - RFID, WSN, Mobile Crowd Sensing (MCS)
  - Interconnecting: data transmission and information exchange
    - Different devices and domains
    - Careful network design is KEY
  - Data: storing massive heterogeneous data
    - extracting useful information from the big sensing data
    - representing the meaningful information
    - decision making and service supporting
    - knowledge maintenance and management
  - Service: services for communities

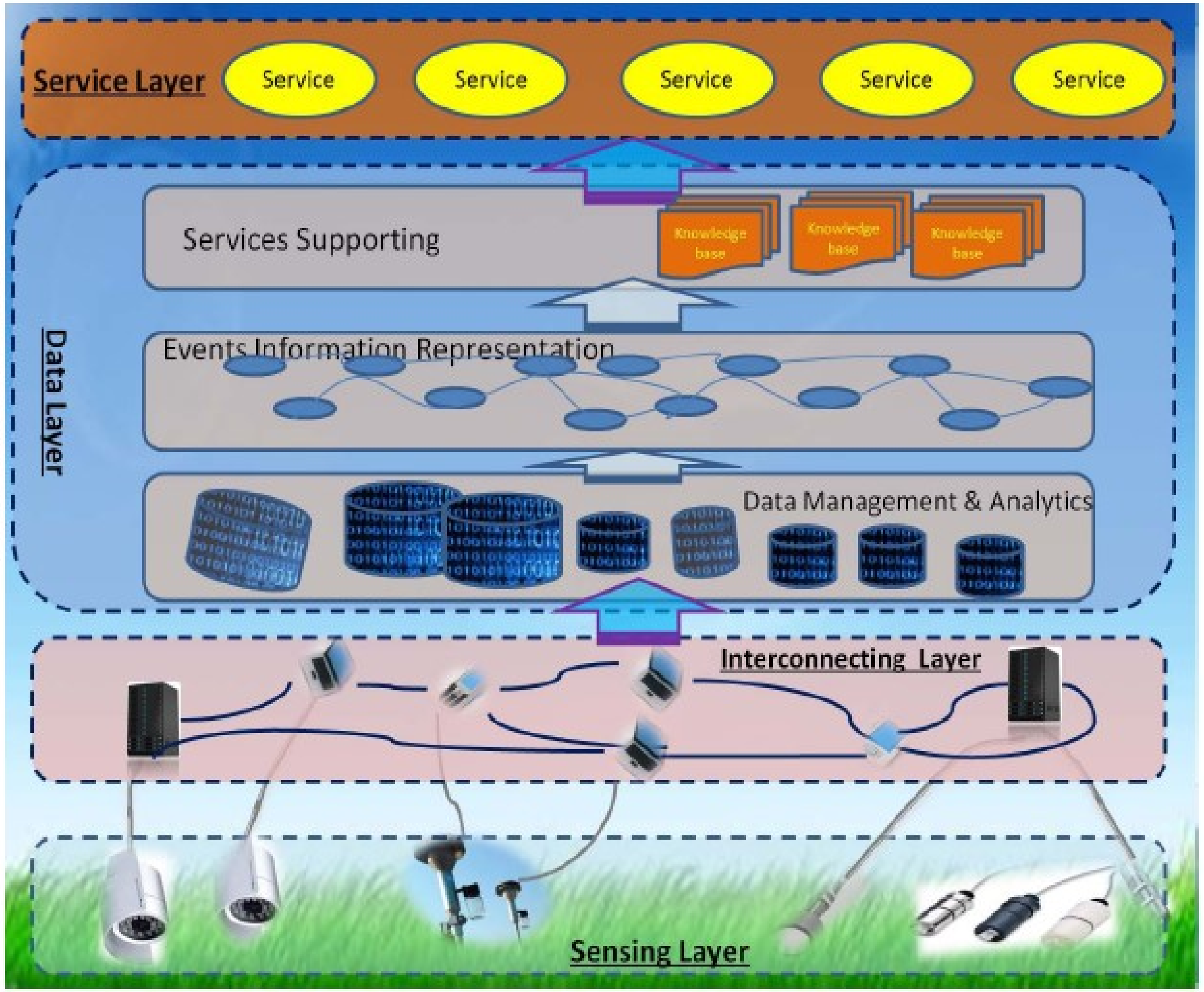


FIGURE 2. Function Architecture of Internet of Things for Smart and Connected Communities.

ARCHITECTURES AND PLATFORMS: OVERVIEW (EXAMPLE)

- Smart city Operating System (SOS) that enables a larger Smart City Application Ecosystem (SCALE)
  - a microservice architecture
    - break out of traditional layered architectures
      - Each component interacts with any other
      - Allows novel synergies between components
- SOS Components
  - Infrastructure and resource management
  - Data management
  - Application runtime and management

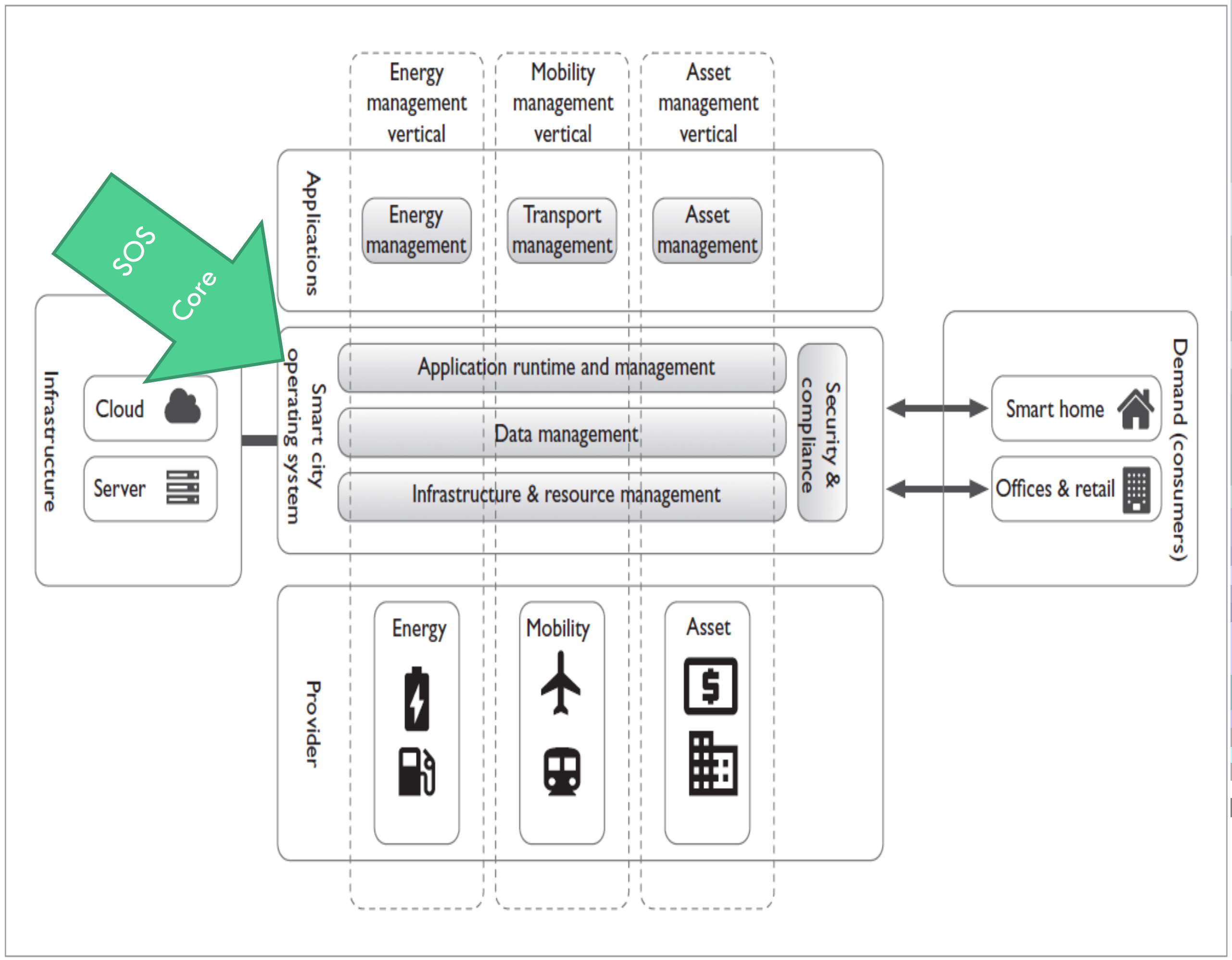


Figure 2. Architecture overview of the Smart City Application Ecosystem (SCALE). Designed around a central middleware — the Smart City Operating System — SCALE allows for the seamless integration of relevant stakeholders and resources to efficiently build, deploy, and operate smart city applications.

ARCHITECTURES AND PLATFORMS: MULTITIER SDN

- *The Need for Software-Defined Infrastructure (1/2)*
  - virtualized resources are offered by a IaaS layer
    - through a set of unified and open interfaces (APIs)
      - allow external entities to acquire, reconfigure, and release virtualized resources.
  - Infrastructure-aware services:
    - e.g. smart resource scheduling, fault tolerance, and green energy management.
  - The smart edge provides virtual machines and software-defined networking services.
  - Resource/Service allocation and deallocation in virtual environments require dynamic management of network resources

Source: Smart City Platforms on Multitier Software-Defined In

ARCHITECTURES AND PLATFORMS: MULTITIER SDN

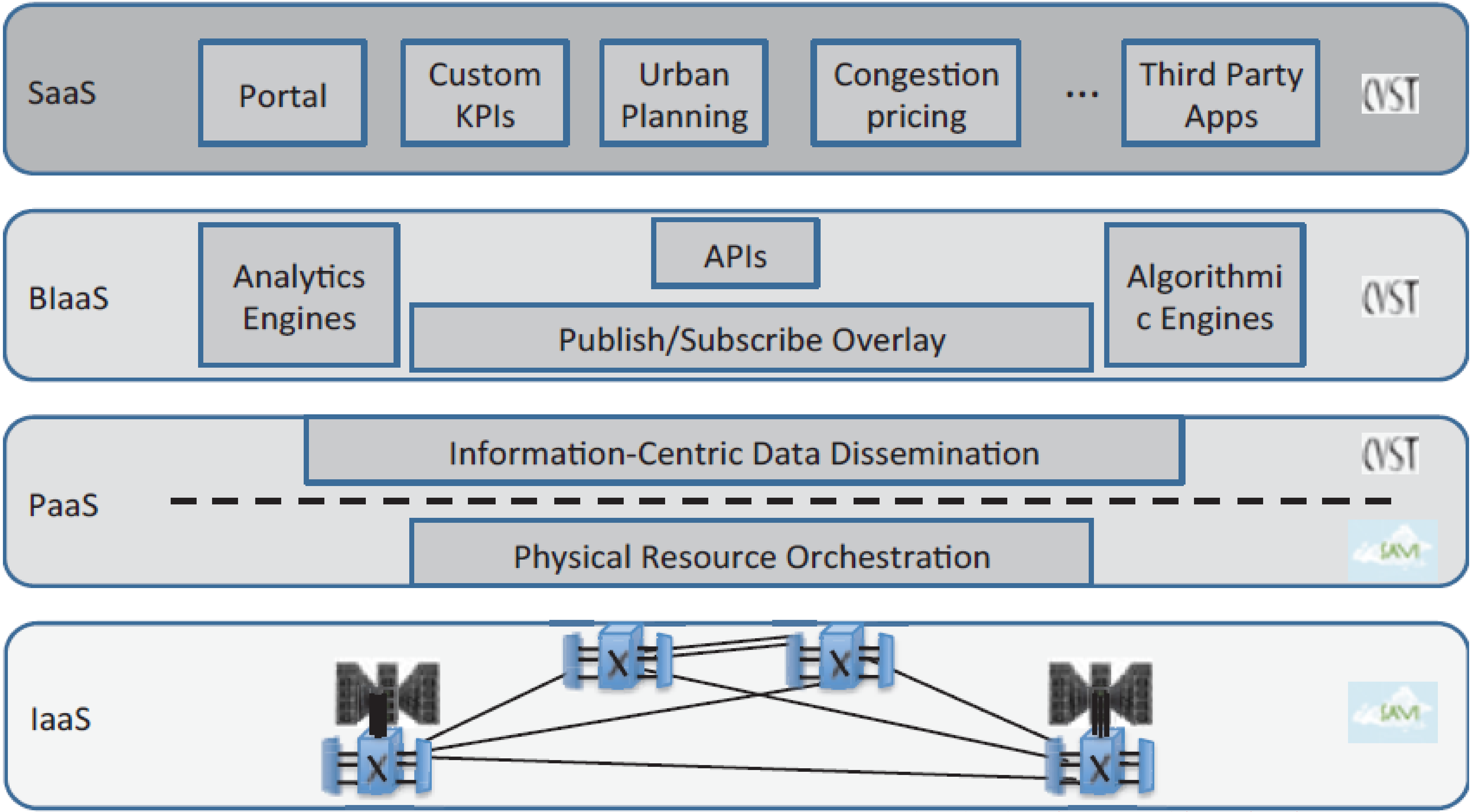
- *The Need for Software-Defined Infrastructure (2/2)*
  - *programmability of resources across the multitier cloud*
  - *Handling of different traffic flows with different QoS requirements.*
- *SDN allows*
  - *Programability of the network*
    - *Facilitating the analytics and intelligence to support smart applications.*
  - *Allows migration of resources (e.g., VM or services) and multilayer monitoring*
    - *Increased resiliency and robustness as well as security, privacy, and isolation*

Source: Smart City Platforms on Multitier Software-Defined Infra

ARCHITECTURES AND PLATFORMS: MULTITIER SDN

- Three-layered architecture for Smart City Platforms
  - An Infrastructure-as-a-Service (IaaS) layer
    - Provides resources in a SDN-based multitier computing cloud that is based on
  - A Platform-as-a-Service (PaaS) layer
    - Provides data dissemination services for various domains
  - A Business-Intelligence-as-a-Service layer (BIaaS)
    - Provides intelligence to support smart applications

Source: Smart City Platforms on Multitier Software-Defined



Source: Smart City Platforms on Multitier Software-Defined Infrastructure Cloud Computing

ARCHITECTURES AND PLATFORMS: SPARTACUS

- SPArTaCuS: Service Priority Adaptiveness for Emergency Traffic in Smart Cities using SDN
  - A framework for smart cities to prioritize services for emergency needs in a stressed situation
  - Relies on the underlying network function provided by SDN and Network Virtualization
  - Create virtual SDN networks for different service classes
    - Mapped to the physical infrastructure

SPArTaCuS: Service priority adaptiveness for emergency traffic in smart cities using

ARCHITECTURES AND PLATFORMS: SPARTACUS

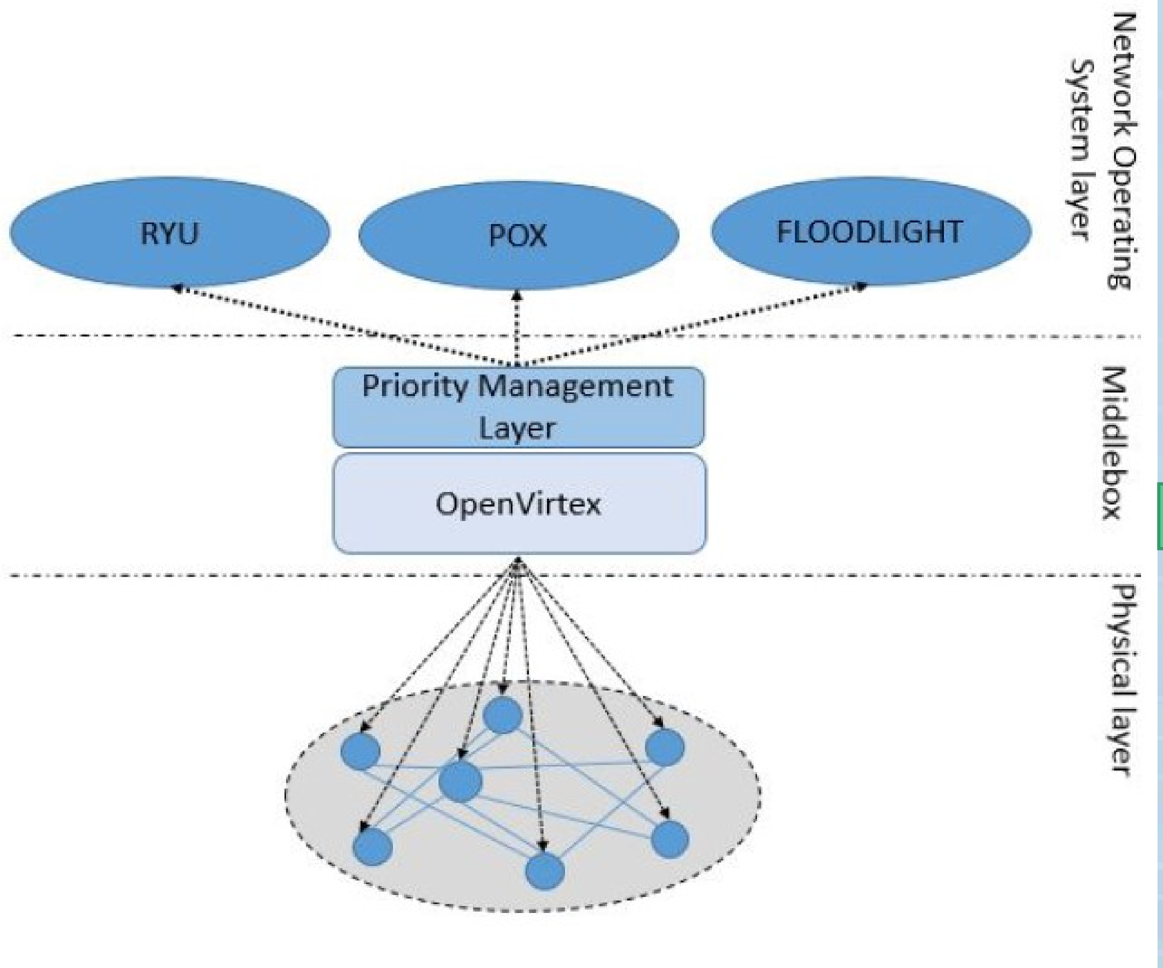


Fig. 4. SPArTaCuS: Architecture Framework

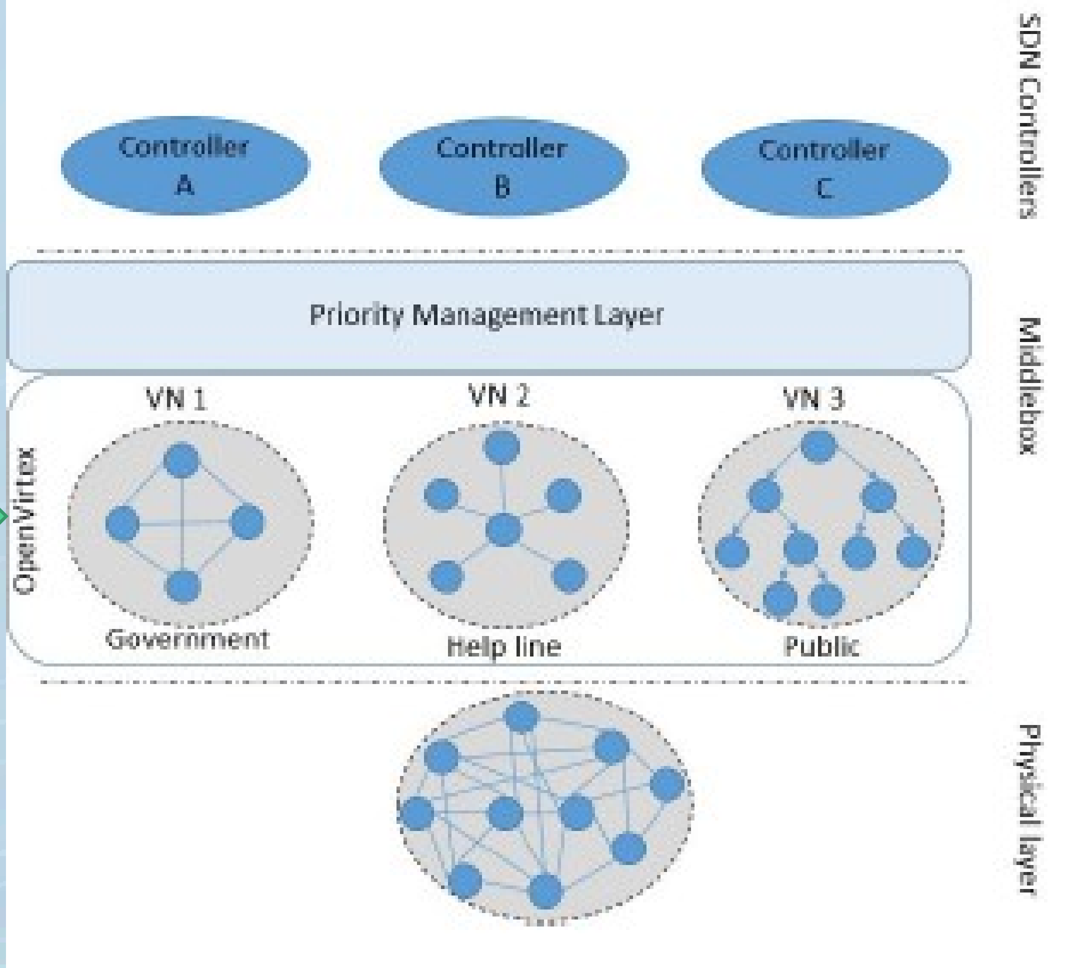


Fig. 5. Modelling Smart Cities networks in SPArTaCuS

ARCHITECTURES AND PLATFORMS: SPARTACUS

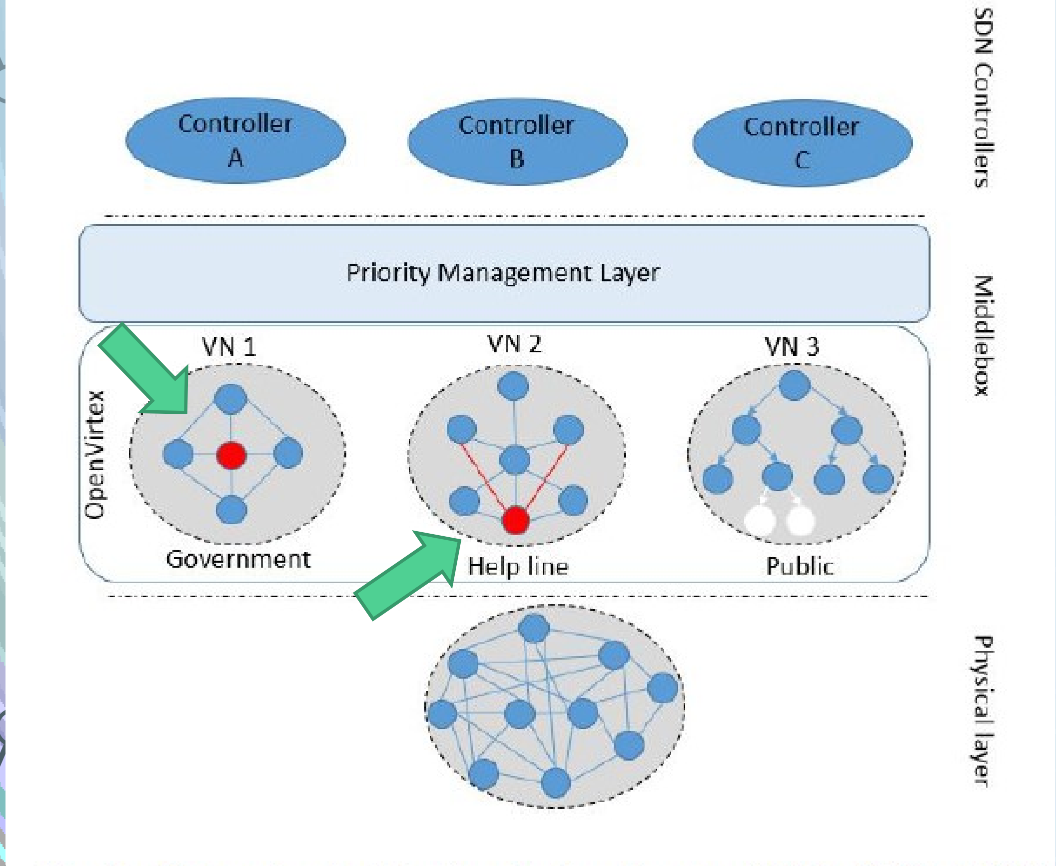


Fig. 6. Network prioritization during disaster (VN2≻VN1≻VN3)

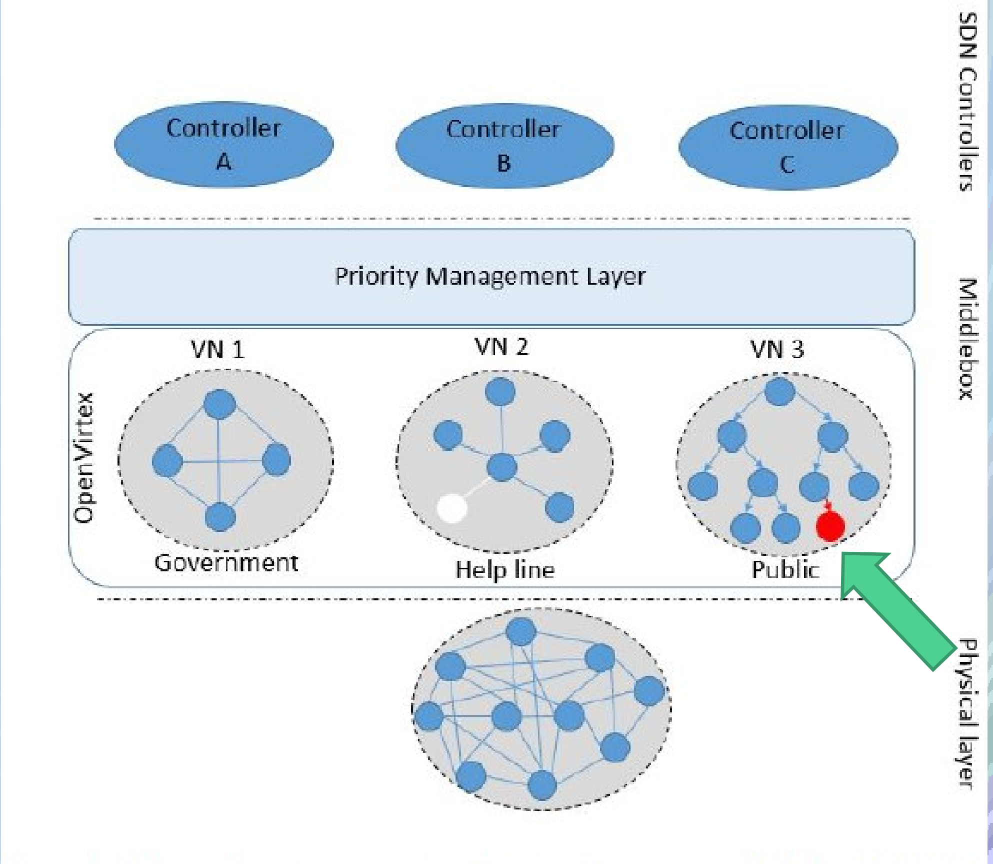


Fig. 7. Network prioritization during big events (VN3≻VN2≻V

SPArTaCuS: Service priority adaptiveness for emergency traffic in smart cities using

ARCHITECTURES AND PLATFORMS: CIGO!

- Smart Mobility and Smart Governance are the most common indicators of Smart Cities
- CIGO!: A novel mobility management platform and business model
  - track people flows in urban areas to achieve global mobility objectives
    - allows city governments formulate mobility policies
  - Includes
    - Architectural concept for the CIGO! platform,
  - A platform prototype and application pilot (city of Barcelona)

Source: CIGO! Mobility Management Platform for Growing Efficient and B

ARCHITECTURES AND PLATFORMS: CITY OF THINGS

- Premises: Testbeds are the preferred tools to validate R&D&I in a larger environment
  - provide access to test technological solutions
    - emulated or real environment
  - provide a control environment
    - Fine tuning parameterization of the experiments
    - Allow reproducibility of the experiments

Source: City of Things: an Integrated and Multi-Technology Testbed for

ARCHITECTURES AND PLATFORMS: CITY OF THINGS

- City of Things testbed: Realistic city living lab and technical testbed environment
  - An integrated approach of doing network, data and living lab experiments
- At the network side: a myriad of wireless technologies
- At the data side: technological capacity to quickly collect, analyse and publish city data
- At the user side: a largescale living lab in Antwerp
  - User research takes place in a real-life environment

Source: City of Things: an Integrated and Multi-Technology Testbed for

ARCHITECTURES AND PLATFORMS: CITY OF THINGS

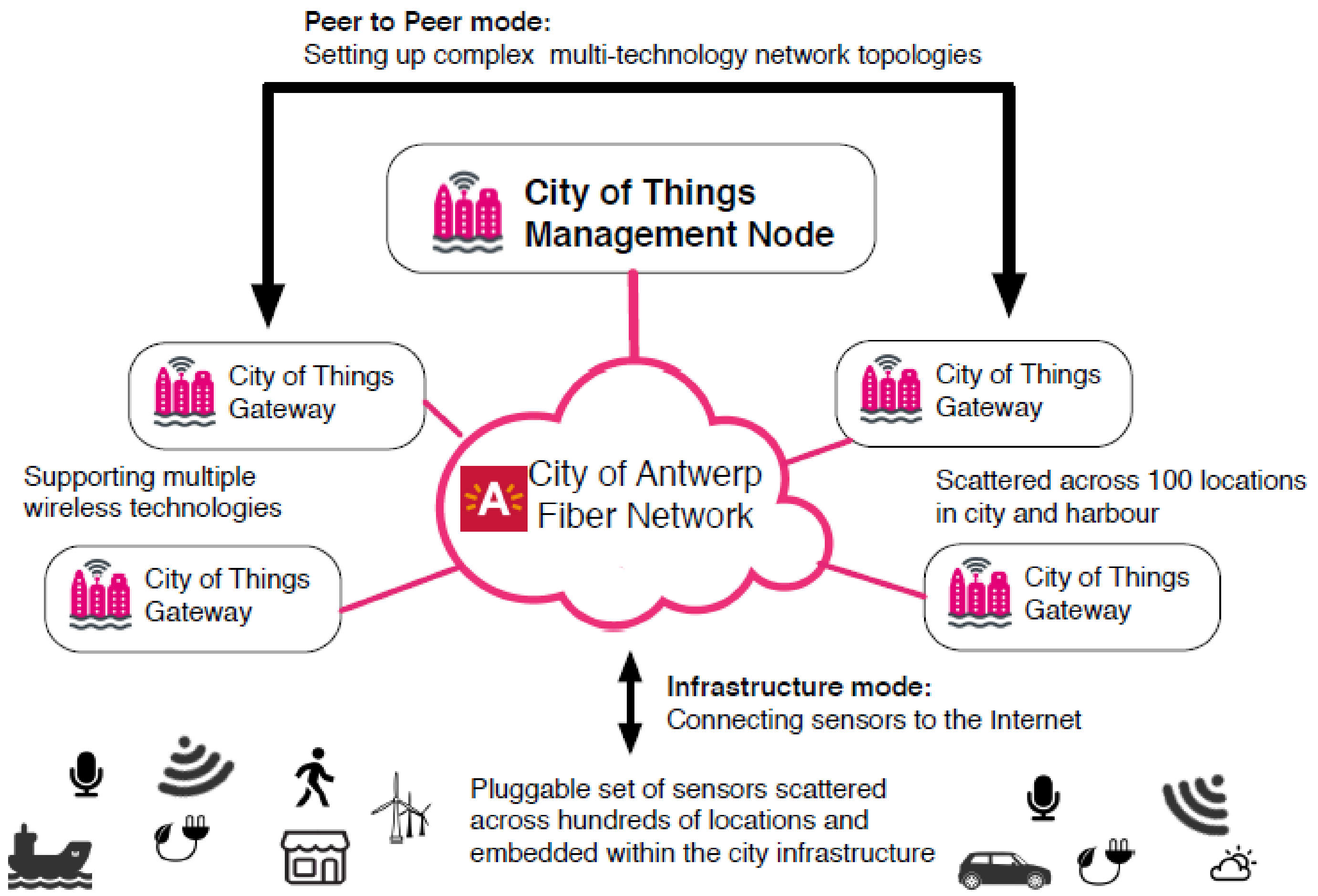


Fig. 1. An overview of the City of Things gateways and their integration within the city's network. Each gateway consists of multiple wireless technologies and can both be used for connecting sensors (infrastructure mode) and complex network topologies (peer to peer mode).

Source: City of Things: an Integrated and Multi-Technology Testbed for

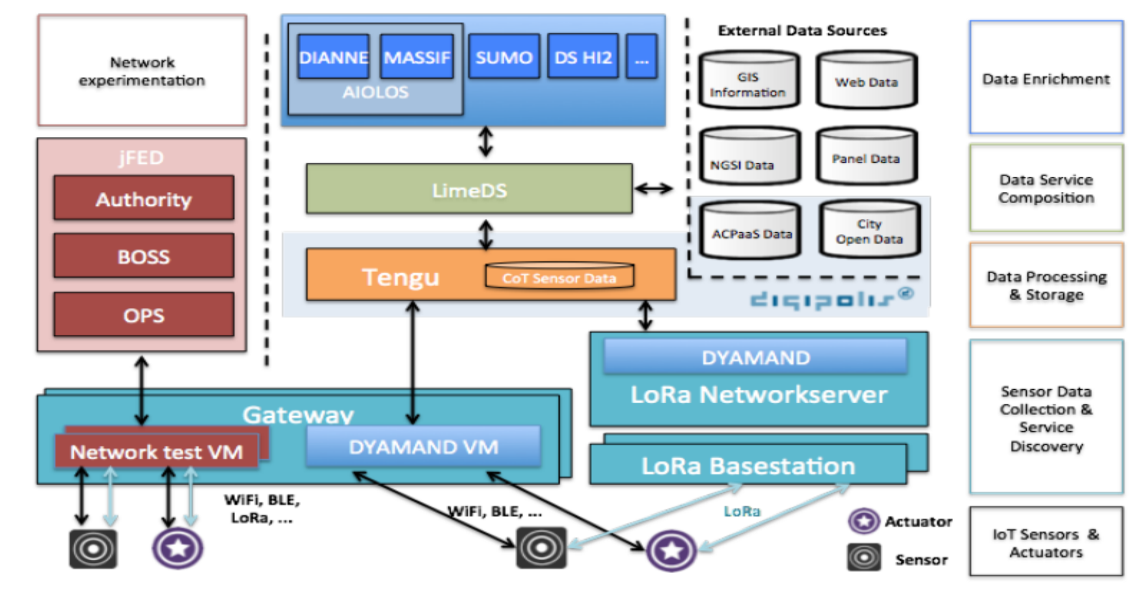
ARCHITECTURES AND PLATFORMS: CITY OF THINGS

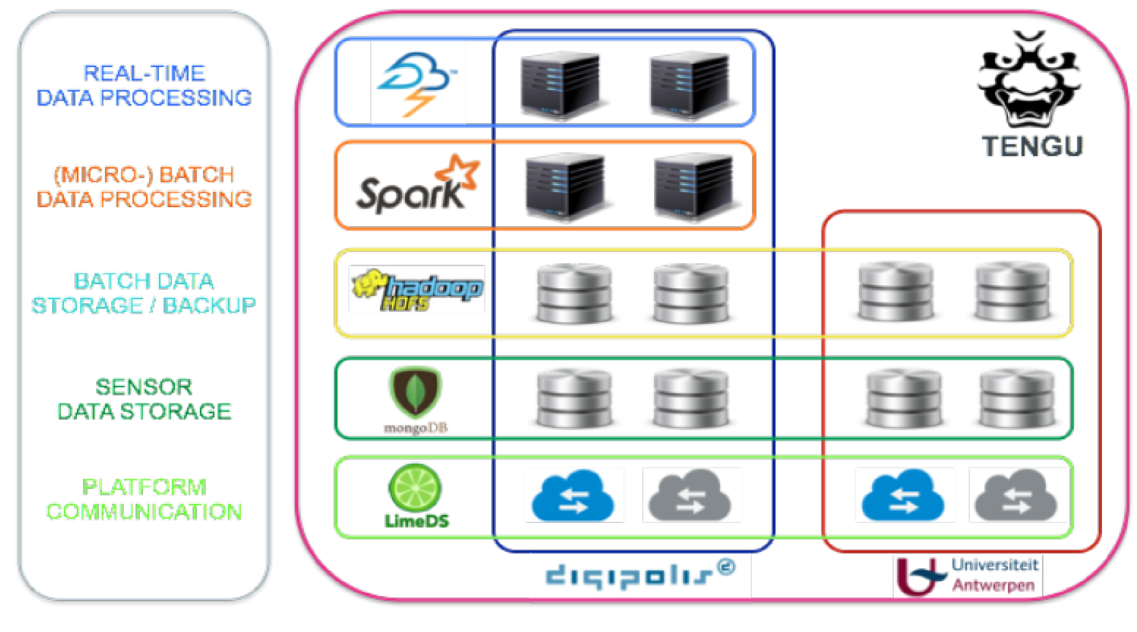
- Network level: Internet of Things infrastructure
  - Multi-technology gateways (dedicated onboard SoC radios)
    - IEEE 802.1ac on 2.4 and 5 Ghz,
    - DASH7 on 433 and 868 Mhz
    - Bluetooth (Low Energy),
    - IEEE 802.15.4, IEEE 802.15.4g
    - LoRa
  - LoRa-based Low Power WAN network
  - City sensors
    - Mobile air quality sensors
    - Traffic monitoring
    - Parking sensors
    - Smart parking signs

Source: City of Things: an Integrated and Multi-Technology Testbed for

ARCHITECTURES AND PLATFORMS: CITY OF THINGS

- Data level: Big data analysis
  - three main tools
    - DYAMAND (DYnamic, Adaptive Management of Networks and Devices) for sensor data collection and discovery
    - Tengu for sensor data processing and storage
      - an experimentation platform for big data applications
    - LimeDS (Lightweight modular environment for Data oriented Services) for data access, service composition and demo prototyping





Source: City of Things: an Integrated and Multi-Technology Testbed for

ARCHITECTURES AND PLATFORMS: CITY OF THINGS

- User level: a large-scale city living lab

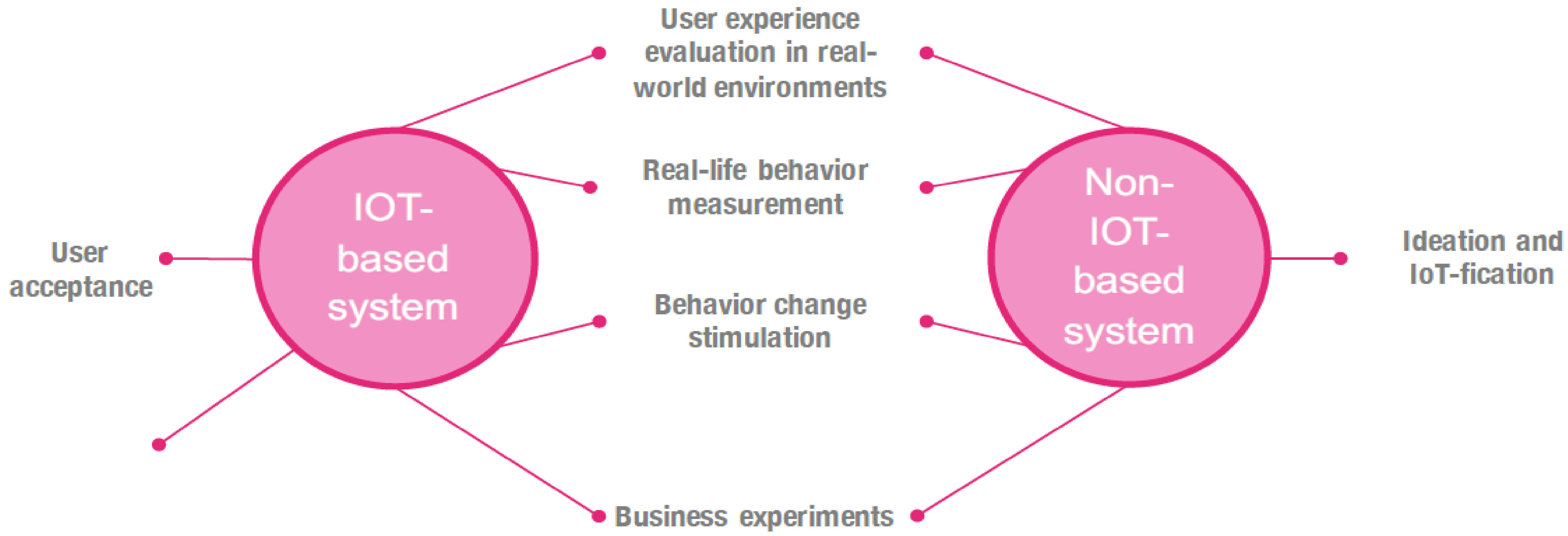


Fig. 4. Overview of the Living Lab services offered as part of the City of Things testbed.

Source: City of Things: an Integrated and Multi-Technology Testbed for IoT Smart City Experiments

ARCHITECTURES AND PLATFORMS: SDIV

- Internet of Vehicles (IoV)
  - Applications and services:
    - road security, fleet management, navigation, billing, and multimedia
- Network Architecture: Software-Defined IoV
  - new network architecture for IoV
  - Three-tier architecture
    - Physical: vehicles (mobile nodes), access points (APs), roadside electronic devices, switches, and servers
    - Control: SDN controller connected to every switch (including APs) via OpenFlow
    - Application: services for vehicles
      - query, location and road information service

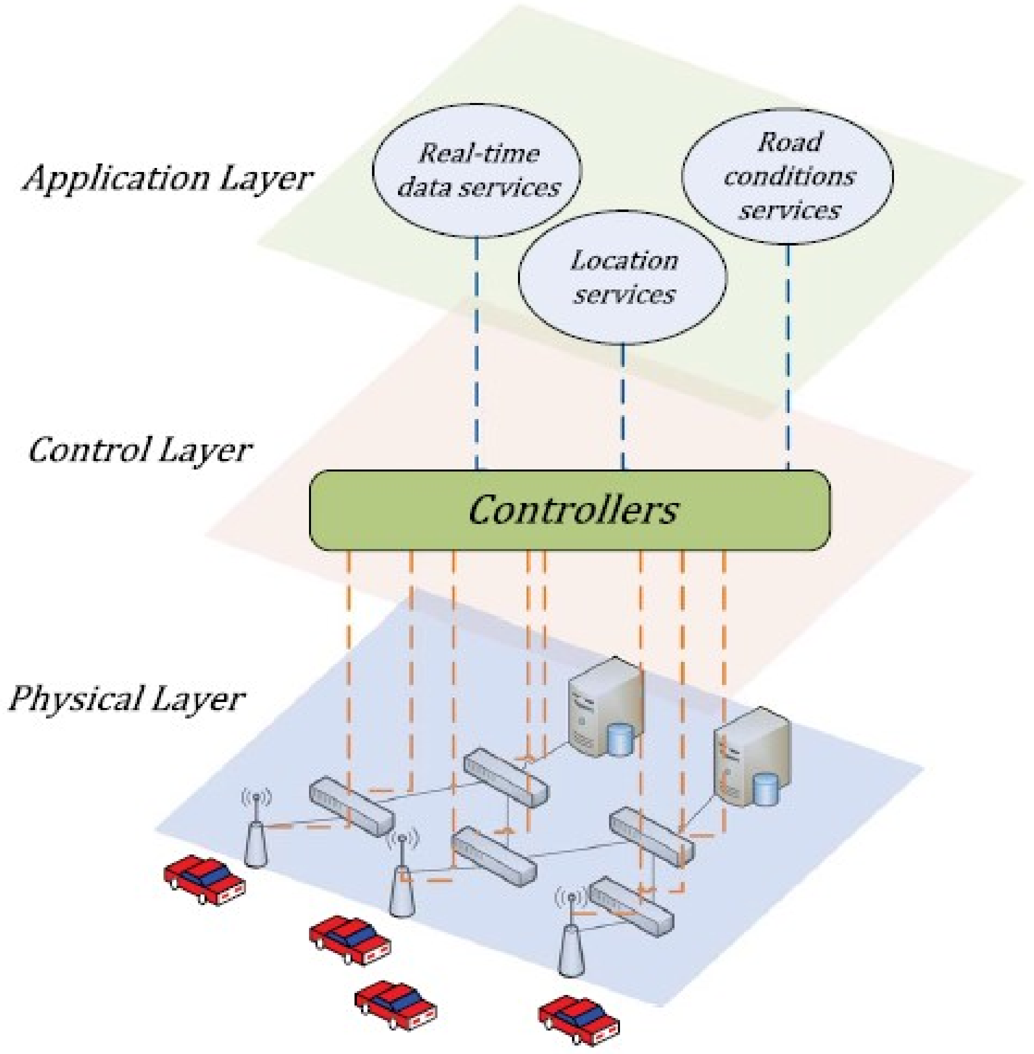


Fig. 2. A three-tier SDIV architecture.

Source: Improved Rule Installation for Real-Time Query Service in Software-D

ARCHITECTURES AND PLATFORMS: SDIV

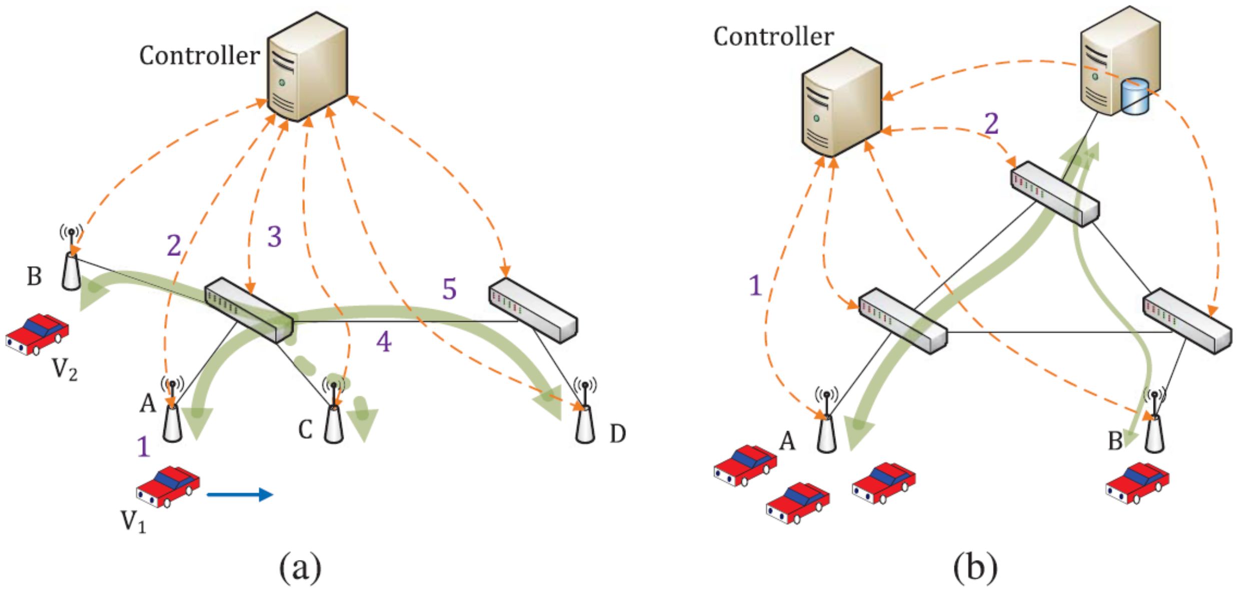


Fig. 3. Scenarios (a) by analyzing vehicle conditions, the controller can install rules (dash line) in advance to avoid extra queries; and (b) achieving intelligent bandwidth allocation based on a global view in the controller.

TABLE I
PROS AND CONS OF TRADITIONAL NETWORK TECHNOLOGIES (MULTICAST) AND SDIV

Traditional Technologies	SDIV
Con: Broadcasting messages periodically	Pro: Reactive mode
Con: Keeping (S, G) entries at routers	Pro: Installing rules when needed
Con: SPT cannot match the driving path	Pro: Finding the path according to the direction of vehicles
Pro: Do not need a controller	Con: Need a controller

Source: Improved Rule Installation for Real-Time Query Service in Softwar

ARCHITECTURES AND PLATFORMS: SC BASED ON IOT USING BDA

- 4-tier architecture
  - Bottom tier-1: IoT sources and data generation and collection
  - Inter-mediate tier-1: all types of communication between sensors, relays, base stations, and the Internet
  - Intermediate tier 2: data management and processing using a Hadoop framework
  - Top tier: application and usage of the data analysis and the results generated

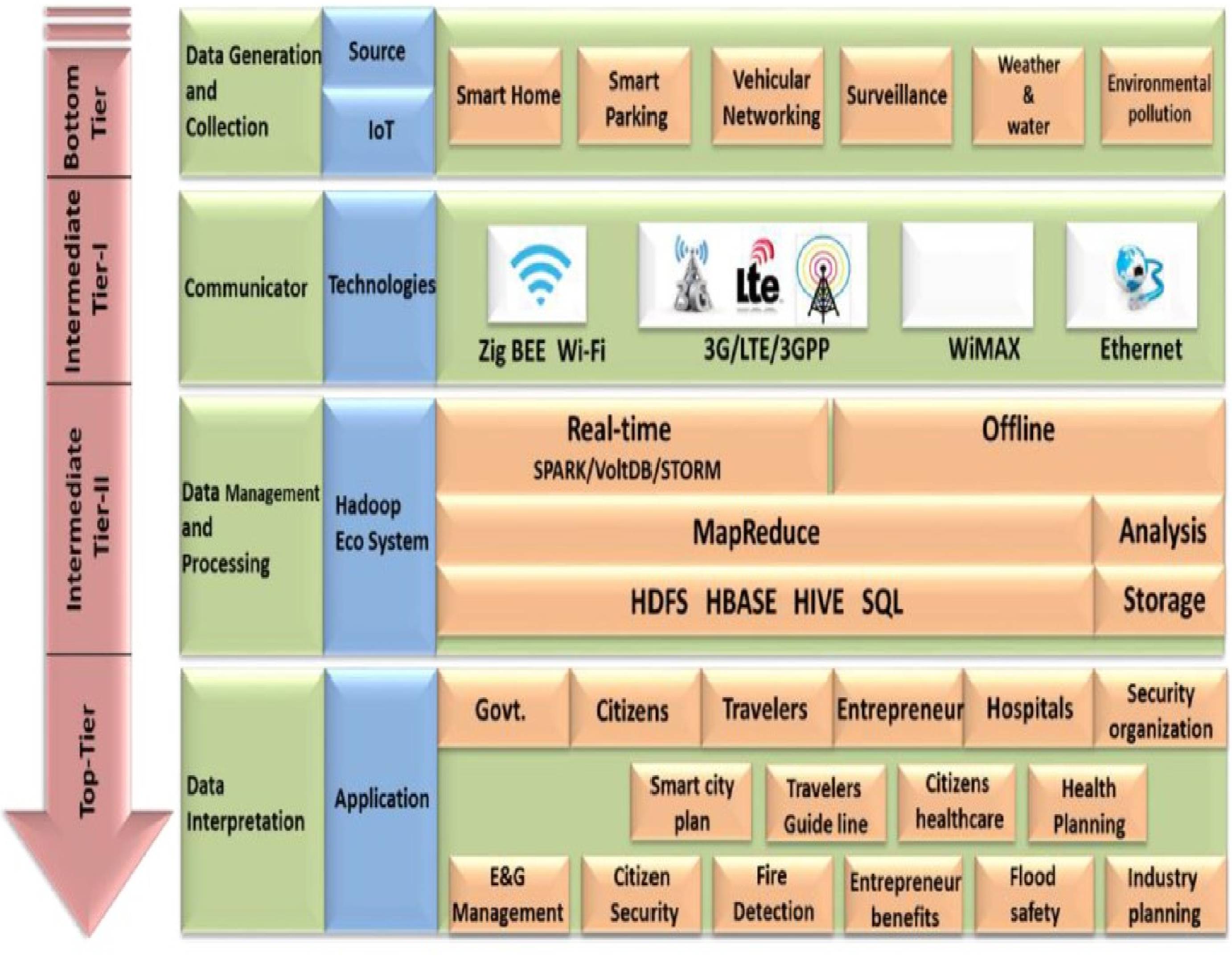


Fig. 2. IV-tier architecture for IoT Big Data analytics for remote smart city and urban planning.

Source: Urban planning and building smart cities based on the Internet of Things using Big D

ARCHITECTURES AND PLATFORMS: SC BASED ON IOT USING BDA

- General Questions
  - How to tackle uncertainty due to real-time and offline dynamic?
  - How to make existing objects smarter?
  - How to enable objects to react accordingly to context?
  - How to minimize the cost of data collection?
  - How to obtain insight into the data in real time?

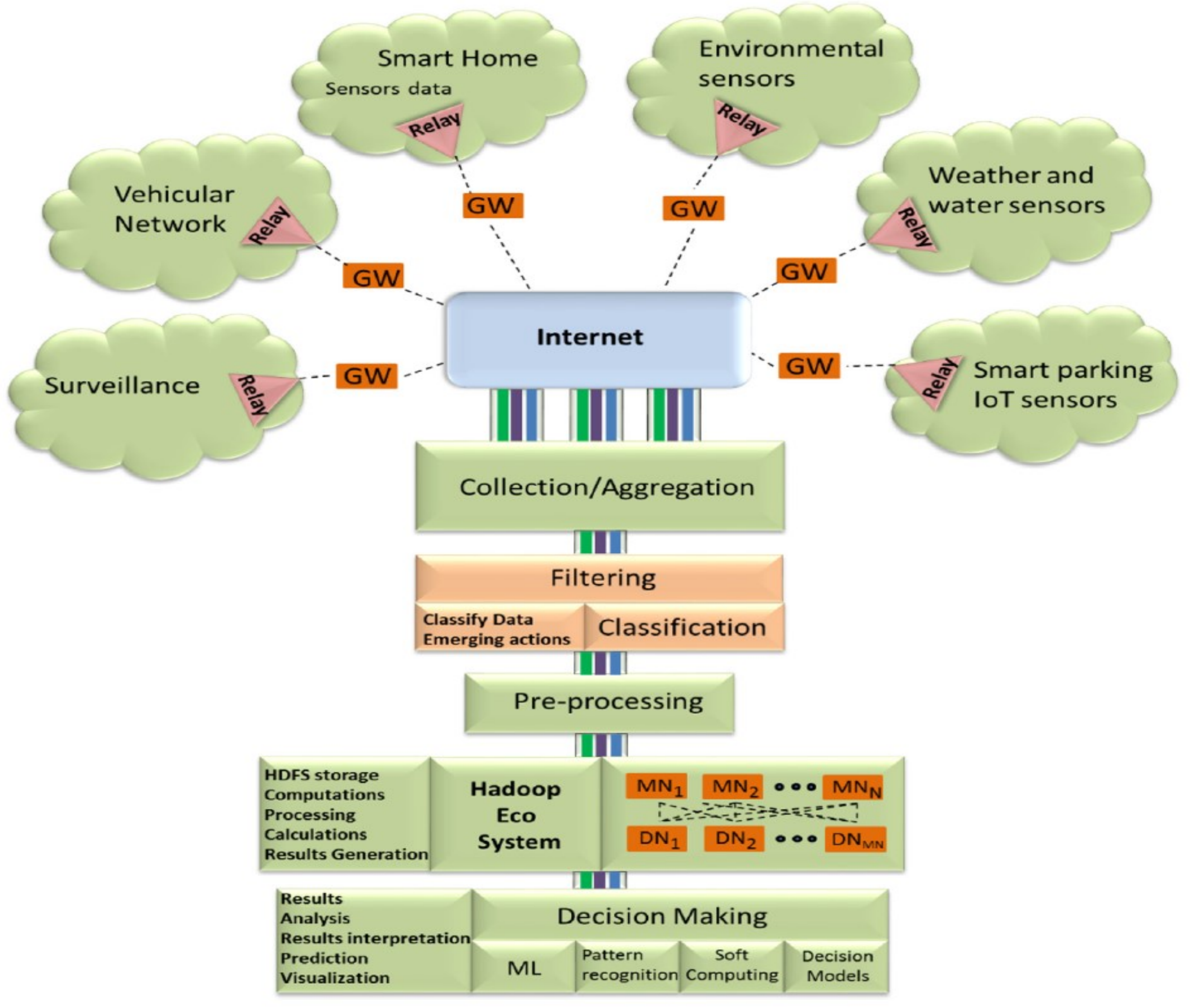


Fig. 3. Implementation model.

Source: Urban planning and building smart cities based on the Internet of Things using Big D

ARCHITECTURES AND PLATFORMS: CITYPULSE

TABLE 1. IoT and Smart City Frameworks Comparison (●:Yes ○: No ⊙: Partial).

IoT Smart City Platforms and their supported features	iCity	Smart Santandler	Open IoT	iCore	Spit Fire	PLAY	Star City	VITAL	CityPulse
IoT Data Collection	●	●	●	●	●	●	●	●	●
Semantic Interoperability	⊙	⊙	●	●	●	○	●	●	●
Event Detection and Data Analytics	○	○	○	○	○	●	●	○	●
Application Development Support	●	●	⊙	⊙	⊙	○	○	⊙	●

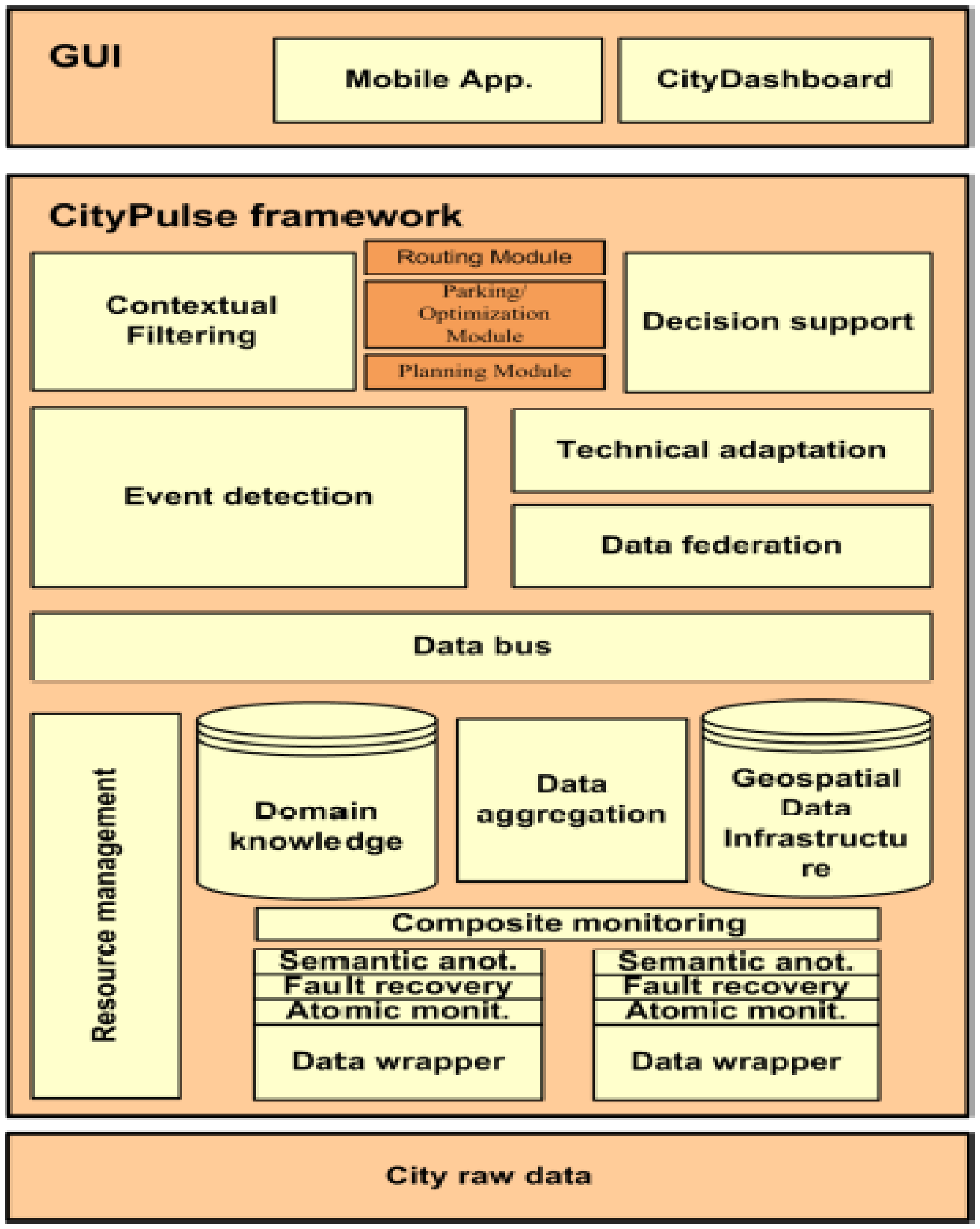


FIGURE 1. The components of the CityPulse framework with their APIs.

Source: CityPulse: Large Scale Data Analytic

CITY-PULSE

- IoT Stream Processing Framework:
  - Virtualisation: facilitates access to heterogeneous data sources and infrastructure
  - Middleware: Advanced Message Queue Protocol (AMQP), an open standard for message oriented middleware
  - Data aggregation: Symbolic Aggregate Approximation (SAX) and SensorSAX
  - Reliable information processing: measures and process accuracy and trust in data acquisition, federation and aggregation

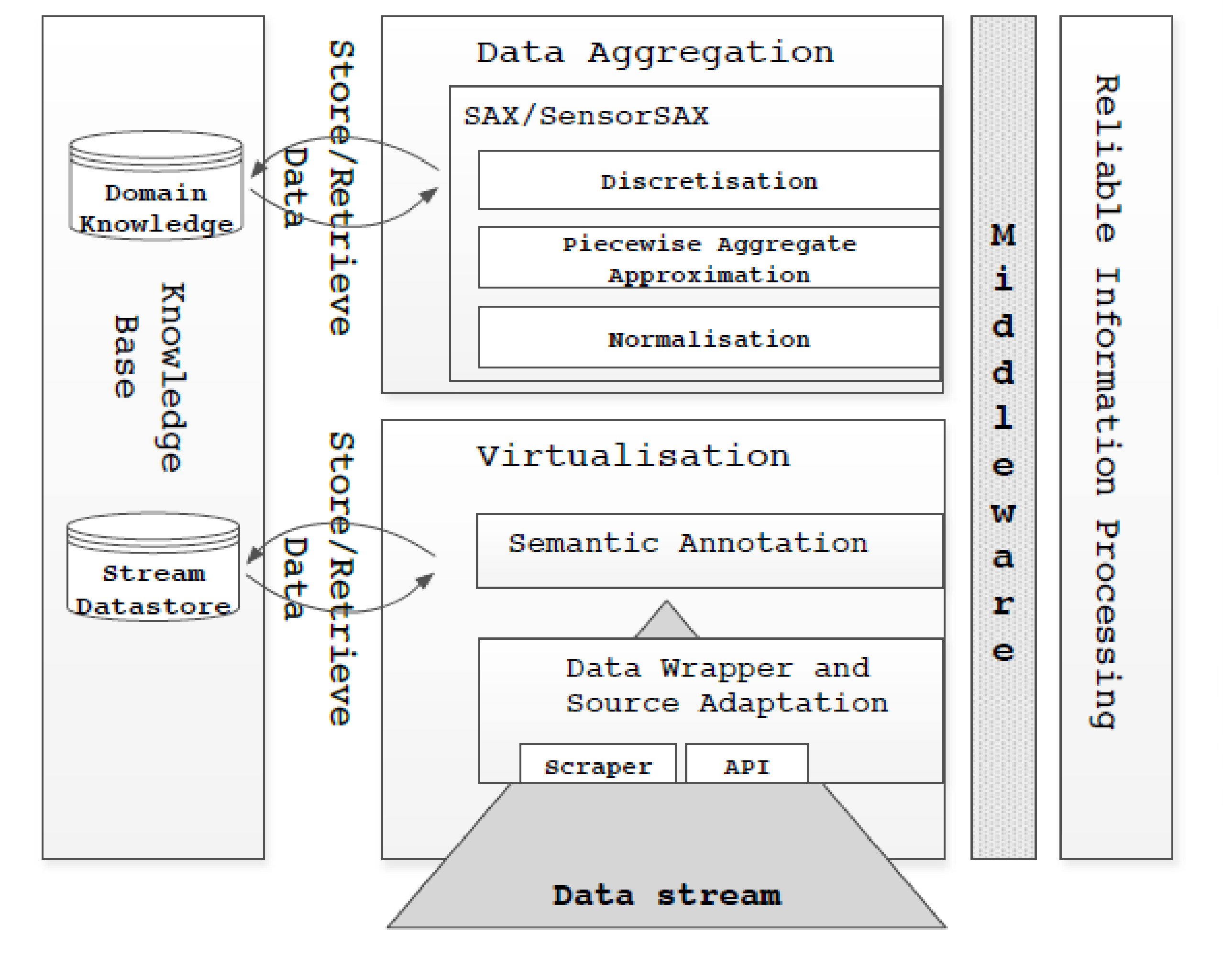


Fig. 1: Real-time IoT stream processing components of the CityPulse smart city framework.

SMART CITIES-RELATED NETWORK SIMULATION, EMULATION, AND *TESTBEDS*

Given the complexity of putting all layers together in the context of Smart Cities

There is NO SIMULATION ENVIRONMENT AVAILABLE

that covers the current networking technologies in a single piece of software

SIMULATION AND EMULATION: BREAKDOWN

- Protocols/WSN/IOT
  - NS-2, OMNeT++, PAWiS, GloMoSim/QualNet, OPNET, SENSE, J-Sim, Ptolemy II, Cell-DEVS, NesCT, GTnets, System C, Prowler, NCTUns2.0, Jist/SWANS, SSFNet, TOSSIM, Atarraya, Avrora, ATEMU, EmStar, SENS, Shawn, PiccSIM, TrueTime 2.0 in MATLAB/Simulink, and native MATLAB/Simulink

Table 1. Comparative perspectives of simulation frameworks.

Framework	Operating System	Compiler	Latest Update	Programming Language	Node Size	MATLAB/ Simulink Integration	ZigBee Support
NS-2	Unix/Windows with Cygwin	C++, JDK 1.6	NS-2.35/2013	Tcl/Object Tcl (OTcl)	100 nodes Maximum	Yes	Yes
OMNeT++	Windows, OS X, Linux	C++11, JDK 1.7 or later	OMNeT 4.6/2014	NED Language	—	Yes	Yes
Prowler	OS that supports MATLAB	Apple Xcode version 4.0 or higher; Windows: C++, JDK	V1.25/2004	Graphical programming tool (graphical user interface)	Based on the type of application	Yes	No
Atarraya	Windows, requires graphical user interface formatting for Linux	Java 6	1.3 beta/2011	Graphical user interface	Can simulate 1,000 nodes	No	No
PiccSIM	Windows, OS X, Linux	Apple Xcode version 4.0 or higher; Windows: C++, JDK	PiccSIM Simulink version 1.16/2013	Tcl/Otcl for network modeling	Similar to NS-2	Yes	Yes
TrueTime	Windows, OS X, Linux	Apple Xcode version 4.0 or higher; Windows: C++, JDK, Microsoft Visual Studio	TrueTime 2.0 beta 7/2012	Graphical Programming tool	Limited	Yes	Yes
MATLAB/ Simulink	OS X, Windows, Linux	Apple Xcode version 4.0 or higher; Windows: C++, JDK	R2015a	C, C++, Fortran	Code: > 100 nodes; Simulation: Restricted	—	Yes

Source: Wireless Sensor Network Simulation Frame

SIMULATION AND EMULATION: BREAKDOWN

- SDN/NFV
  - **Mininet** (Emulated Environment)
    - Quick and easy way to prototype and evaluate SDN protocols and applications
    - Use of software-based OpenFlow switches in virtualized containers
    - Controllers or applications developed and tested in Mininet can be (in theory) deployed in an OpenFlow-enabled network without any modification
  - **Mininet-HiFi**: enhances the container-based (lightweight) virtualization with mechanisms to enforce performance isolation
  - **Mininet CE and SDN Cloud DC**: extensions to Mininet
    - enabling large scale simulations.
  - **ns-3**: OpenFlow devices has been added
  - **fs-sdn**: extends the fs simulation engine
  - **STS** - SDN troubleshooting simulator

Source: Wireless Sensor Network Simulation Frame

FACILITIES FOR EXPERIMENTAL IOT RESEARCH

- Main requirements for a next generation of experimental research facilities for the IoT
  - Scale: many IoT experiments demand larger scale
    - 1000’s of nodes
  - Heterogeneity: heterogeneity of devices and underlying solutions
  - Repeatability: requires agreement on standards for the specification of experiments
  - Federation: achieve scale or add capabilities for experimentation that are not locally available
  - Concurrency: support multiplexing of concurrent experiments
  - Mobility: mechanisms to control and exploit realistic mobility of devices
  - User Involvement: offer mechanisms allowing for evaluating social impact and acceptance of IoT solutions and applications

Source: A Survey on Facilities for Experimental

FACILITIES FOR EXPERIMENTAL IOT RESEARCH

- Taxonomy

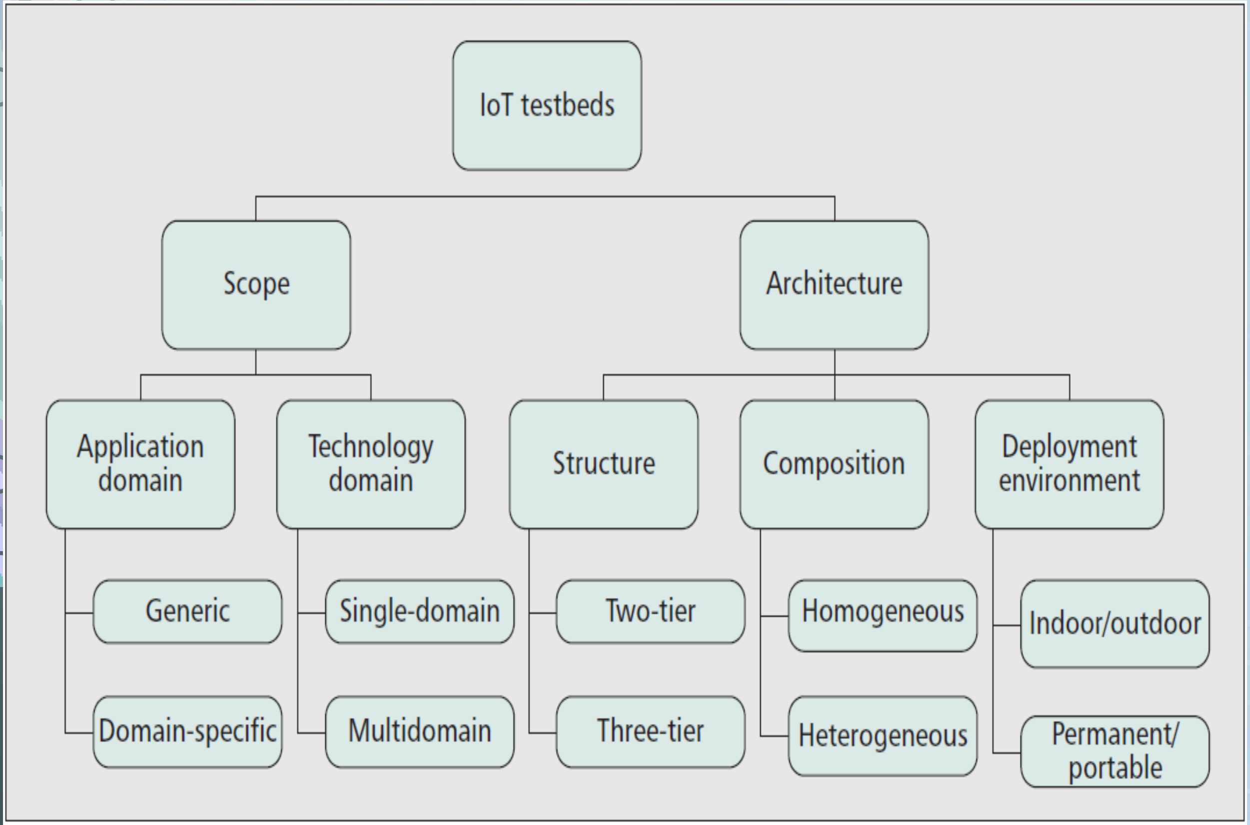


Figure 1. *Scope and architecture of testbeds.*

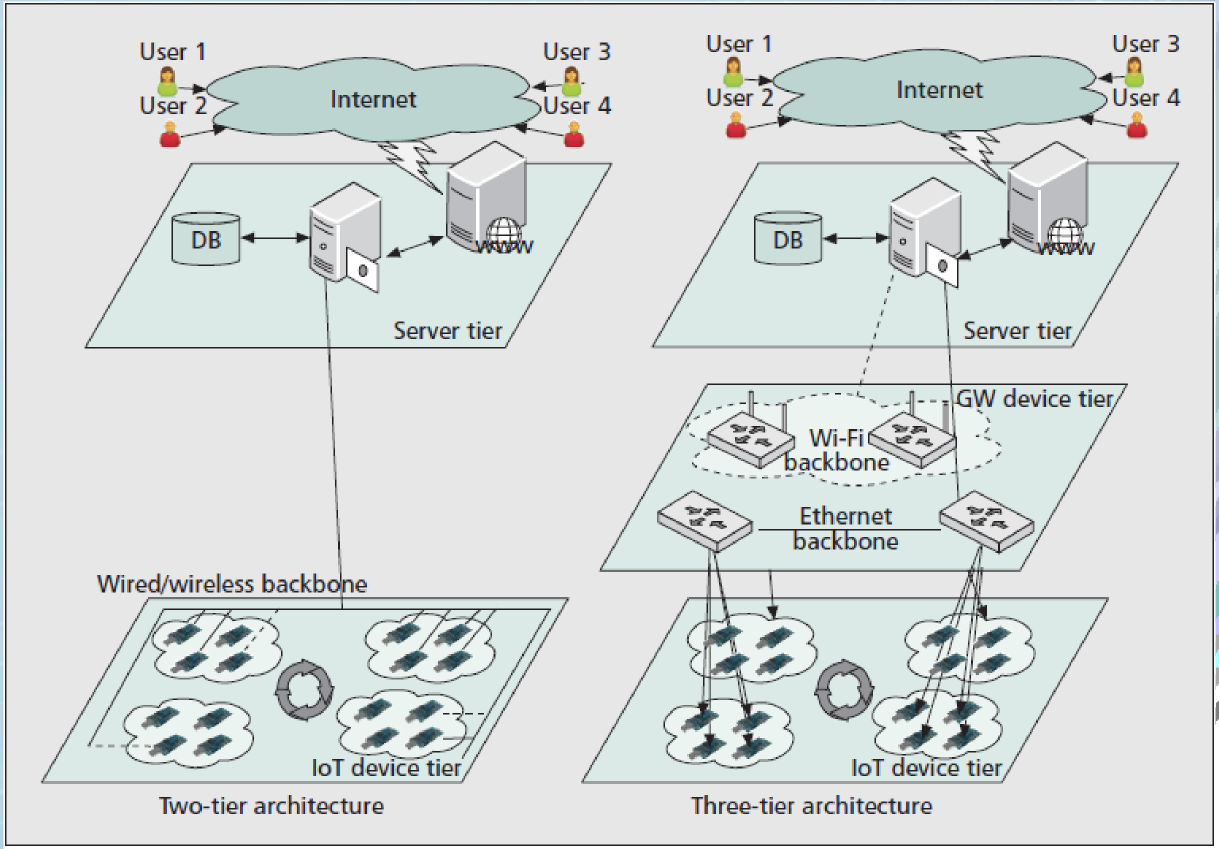


Figure 2. *Typica IoT testbed architectures.*

Source: A Survey on Facilities for Experimental

FACILITIES FOR EXPERIMENTAL IOT RESEARCH

- Existing testbeds (most are WSN)

MoteLab	NetEye	TutorNet	MIRAGE/Intel
WISEBED	FRONTS	DES-Testbed	w-iLab.t Testbed
Senslab	KanseiGeni	TWIST	CitySense
Oulu Smart City	Friedrichshafen	Motescope	FlockLab
METRO real	VizBee	CITC testbed	RFIT Lab
SAP Future Retail Centre			

Source: A Survey on Facilities for Experimental

ISSUES, CHALLENGES AND OPPORTUNITIES

SECURITY, PRIVACY, MOBILITY MANAGEMENT

ANALYTICAL MODELLING

EXPERIMENTAL AND SIMULATION PLATFORMS

ISSUES AND CHALLENGES

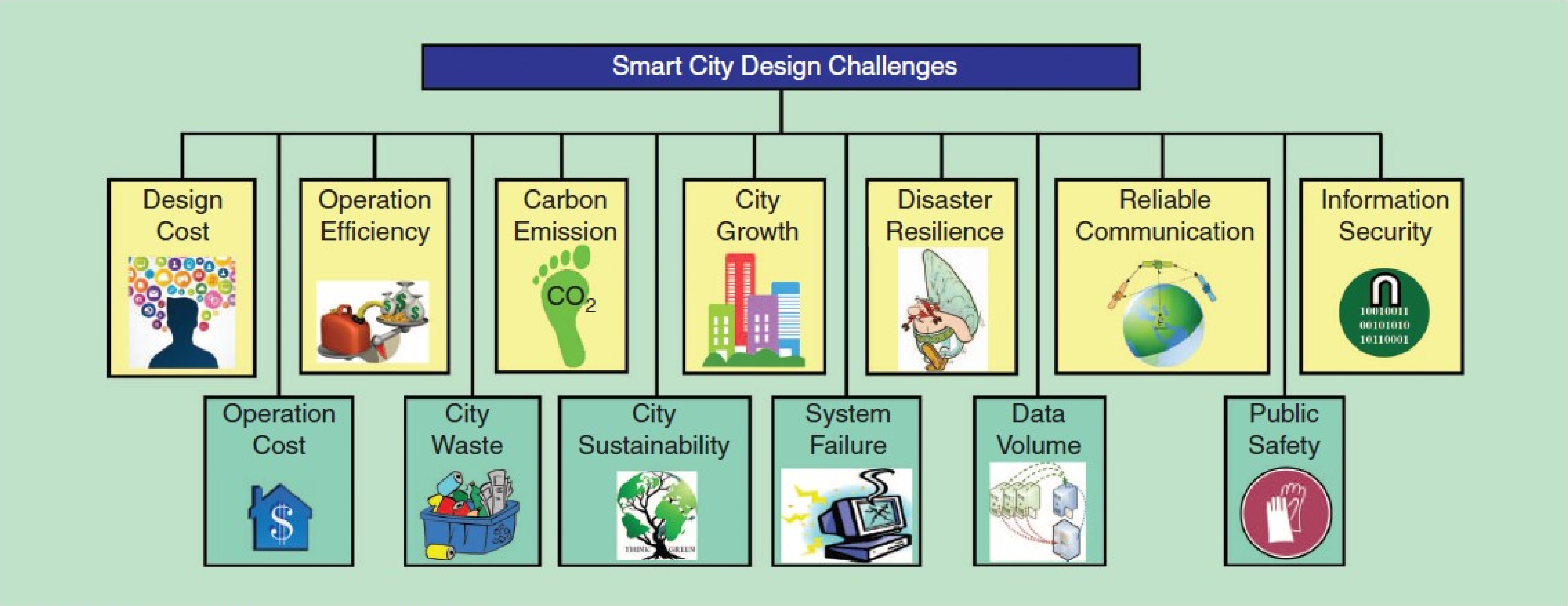


FIGURE 8. Some challenges in smart city design.

Source: Everything You Wanted to Know About Smart Cities: *The Internet*

ISSUES AND CHALLENGES: PRIVACY

- Myriads of sensors constantly register and process our private data
  - our daily commute or our shopping habit
- Seoul: sensors and cameras at every corner monitoring temperature, traffic, electricity
  - "Citizens don't just reject more surveillance, they also slow down the development of smart cities, and the convenience, efficiencies and energy savings they bring"
- "If we have a 'free' service, we pay for it with our privacy"
  - Jarmo Eskelinen, a Finnish data privacy expert

Source: PRIVACY AND SNOOPING IN SMART CITIE

ISSUES AND CHALLENGES: SECURITY

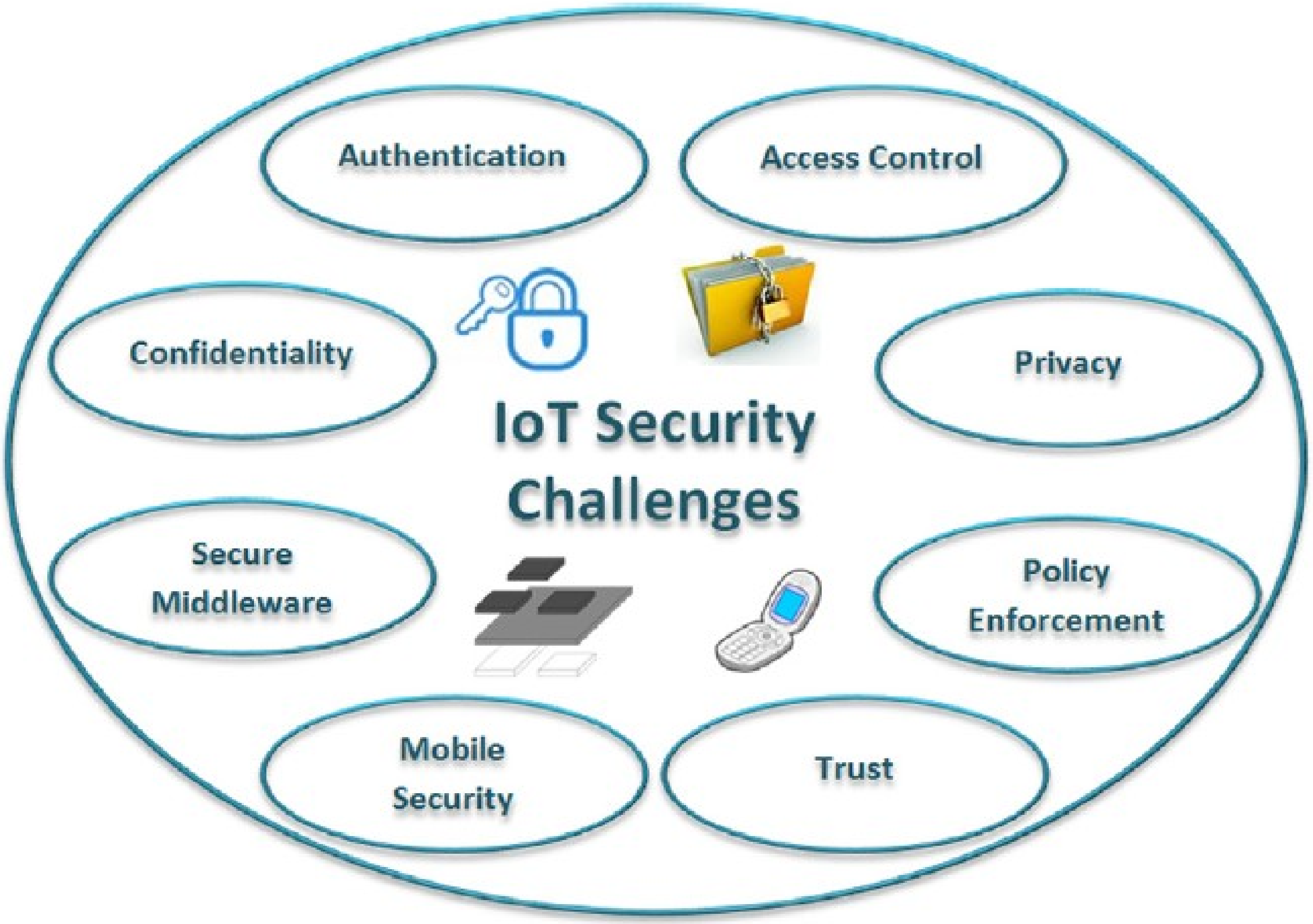


Fig. 1. Main security issues in IoT.

Source: PRIVACY AND SNOOPING IN SMART

ISSUES AND CHALLENGES: MEASUREMENTS AND NETWORK DESIGN

- GPS sensor readings
  - estimate traffic congestion levels OR
  - infer private information about the individual
    - routes users take in their daily commutes, home, and work locations
- Network requirements in SCC applications
  - Much harder than traditional WSN
    - Scheduling sensing and communication tasks
    - Aggregate bandwidth requirements
    - QoS and QoE guarantees
    - Dependability issues: reliability, availability, performability

ISSUES AND CHALLENGES: MODELLING CITIES

- COMPLEXITY THEORY OF CITIES: a popular theory used to explain urban phenomena
  - Complexity Theory: group of theories concerned with complex systems and how they evolve
  - Complex system: elements interact and effect each other
    - difficult to determine which interaction is responsible for each outcome
    - Interactions are interwoven
- Premise: Cities as complex, self-organizing and nonlinear systems
  - Future behavior is not predictable with a top-down approach

Source: The Well-Informed City - A Decentralized, Bottom-Up Model for a Smart City Service Using Infor

ISSUES AND CHALLENGES: REQUIREMENTS FOR MIDDLEWARES

- Challenges Related to Functional Requirements
  - Resource discovery: dynamic and ultra-large-scale
  - Resource management: conflicts in resource allocation among multiple concurrent services or applications
  - Data management: raw data to be converted into knowledge
  - Event management: middleware components may become bottlenecks
  - Code management: reprogrammability, Updates or changes in business logic should be supported by any IoT component

ISSUES AND CHALLENGES: REQUIREMENTS FOR MIDDLEWARES

- Challenges Related to *Nonfunctional* Requirements
  - Scalability
  - Real time: Getting real-time information
  - Dependability: Reliability, Availability
  - Security and privacy
- Challenges Related to Architectural Requirements
  - Programming abstraction
  - Interoperability
  - Context-awareness and autonomous behavior

EXTRA INFO

LIST OF IOT PLATFORMS

Zoom in!!

Table 1
Available IoT platforms.

Ref	Platforms	a) Support of heterogeneous devices	b) Type	c) Architecture	d) Open source	e) REST	f) Data access control	g) Service discovery
1	AirVantage™	Needs gateway	M2M PaaS	Cloud-based	Libraries only (Apache v2, MIT and Eclipse v1.0	Yes	OAuth2	No
2	Arkessa	Yes	M2M PaaS	Cloud-based	No	n.a.	Facebook like privacy settings	No
3	ARM mbed	Embedded devices	M2M PaaS	Centralized/ Cloud-based	No	CoAP	User's choice	No
4	Carriots®	Yes	PaaS	Cloud-based	No	Yes	Secured access	No
5	DeviceCloud	Yes	PaaS	Cloud-based	No	Yes	n.a.	No
7	EveryAware	Yes	Server	Centralized	No	Yes	4 levels	No
8	Everyware	Needs gateway	PaaS	Cloud-based	No	Yes	n.a.	No
9	EvryThng	Yes	M2M SaaS	Centralized	No	Yes	Fine-grained	No
10	Exosite	Yes	PaaS	Cloud-based	Libraries only (BSD license)	Yes	n.a.	No
11	Fosstrack	RFID	Server	Centralized	No	No	Locally stored	No
12	GroveStreams	No	PaaS	Cloud-based	No	Yes	Role-based	No
13	H.A.T.	Home devices	PaaS	Decentralized	Yes	Yes	Locally stored	Yes
14	IoT-framework	Yes	Server	Centralized	Apache license 2.0	Yes	Locally stored	Yes
15	IFTTT	Yes	SaaS	Centralized	No	No	No storage	Limited
16	Kahvihub	Yes	Server	Centralized	Apache license 2.0	Yes	Locally stored	Yes
17	LinkSmart™	Embedded devices	P2P	Decentralized	LGPLv3	No	Locally stored	Yes
18	MyRobots	Robots	Robots PaaS	Cloud-based	No	Yes	2 levels	No
19	Niagara^{AX}	Yes	M2M SaaS	Distributed	No	n.a.	n.a.	n.a.
20	Nimbits	Yes	Server	Centralized/ Cloud-based	Apache license 2.0	Yes	3 levels	No
21	NinjaPlatform	Needs gateway	PaaS	Cloud-based	Open source hardware and Operating System	Yes	OAuth2	No
22	Node-RED	Yes	Server	Centralized	Apache license 2.0	No	User-based privileges	No
23	OpenIoT	Yes	Hub	Decentralized	LGPLv3	No	User-based privileges	Yes
24	OpenMTC	Yes	M2M client/ Server	Centralized/ Cloud-based	No	Yes	Secured access	No
25	OpenRemote	Home devices	Server	Centralized	Affero GNU Public License	Yes	Locally stored	No
26	Open.Sen.se	Ethernet enabled	PaaS/SaaS	Cloud-based	No	Yes	2 levels	Limited
27	realTime.io	Needs gateway	PaaS	Cloud-based	No	Yes	Secured access	No
28	SensorCloud™	No	PaaS	Cloud-based	No	Yes	n.a.	No
29	SkySpark	No	SaaS	Centralized/ Cloud-based	No	Yes	n.a.	No
30	Swarm	Yes	PaaS	Cloud-based	Client is open source (unknown license)	Yes	n.a.	n.a.
31	TempoDB	No	PaaS	Cloud-based	No	Yes	Secured access	No
32	TerraSwarm	Yes	OS	Decentralized	n.a.	n.a.	n.a.	Yes
33	The thing system	Home devices	Server	Centralized	M.I.T.	Yes	User's choice	No
34	Thing Broker	Yes	Server	Centralized	Yes	Yes	Locally stored	No
35	ThingSpeak	Yes	Server	Centralized/ Cloud-based	GNU GPLv3	Yes	2 levels	Limited
36	ThingSquare	Embedded devices	Mesh	Cloud-based	Gateway firmware is open source	Yes	No	No
37	ThingWorx	Yes	M2M PaaS	Cloud-based	No	Yes	User-based privileges	Yes
38	WoTkit	Yes	PaaS	Cloud-based	No	Yes	Secured access	Yes
39	Xively	Yes	PaaS	Cloud-based	Libraries are open source (BSD 3-clause), platform is not	Yes	Secured access	Yes

LIST OF SC PROJECTS AROUND THE WORLD

Zoom in!!

Table 1. Smart city projects around the world.

Project/location	Funding	Duration	Goals	Smart city characteristics	Partners
Yokohama Smart City Project,[a] Japan	Ministry of Economy, Trade and Industry (METI)	2010–2015	Low-carbon city, hierarchical energy management systems (EMS), sensitive photovoltaic (PV) generation	Smart environment, smart living	Tokyo Institute of Technology, Toshiba, Mitsubishi, Hitachi
Smart Mobility & Energy Life in Toyota City,[b] Japan	METI	2010–2015	PV generation, intelligent transportation systems, hierarchical EMS, 61.2% renewable energy, 4,000 next-generation vehicles	Smart mobility, smart environment	Nagoya University, Toyota City, Fujitsu, Hitachi, Toyota Motor Corporation, Chubu Electric Power Co.
Keihanna Eco City Next-Generation Energy and Social Systems project,[c] Japan	METI	2010–2015	Develop community EMS to minimize CO_2 emissions, vehicle-to-infrastructure, and to-vehicle	Smart environment	Kyoto, Kizugawa, Kyotanabe, Fuji Electric, Kyoto Center for Climate Actions, Mitsubishi
Kitakyushu Smart Community Project,[d] Japan	METI	2010–2015	Participation by citizens and companies in the energy-distribution process, PV generation, establishing charging infrastructure, and next-generation traffic systems (bicycles and public transport)	Smart mobility, smart environment	Toyota Motor Corporation, IBM Japan, Japan Telecom Information Service Corporation, Mitsubishi Heavy Industries
CITYKEYS,[e] European Union	H2020 project, European Union	2015–2017	Develop and validate key performance indicators and data-collection procedures for smart cities, sharing best practices on user privacy and other legislative issues among cities	Smart mobility, smart environment, smart living, smart people	Research organizations: VTT (Finland), AIT (Austria), TNO (The Netherlands); and five partner cities: Rotterdam, Tampere, Vienna, Zagreb, Zaragoza
LIVE Singapore project,[f] Singapore	National Research Foundation of Singapore	2011–2016	Develop open platform for collecting, elaborating, and distributing real-time data reflecting urban activities: tracking vehicular traffic and estimated temperature rise, energy consumption, and taxi operations	Smart living, smart people	MIT's SENSEable City Lab, Future Urban Mobility research initiative, Changi Airport Group, ComfortDelGro, NEA, PSA, SP Services, SingTel
SmartSantander,[g] Europe	European Union	2010–2013	Deploy 20,000 sensors in Belgrade, Guildford, Lübeck, and Santander, exploiting multiple technologies to collect information on parking spaces, public transport, and automatic management of light; currently uses 2,000 IEEE 802.15.4 devices	Smart living, smart environment	Telefonica I+D (Spain) Universität zu Lübeck (Germany), Ericsson (Serbia), Alcatel-Lucent (Italy), Alexandra Instituttet A/S (Denmark)
Open Cities project,[h] Europe	European Union	2011–2013	Explore how to implement open and user-driven innovation methodologies in the public sector in European cities, including Amsterdam, Barcelona, Berlin, Bologna, Helsinki, Paris, and Rome	Smart governance	Fraunhofer Institute FOKUS (Germany), ATOS (Spain), ESADE Business School (Spain), Berlin Government Senate Department for Economics, Technology and Women's Issues (Germany), Institut Telecom (France), NESTA (U.K.)
Vehicle2Grid,[i] The Netherlands	Top consortium on Knowledge and Innovation Switch2SmartGrids	2014–2017	Deliver European Open Data repository	Smart mobility, smart environment	Cofely, Alliander, ABB, Mitsubishi Motors Corporation, Amsterdam Smart City, Amsterdam University of Applied Sciences
City Science Initiative,[j] MIT/U.S.	Corporate sponsorship, industrial funding, National Science Foundation, Defense Advanced Research Projects Agency, National Institutes of Health	—	Use batteries in electric cars to store locally produced energy	Smart mobility, smart environment	27 scientific research teams[k]
"Green Vision"[l] initiative, San Jose, CA	State and federal funding	2007–2022	Gain scientific understanding of cities: urban analytics, governance, mobility networks, electronic and social networks, and energy networks Create clean tech jobs, reducing energy use by 50%, generating 100% energy from renewable sources, reusing water, installing zero-emission lighting, and having 100% public vehicles run on alternative fuels	Smart mobility, smart environment	Universities, private companies, regional agencies

a http://www.city.yokohama.lg.jp/ondan/english/yscp/
b http://jscp.nepc.or.jp/article/jscpen/20150528/445244/
c http://jscp.nepc.or.jp/en/keihanna/index.shtml
d http://www.nedo.go.jp/content/100639530.pdf
e http://www.citykeys-project.eu
f http://senseable.mit.edu/livesingapore/
g http://www.smartsantander.eu/
h http://www.opencities.net/content/project
i http://amsterdamsmartcity.com/?lang=en
j http://cities.media.mit.edu/
k http://www.media.mit.edu/research/groups-projects
l http://www.sanjoseca.gov/DocumentCenter/View/42557