```
exceptKeyboardInterrupt:

    exit()
```

Other Devices

1. pcDuino
2. BeagleBoneBlack
3. Cubieboard

UNIT V

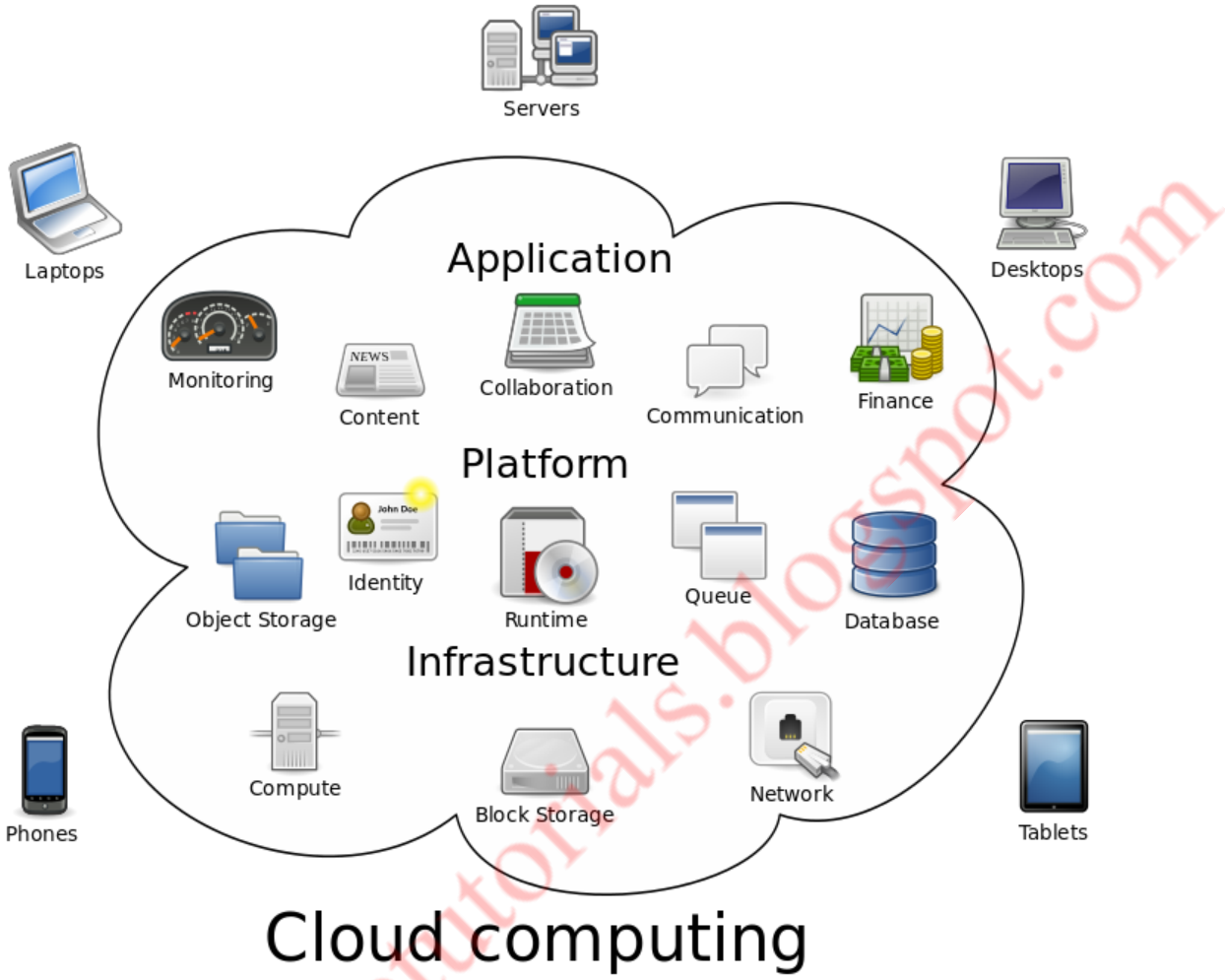
IoT PHYSICAL SERVERS AND CLOUD OFFERINGS

Introduction to Cloud Computing

The Internet of Things (IoT) involves the internet-connected devices we use to perform the processes and services that support our way of life. Another component set to help IoT succeed is cloud computing, which acts as a sort of front end. Cloud computing is an increasingly popular service that offers several advantages to IOT, and is based on the concept of allowing users to perform normal computing tasks using services delivered entirely over the internet. A worker may need to finish a major project that must be submitted to a manager, but perhaps they encounter problems with memory or space constraints on their computing device. Memory and space constraints can be minimized if an application is instead hosted on the internet. The worker can use a cloud computing service to finish their work because the data is managed remotely by a server. Another example: you have a problem with your mobile device and you need to reformat it or reinstall the operating system. You can use Google Photos to upload your photos to internet-based storage. After the reformat or reinstall, you can then either move the photos back to you device or you can view the photos on your device from the internet when youwant.

Concept

In truth, cloud computing and IoT are tightly coupled. The growth of IoT and the rapid development of associated technologiescreate a widespread connection of –things.‖This has lead to the production of large amounts of data, which needs to be stored, processed and accessed. Cloud computing as a paradigm for big data storage and analytics. While IoT is exciting on its own, the real innovation will come from combining it with cloud computing. The combination of cloud computing and IoT will enable new monitoring services and powerful processing of sensory data streams. For example, sensory data can be uploaded and stored with cloud computing, later to be used intelligently for smart monitoring and actuation with other smart devices. Ultimately, the goal is to be able to transform data to insight and drive productive, cost-effective action from those insights. The cloud effectively serves as the brain to improved decision-making and optimized internet-based interactions.However, when IoT meets cloud, new challenges arise. There is an urgent need for novel network architectures that seamlessly integrate them. The critical concerns during integration are quality of service (QoS) and qualityof experience (QoE), as well as data security, privacy and reliability. The virtual infrastructure for practical mobile computing and interfacing includes integrating applications, storage devices, monitoring devices, visualization platforms, analytics tools and client delivery. Cloud computing offers a practical utility-based model that will enable businesses and users to access applications on demand anytime and fromanywhere.



Characteristics

First, the cloud computing of IoT is an on-demand self service, meaning it‘s there when you need it. Cloud computing is a web-based service that can be accessed without any special assistance or permission from other people; however, you need at minimum some sort of internet access.

Second, the cloud computing of IoT involves broad network access, meaning it offers several connectivity options. Cloud computing resources can be accessed through a wide variety of internet-connected devices such as tablets, mobile devices and laptops. This level of convenience means users can access those resources in a wide variety of manners, even from older devices. Again, though, this emphasizes the need for network access points.

Third, cloud computing allows for resource pooling, meaning information can be shared with those who know where and how (have permission) to access the resource, anytime and anywhere. This lends to broader collaboration or closer connections with other users. From an IoT perspective, just as we can easily assign an IP address to every "thing" on theplanet, we can

share the "address" of the cloud-based protected and stored information with others and pool resources.

Fourth, cloud computing features rapid elasticity, meaning users can readily scale the service to their needs. You can easily and quickly edit your software setup, add or remove users, increase storage space, etc. This characteristic will further empower IoT by providing elastic computing power, storage and networking.

Finally, the cloud computing of IoT is a measured service, meaning you get what you pay for. Providers can easily measure usage statistics such as storage, processing, bandwidth and active user accounts inside your cloud instance. This pay per use (PPU) model means your costs scale with your usage. In IoT terms, it's comparable to the ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other internet-enabled devices and systems; just like your cloud service, the service rates for that IoT infrastructure may also scale with use.

Service and Deployment

Service models

Service delivery in cloud computing comprises three different service models: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

Software as a service (SaaS) provides applications to the cloud‘s end user that are mainly accessed via a web portal or service-oriented architecture-based web service technology. These services can be seen as ASP (application service provider) on the application layer. Usually, a specific company that uses the service would run, maintain and give support so that it can be reliably used over a long period of time.

Platform as a service (PaaS) consists of the actual environment for developing and provisioning cloud applications. The main users of this layer are developers that want to develop and run a cloud application for a particular purpose. A proprietary language was supported and provided by the platform (a set of important basic services) to ease communication, monitoring, billing and other aspects such as startup as well as to ensure an application‘s scalability and flexibility. Limitations regarding the programming languages supported, the programming model, the ability to access resources, and the long-term persistence are possibledisadvantages.

Infrastructure as a service (IaaS) provides the necessary hardware and software upon which a customer can build a customized computing environment. Computing resources, data storage resources and the communications channel are linked together with these essential IT resources to ensure the stability of applications being used on the cloud. Those stack models can be referred to as the medium for IoT, being used and conveyed by the users in different methods for the greatest chance of interoperability. This includes connecting cars, wearables, TVs, smartphones, fitness equipment, robots, ATMs, and vending machines as well as the vertical applications, security and professional services, and analytics platforms that come withthem.

Deployment models
Deployment in cloud computing comprises four deployment models: private cloud, public cloud, community cloud and hybrid cloud.

A private cloud has infrastructure that‘s provisioned for exclusive use by a single organization comprising multiple consumers such as business units. It may be owned, managed and operated by the organization, a third party or some combination of them, and it may exist on or off premises.

A public cloud is created for open use by the general public. Public cloud sells services to anyone on the internet. (Amazon Web Services is an example of a large public cloud provider.) This model is suitable for business requirements that require management of load spikes and the applications used by the business, activities that would otherwise require greater investment in infrastructure for the business. As such, public cloud also helps reduce capital expenditure and bring down operational ITcosts.

A community cloud is managed and used by a particular group or organizations that have shared interests, such as specific security requirements or a common mission.
Finally, a hybrid cloud combines two or more distinct private, community or public cloud infrastructures such that they remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. Normally, information that‘s not critical is outsourced to the public cloud, while business-critical services and data are kept within the control of the organization.

CLOUD STORAGE API

A cloud storage API is an application program interface that connects a locally-based application to a cloud-based storage system, so that a user can send data to it and access and work with data stored in it. To the application, the cloud storage system is just another target device, like tape or disk-based storage. An application program interface (API) is code that allows two software programs to communicate with each other. The API defines the correct way for a developer to write a program that requests services from an operating system (OS) or other application. APIs are implemented by function calls composed of verbs and nouns. The required syntax is described in the documentation of the application beingcalled.

How APIs work
APIs are made up of two related elements. The first is a specification that describes how information is exchanged between programs, done in the form of a request for processing and a return of the necessary data. The second is a software interface written to that specification and published in some way for use.The software that wants to access the features and capabilities of the API is said to call it, and the software that creates the API is said to publish it.

Why APIs are important for business
The web, software designed exchange information via the internet and cloud computinghave all combined to increase the interest in APIs in general and services in particular.Software that was

once custom-developed for a specific purpose is now often written referencing APIs that provide broadly useful features, reducing development time and cost and mitigating the risk of errors.APIs have steadily improved software quality over the last decade, and the growing number of web services exposed through APIs by cloud providers is also encouraging the creation of cloud-specific applications, internet of things (IoT) efforts and apps to support mobile devices and users.

Three basic types of APIs
APIs take three basic forms: local, web-like and program-like.

1. **Local APIs** are the original form, from which the name came. They offer OS or middleware services to application programs. Microsoft's .NET APIs, the TAPI (Telephony API) for voice applications, and database access APIs are examples of the local APIform.

2. **Web APIs** are designed to represent widely used resources like HTML pages and are accessed using a simple HTTP protocol. Any web URL activates a web API. Web APIs are often called REST (representational state transfer) or RESTful because the publisher of REST interfaces doesn't save any data internally between requests. As such, requests from many users can be intermingled as they would be on theinternet.

3. **Program APIs** are based on remote procedure call (RPC) technology that makes a remote program component appear to be local to the rest of the software. Service oriented architecture (SOA) APIs, such as Microsoft's WS-series of APIs, are programAPIs.

IoT / Cloud Convergence
Internet-of-Things can benefit from the scalability, performance and pay-as-you-go nature of cloud computing infrastructures. Indeed, as IoT applications produce large volumes of data and comprise multiple computational components (e.g., data processing and analytics algorithms), their integration with cloud computing infrastructures could provide them with opportunities for cost-effective on-demand scaling. As prominent examples consider the following settings:

A Small Medium Enterprise (SME) developing an energy management IoT product, targeting smart homes and smart buildings. By streaming the data of the product (e.g., sensors and WSN data) into the cloud it can accommodate its growth needs in a scalable and cost effective fashion. As the SMEs acquires more customers and performs more deployments of its product, it is able tocollectandmanagegrowingvolumesofdatainascalableway,thustakingadvantageofa –pay-as-you-grow‖model. Moreover, cloud integration allows the SME to store and processmassive datasets collected from multiple (rather than a single)deployments.

A smart city can benefit from the cloud-based deployment of its IoT systems and applications. A city is likely to deploy many IoT applications, such as applications for smart energy management, smart water management, smart transport management, urban mobility of the citizensandmore.Theseapplicationscomprisemultiplesensorsanddevices,alongwith

computational components. Furthermore, they are likely to produce very large data volumes. Cloud integration enables the city to host these data and applications in a cost-effective way. Furthermore, the elasticity of the cloud can directly support expansions to these applications, but also the rapid deployment of new ones without major concerns about the provisioning of the required cloud computing resources.

A cloud computing provider offering pubic cloud services can extend them to the IoT area, through enabling third-parties to access its infrastructure in order to integrate IoT data and/or computational components operating over IoT devices. The provider can offer IoT data access and services in a pay-as-you-fashion, through enabling third-parties to access resources of its infrastructure and accordingly to charge them in a utility-based fashion.

These motivating examples illustrate the merit and need for converging IoT and cloud computing infrastructure. Despite these merits, this convergence has always been challenging mainly due to the conflicting properties of IoT and cloud infrastructures, in particular, IoT devices tend to be location specific, resource constrained, expensive (in terms of development/ deployment cost) and generally inflexible (in terms of resource access and availability). On the other hand, cloud computing resources are typically location independent and inexpensive, while at the same time providing rapid and flexibly elasticity. In order to alleviate these incompatibilities, sensors and devices are virtualized prior to integrating their data and services in the cloud, in order to enable their distribution across any cloud resources. Furthermore, service and sensor discovery functionalities are implementing on the cloud in order to enable the discovery of services and sensors that reside in different locations.

Based on these principles the IoT/cloud convergence efforts have started since over a decade i.e. since they very early days of IoT and cloud computing. Early efforts in the research community (i.e. during 2005-2009) have focused on streaming sensor and WSN data in a cloud infrastructure. Since 2007 we have also witnessed the emergence of public IoT clouds, including commercial efforts. One of the earliest efforts has been the famous Pachube.com infrastructure (used extensively for radiation detection and production of radiation maps during earthquakes in Japan). Pachube.com has evolved (following several evolutions and acquisitions of this infrastructure) to Xively.com, which is nowadays one of the most prominent public IoT clouds. Nevertheless, there are tens of other public IoT clouds as well, such as ThingsWorx, ThingsSpeak,Sensor-Cloud,Realtime.io and more. The list is certainly non-exhaustive. These public IoT clouds offer commercial pay-as-you-go access to end-users wishing to deploying IoT applications on the cloud. Most of them come with developer friendly tools, which enable the development of cloud applications, thus acting like a PaaS for IoT in the cloud. Similarly to cloud computing infrastructures, IoT/cloud infrastructures and related services can be classified to the followingmodels:

1. **Infrastructure-as-a-Service (IaaS) IoT/Clouds**: These services provide the means for accessing sensors and actuator in the cloud. The associated business model involves the IoT/Cloud provide to act either as data or sensor provider. IaaS services for IoT provide access control to resources as a prerequisite for the offering of related pay-as-you-go services.

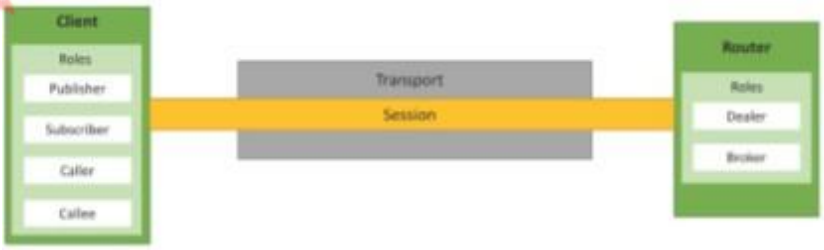
2. **Platform-as-a-Service (PaaS) IoT/Clouds**: This is the most widespread model for IoT/cloud services, given that it is the model provided by all public IoT/cloud infrastructures outlined above. As already illustrate most public IoT clouds come with a range of tools and related environments for applications development and deployment in a cloud environment. A main characteristic of PaaS IoT services is that they provide access to data, not to hardware. This is a clear differentiator comparing toIaaS.

3. **Software-as-a-Service (SaaS) IoT/Clouds**: SaaS IoT services are the ones enabling their uses to access complete IoT-based software applications through the cloud, on-demand and in a pay-as-you-go fashion. As soon as sensors and IoT devices are not visible, SaaS IoT applications resemble very much conventional cloud-based SaaS applications. There are however cases where the IoT dimension is strong and evident, such as applications involving selection of sensors and combination of data from the selected sensors in an integrated applications. Several of these applications are commonly called Sensing-as-a-Service, given that they provide on-demand access to the services of multiple sensors. Note that SaaS IoT applications are typically built over a PaaS infrastructure and enable utility based business models involving IoT software andservices.

These definitions and examples provide an overview of IoT and cloud convergence and why it is important and useful. More and more IoT applications are nowadays integrated with the cloud in order to benefit from its performance, business agility and pay-as-you-go characteristics. In following chapters of the tutorial, we will present how to maximize the benefits of the cloud for IoT, through ensuring semantic interoperability of IoT data and services in the cloud, thus enabling advanced data analytics applications, but also integration of a wide range of vertical (silo) IoT applications that are nowadays available in areas such as smart energy, smart transport and smart cities. We will also illustrate the benefits of IoT/cloud integration for specific areas and segments of IoT, such as IoT-based wearablecomputing.

WAMP for IoT

Web Application Messaging Protocol (WAMP) is a sub-protocol of Websocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.



WAMP

1. **Transport:** Transport is channel that connects two peers.
2. **Session:** Session is a conversation between two peers that runs over atransport.
3. **Client:** Clients are peers that can have one or more roles. In publish-subscribe model client can have followingroles:
    a) Publisher: Publisher publishes events (including payload) to the topic maintained by thebroker.
    b) Subscriber: Subscriber subscribes to the topics and receives the events including the payload.

In RPC model client can have following roles: –

1. **Caller:** Caller issues calls to the remote procedures along with call arguments. – Callee: Callee executes the procedures to which the calls are issued by the caller and returns the results back to the caller. • Router: Routers are peers that perform generic call and event routing. In publish-subscribe model Router has the role of a Broker: – Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to thetopic.

In RPC model Router has the role of a Broker: –

1. Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from Callee toCaller.
2. Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

Amazon EC2 – Python Example

Boto is a Python package that provides interfaces to Amazon Web Services (AWS). In this example, a connection to EC2 service is fi rst established by calling boto.ec2.connect_to_region. The EC2 region, AWS access key and AWS secret key are passed to this function. After connecting to EC2 , a new instance is launched using the conn.run_instances function. The AMI-ID, instance type, EC2 key handle and security group are passed to this function.

```
#Python program for launching an EC2 instance
import boto.ec2
from time import sleep
ACCESS_KEY="<enter access key>"
SECRET_KEY="<enter secret key>"

REGION="us-east-1"
AMI_ID = "ami-d0f89fb9"
EC2_KEY_HANDLE = "<enter key handle>"
INSTANCE_TYPE="t1.micro"
SECGROUP_HANDLE="default"

conn = boto.ec2.connect_to_region(REGION, aws_access_key_id=ACCESS_KEY,
                                  aws_secret_access_key=SECRET_KEY)

reservation = conn.run_instances(image_id=AMI_ID, key_name=EC2_KEY_HANDLE,
                                 instance_type=INSTANCE_TYPE,
                                 security_groups = [ SECGROUP_HANDLE, ] )
```

Amazon AutoScaling – Python Example

1. **AutoScaling Service:** A connection to AutoScaling service is first established by calling boto.ec2.autoscale.connect_to_regionfunction.
2. **Launch Configuration**: After connecting to AutoScaling service, a new launch configuration is created by calling conn.create_launch_con f iguration. Launch configuration contains instructions on how to launch new instances including the AMI-ID, instance type, security groups,etc.
3. **AutoScaling Group** : After creating a launch configuration, it is then associated with a new AutoScaling group. AutoScaling group is created by calling conn.create_auto_scaling_group. The settings for AutoScaling group such as the maximum and minimum number of instances in the group, the launch configuration, availability zones, optional load balancer to use with the group,etc.

```
#Python program for creating an AutoScaling group (code excerpt)
import boto.ec2.autoscale
:
print "Connecting to Autoscaling Service"
conn = boto.ec2.autoscale.connect_to_region(REGION,
  aws_access_key_id=ACCESS_KEY,
  aws_secret_access_key=SECRET_KEY)

print "Creating launch configuration"

lc = LaunchConfiguration(name='My-Launch-Config-2',
                         image_id=AMI_ID,
                         key_name=EC2_KEY_HANDLE,
                         instance_type=INSTANCE_TYPE,
                         security_groups = [ SECGROUP_HANDLE, ])
conn.create_launch_configuration(lc)

print "Creating auto-scaling group"

ag = AutoScalingGroup(group_name='My-Group',
              availability_zones=['us-east-1b'],
              launch_config=lc, min_size=1, max_size=2,
              connection=conn)
conn.create_auto_scaling_group(ag)
```

Amazon AutoScaling – Python Example

```
#Creating auto-scaling policies
scale_up_policy = ScalingPolicy(name='scale_up',
        adjustment_type='ChangeInCapacity',
        as_name='My-Group',
        scaling_adjustment=1,
        cooldown=180)
scale_down_policy =ScalingPolicy(name='scale_down',
        adjustment_type='ChangeInCapacity',
        as_name='My-Group', scaling_adjustment=-1,
        cooldown=180)
conn.create_scaling_policy(scale_up_policy)
conn.create_scaling_policy(scale_down_policy)
```

AutoScaling Policies:

1. After creating an AutoScaling group, the policies for scaling up and scaling down are defined.
2. In this example, a scale up policy with adjustment type ChangeInCapacity and scaling_ad justment = 1 isdefined.
3. Similarly a scale down policy with adjustment type ChangeInCapacity and scaling_ad justment = -1 isdefined.

CloudWatch Alarms

```
#Connecting to CloudWatch
cloudwatch = boto.ec2.cloudwatch.connect_to_region(REGION,
                aws_access_key_id=ACCESS_KEY,
```

```
                    aws_secret_access_key=SECRET_KEY)
alarm_dimensions = {"AutoScalingGroupName": 'My-Group'}
#Creating scale-up alarm
scale_up_alarm = MetricAlarm(
        name='scale_up_on_cpu', namespace='AWS/EC2',
        metric='CPUUtilization', statistic='Average',
        comparison='>', threshold='70',
        period='60', evaluation_periods=2,
alarm_actions=[scale_up_policy.policy_arn],
dimensions=alarm_dimensions)
cloudwatch.create_alarm(scale_up_alarm)
#Creating scale-down alarm
scale_down_alarm =MetricAlarm(
    name='scale_down_on_cpu',namespace='AWS/EC2',
    metric='CPUUtilization', statistic='Average',
    comparison='<',threshold='40',
    period='60', evaluation_periods=2,
    alarm_actions=[scale_down_policy.policy_arn],
    dimensions=alarm_dimensions) cloudwatch.create_alarm(scale_down_alarm)
```

1. With the scaling policies defined, the next step is to create Amazon CloudWatch alarms that trigger thesepolicies.
2. The scale up alarm is defined using the CPUUtilization metric with the Average statistic and threshold greater 70% for a period of 60 sec. The scale up policy created previously is associated with this alarm. This alarm is triggered when the average CPU utilization of the instances in the group becomes greater than 70% for more than 60seconds.
3. The scale down alarm is defined in a similar manner with a threshold less than50%.

Python for MapReduce

#Inverted Index Mapper in Python

#!/usr/bin/env python import sys for line in sys.stdin: doc_id, content = line.split(‘‘) words = content.split() for word in words: print ‘%s%s‘ % (word, doc_id)

The example shows inverted index mapper program. The map function reads the data from the standard input (stdin) and splits the tab-limited data into document-ID and contents of the document. The map function emits key-value pairs where key is each word in the document and value is the document-ID.

Python for MapReduce

#Inverted Index Reducer in Python

#!/usr/bin/env python import sys current_word = None current_docids = [] word = None

for line in sys.stdin: # remove leading and trailing whitespace line = line.strip() # parse the input we got from mapper.py word, doc_id = line.split(‘‘) if current_word == word: current_docids.append(doc_id) else: if current_word: print ‘%s%s‘ % (current_word, current_docids) current_docids = [] current_docids.append(doc_id) current_word = word

The example shows inverted index reducer program. The key-value pairs emitted by the map phase are shuffled to the reducers and grouped by the key. The reducer reads the key-value pairs grouped by the same key from the standard input (stdin) and creates a list of document-IDs in which the word occurs. The output of reducer contains key value pairs where key is a unique word and value is the list of document-IDs in which the word occurs.

Python Packages of Interest

1. **JSON:** JavaScript Object Notation (JSON) is an easy to read and write data-interchange format. JSON is used as an alternative to XML and is is easy for machines to parse and generate. JSON is built on two structures - a collection of name-value pairs (e.g. a Python dictionary) and ordered lists of values (e.g.. a Pythonlist).
2. **XML:** XML (Extensible Markup Language) is a data format for structured document interchange. The Python minidom library provides a minimal implementation of the Document Object Model interface and has an API similar to that in otherlanguages.
3. **HTTPLib & URLLib:** HTTPLib2 and URLLib2 are Python libraries used in network/internetprogramming
4. **SMTPLib:** Simple Mail Transfer Protocol (SMTP) is a protocol which handles sending email and routing e-mail between mail servers. The Python smtplib module provides an SMTP client session object that can be used to send email.
5. **NumPy:**NumPy is a package for scientific computing in Python. NumPy provides support for large multi-dimensional arrays andmatrices
6. **Scikit-learn:** Scikit-learn is an open source machine learning library for Python that provides implementations of various machine learning algorithms for classification, clustering, regression and dimension reduction problems.

Python Web Application Framework - Django

Django is an open source web application framework for developing web applications in Python. A web application framework in general is a collection of solutions, packages and best practices that allows development of web applications and dynamic websites. Django is based on the Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface. Django provides a unified API to a database backend. Thus web applications built with Django can work with different databases without requiring any code changes. With this fiexibility in web application design combined with thepowerfulcapabilitiesofthePythonlanguageandthePythonecosystem,

Django is best suited for cloud applications. Django consists of an object-relational mapper, a web templating system and a regular-expressionbased URL dispatcher.

Django Architecture

Django is Model-Template-View (MTV) framework.

1. Model: The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relational database, an XML file, etc. A Django model is a Python class that outlines the variables and methods for a particular type ofdata.
2. Template: In a typical Django web application, the template is simply an HTML page with a few extra placeholders. Django‘s template language can be used to create various forms of text files (XML, email, CSS, Javascript, CSV,etc.)
3. View :The view ties the model to the template. The view is where you write the code that actually generates the web pages. View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

Case studies illustrating IoT design

Case Study in IoT: Home Automation

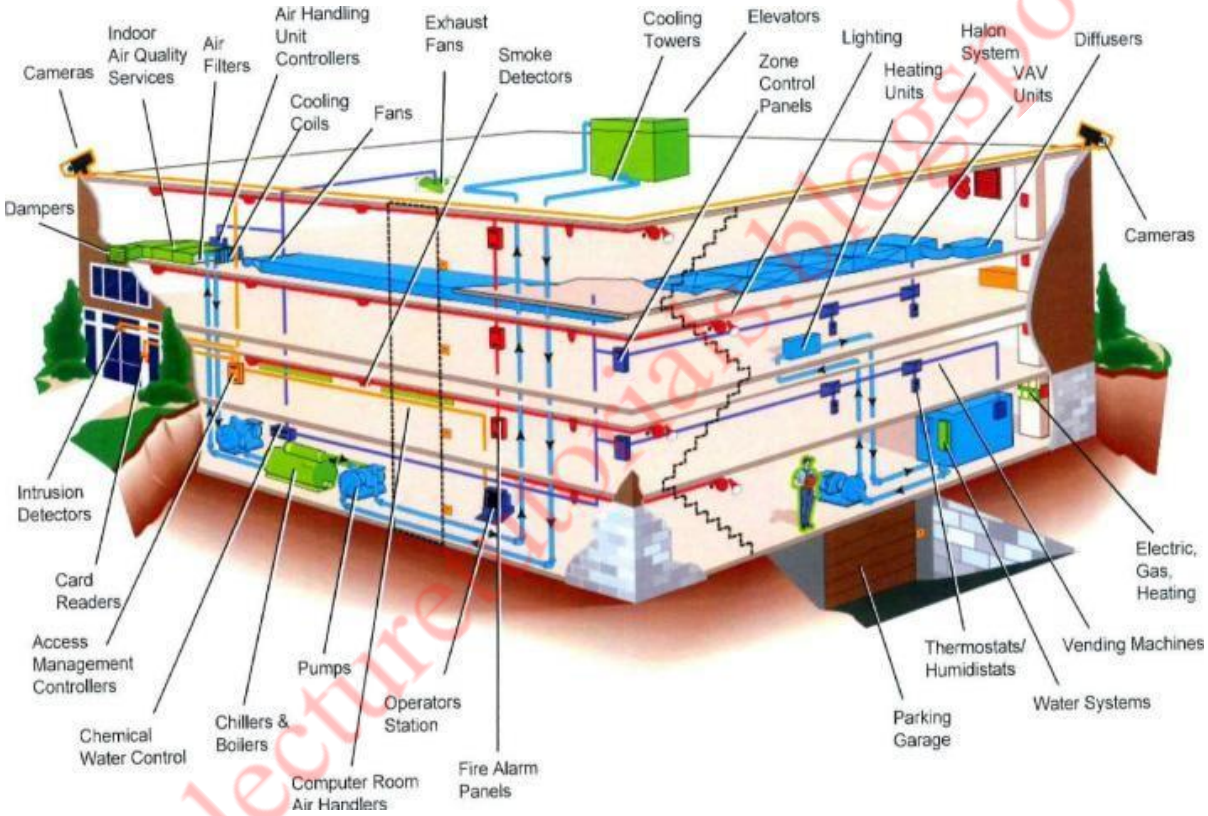
An IoT software-based approach on the field of Home Automation. Common use-cases include measuring home conditions, controlling home appliances and controlling home access through RFID cards as an example and windows through servo locks. However, the main focus of this paper is to maximize the security of homes through IoT. More specifically, monitoring and controlling servo door locks, door sensors, surveillance cameras, surveillance car and smoke detectors, which help ensuring and maximizing safety and security ofhomes.

A user has the following features through a mobile application in which he/she:

1. can turn on or o_ LED lights and monitor the state of theLED.
2. can lock and unlock doors through servo motors and monitor if the doors are locked orunlocked.
3. can monitor if the doors are closed or opened through IR sensors.

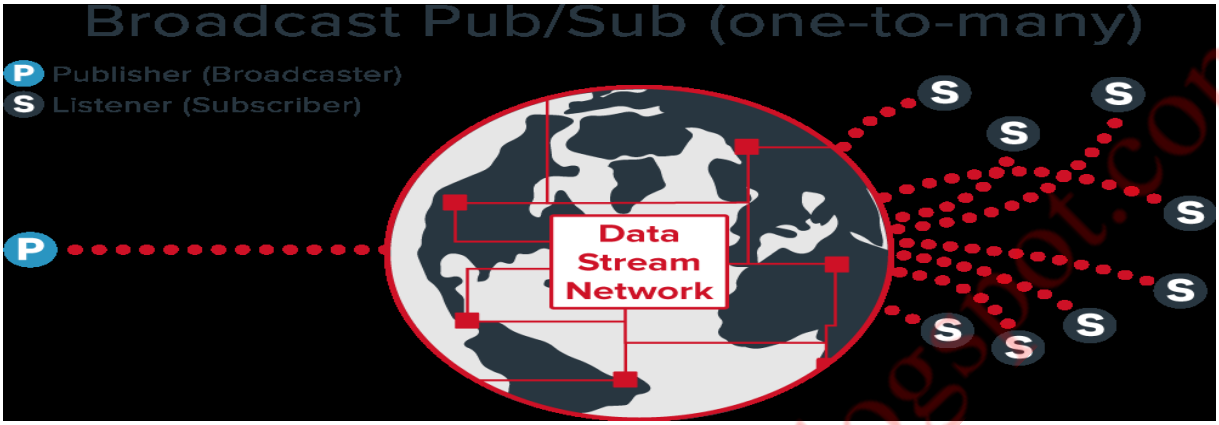
4. is notified through email if the door is left open for toolong.
5. is notified of who entered through the door as the camera captures the faceimage and send it to him/her viaemail.
6. is notified through email if the _re detector detectssmoke.
7. is able to control the surveillance car from anywhere to monitor his/herhome.

As the field of Home Automation through IoT is a wide application in a very wide and challenging field due to the reasons mentioned in the previous paragraphs, I chose to work on that field as part of this thesis, specifically in maintaining and ensuring security and safety inside home.

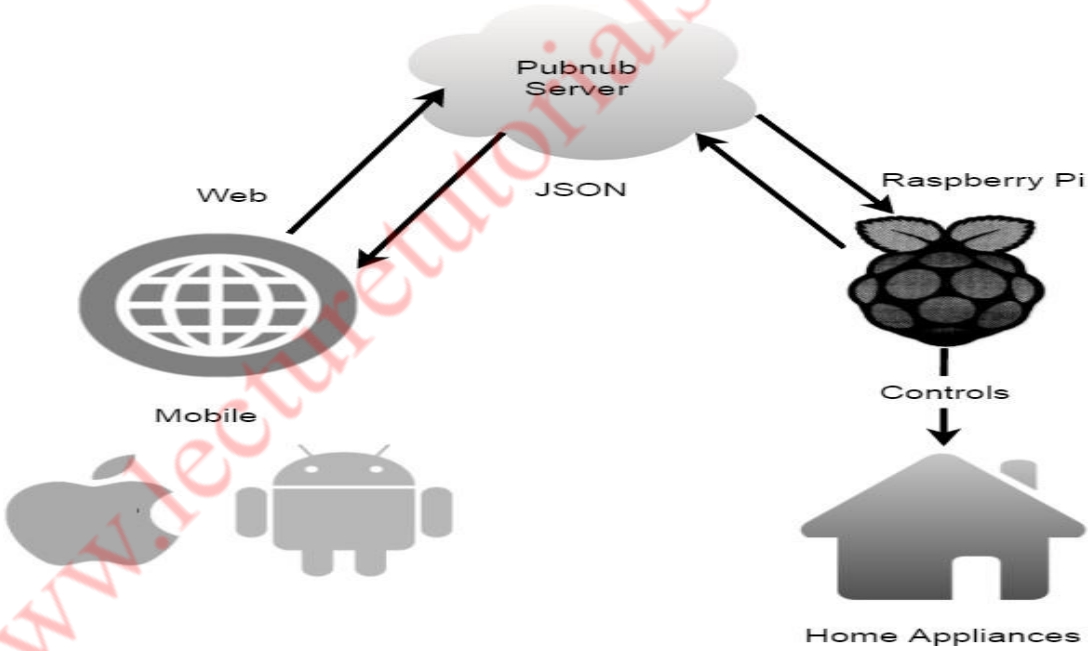


IoT aims in creating a network between objects embedded with sensors, that canstore, analyze, communicate and exchange data together over the internet. This leads to efficient industry, manufacturing, efficient energy management, resource management, accurate health care, smarter business decisions based on analyzed data, safer driving through smart cars that are able to communicate together, smart home automation and countless moreapplications.

The system designed for the home automation project presented in this paper needs a control unit, a computer, to be able to control the different electrical devices connected to it. Raspberry Pi, is a credit-card tiny computer, that can be plugged to a monitor, uses standard keyboard and mouse, that enables people of different ages learn how toprogram.



Illustrates the publish/subscribe model provided by PubNub



Illustrates the system architecture used in this home automation project.

To simplify the publish/subscribe model along with the system architecture used in this Home Automation project, here is the explanation of the steps of constructing it: Different sensors,

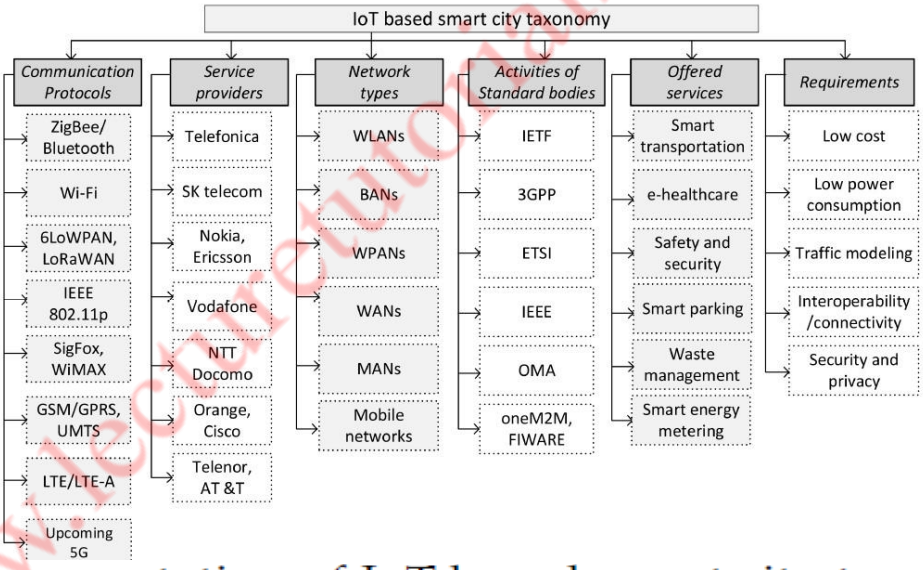
cameras and servo motors were connected to the Raspberry Pi. It was programmed to collect and publish the data, in the form of JSON string, acquired from these devices to PubNub. Data is published from the Raspberry Pi by providing it with the "publish key" and the "channel name". The data is sent to the channel provided by PubNub servers, and forwarded by PubNub to the subscribers of thischannel.

The subscriber in this scenario, of a user acquiring data and readings by the sensors and monitoring devices, is the web/mobile application. The "subscription key" and "channel name" is embedded in the web/mobile application's code. Allowing it to receive messages forwarded by PubNub. On the other hand, in a scenario where the user wants to send a command to home appliances, controlling the LED lights for example, the web/mobile application is the publisher provided by the "publish key" and the "channel name".The commend is sent in the form of JSON string to PubNub servers, while the "subscription key" and "channel name" is embedded in the Raspberry Pi code. This allows the Raspberry Pi to receive any published strings on the channel it is subscribed to. Upon receiving the JSON string, the Raspberry Pi take the action specified by that string. This allows full control and monitoring of all devices connected to the Raspberry Pi by theuser.

Case Study in IoT: Smart Cities

The Internet-of-Things (IoT) is the novel cutting-edge technology which proffers to connect plethora of digital devices endowed with several sensing, actuation and computing capabilities with the Internet, thus offers manifold new services in the context of a smart city. The appealing IoT services and big data analytics are enabling smart city initiatives all over the world. These services are transforming cities by improving infrastructure, transportation systems, reduced traffic congestion, waste management and the quality of human life. In this paper, we devise a taxonomy to best bring forth a generic overview of IoT paradigm for smart cities, integrated information and communication technologies (ICT), network types, possible opportunities and major requirements. Moreover, an overview of the up-to-date efforts from standard bodies is presented. Later, we give an overview of existing open source IoT platforms for realizing smart city applications followed by several exemplary case studies. In addition, we summarize the latest synergies and initiatives worldwide taken to promote IoT in the context of smart cities. Finally, we highlight several challenges in order to give future researchdirections.

An illustration of IoT based smart city



A representation of IoT-based smart city taxonomy

IOT BASED SMART CITY TAXONOMY

This section presents a taxonomy of IoT based smart cities which categorizes the literature on the basis of existing communication protocols, major service providers, network types, standardization efforts, offered services, and crucial requirements.

Communication Protocols

IoT based smart city realization significantly relies on numerous short and wide range communication protocols to transport data between devices and backend servers. Most prominent short range wireless technologies include Zig-Bee, Bluetooth, Wi-Fi, Wireless Metropolitan Area Network (WiMAX) and IEEE 802.11p which are primarily used in smart metering, e-healthcare and vehicular communication. Wide range technologies such as Global System for Mobile communication (GSM) and GPRS, Long-Term Evolution (LTE), LTE-Advanced are commonly utilized in ITS such as vehicle-to infrastructure (V2I), mobile e-healthcare, smart grid and infotainment services. Additionally, LTE-M is considered as an evolution for cellular IoT (C-IoT). In Release 13, 3GPP plans to further improve coverage, battery lifetime as well as device complexity [7]. Besides well-known existing protocols, LoRa alliance standardizes the LoRaWAN protocol to support smart city applications to primarily ensure interoperability between several operators. Moreover, SIGFOX is an ultra narrowband radio technology with full star-based infrastructure offers a high scalable global network for realizing smart city applications with extremely low power consumption. A comparative summary2 of the major communicationprotocols.

Service Providers

Pike Research on smart cities estimated this market will grow to hundreds of billion dollars by 2020, with an annual growth of nearly 16 billion. IoT is recognized as a potential source to increase revenue of service providers. Thus, well-known worldwide service providers have already started exploring this novel cutting edge communication paradigm. Major service providers include Telefonica, SK telecom, Nokia, Ericsson, Vodafone, NTT Docomo, Orange, Telenor group and AT&T which offer variety of services and platforms for smart city applications such as ITS and logistics, smart metering, home automation and e-healthcare.

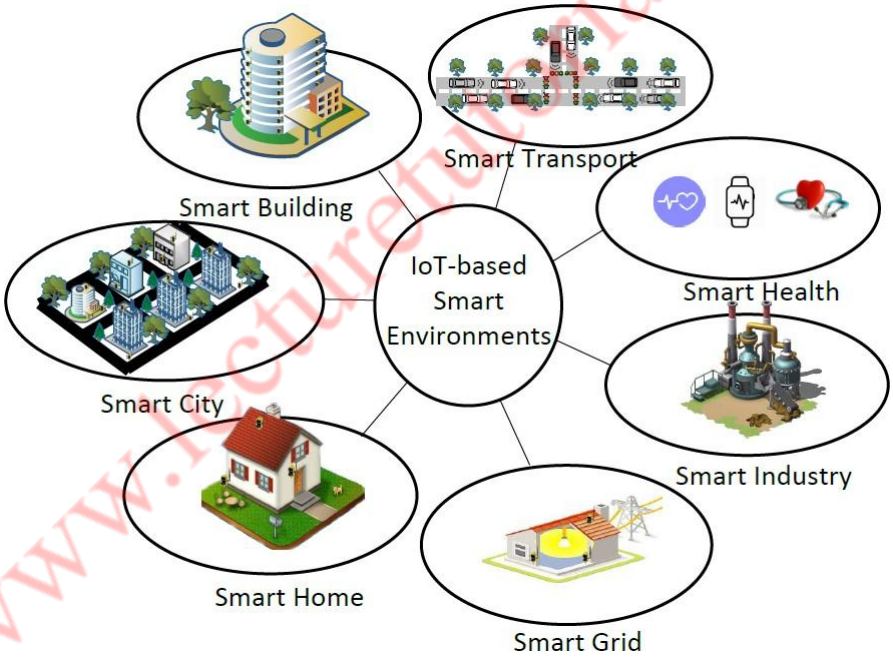
Network Types

IoT based smart city applications rely on numerous network topologies to accomplish a fully autonomous environment. The capillary IoT networks offer services over a short range. Examples include wireless local area networks (WLANs), BANs and wireless personal area networks (WPANs). The application areas include indoor e-healthcare services, home automation, street lighting. On the other hand, applications such as ITS, mobile e-healthcare and waste management use wide area networks (WANs), metropolitan area networks (MANs),and

mobile communication networks. The above networks pose distinct features in terms of data, size, coverage, latency requirements, and capacity.

Case Study in IoT: Smart Environment

The rapid advancements in communication technologies and the explosive growth of Internet of Things (IoT) have enabled the physical world to invisibly interweave with actuators, sensors, and other computational elements while maintaining continuous network connectivity. The continuously connected physical world with computational elements forms a smart environment. A smart environment aims to support and enhance the abilities of its dwellers in executing their tasks, such as navigating through unfamiliar space and moving heavy objects for the elderly, to name a few. Researchers have conducted a number of efforts to use IoT to facilitate our lives and to investigate the effect of IoTbased smart environments on human life. This paper surveys the state-of-the-art research efforts to enable the IoT-based smart environments. We categorize and classify the literature by devising a taxonomy based on communication enablers, network types, technologies, local area wireless standards, objectives, and characteristics. Moreover, the paper highlights the unprecedented opportunities brought about by IoT-based smart environments and their effect on human life. Some reported case studies from different enterprises are also presented. Finally, we discuss open research challenges for enabling IoT-based smart environments.

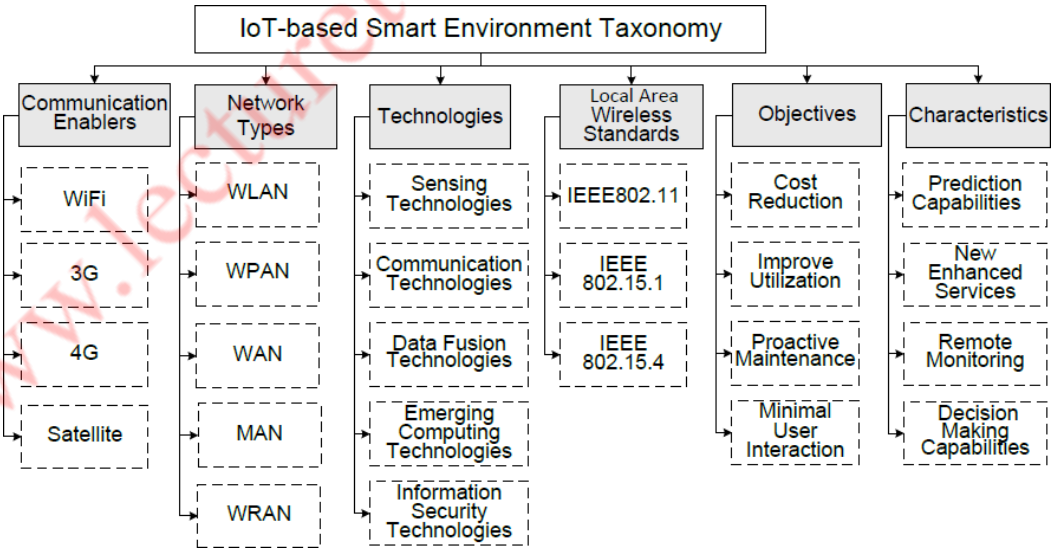


IoT-based Smart Environments

Immense developments and increasing miniaturization of computer technology have enabled tiny sensors and processors to be integrated into everyday objects. This advancement is further supported by tremendous developments in areas such as portable appliances and devices, pervasive computing, wireless sensor networking, wireless mobile communications, machine learning-based decision making, IPv6 support, human computer interfaces, and agent technologies to make the dream of smart environment a reality. A smart environment is a connected small world where sensor-enabled connected devices work collaboratively to make the lives of dwellers comfortable. The term smart refers to the ability to autonomouslyobtain and applies knowledge; and the term environment refers to the surroundings. Therefore, a smart environment is one that is capable of obtaining knowledge and applying it to adapt according to its inhabitants‘ needs to ameliorate their experience of thatenvironment.

The functional capabilities of smart objects are further enhanced by interconnecting them with other objects using different wireless technologies. In this context, IPv6 plays a vital role because of several features, including better security mechanisms, scalability in case of billion of connected devices, and the elimination of NAT barriers1. This concept of connecting smart objects with the Internet was first coined by Kevin Ashton as ―Internet of Things‖ (IoT).

Nowadays, IoT is receiving attention in a number of fields such as healthcare, transport, and industry, among others. Several research efforts have been conducted to integrate IoT with smart environments. The integration of IoT with a smart environment extends the capabilities of smart objects by enabling the user to monitor the environment from remote sites. IoT can be integrated with different smart environments based on the application requirements. The work on IoT-based smart environments can generally be classified into the following areas: a) smart cities, b) smart homes, c) smart grid, d) smart buildings, e) smart transportation, f) smart health, and g) smart industry.illustrates the IoT-based smartenvironments.



The taxonomy of the IoT based smart environment. The devised taxonomy is based on the following parameters: communication enablers, network types, technologies, wireless standards, objectives, and characteristics

Communication Enablers

Communication enablers refer to wireless technologies used to communicate across the Internet. The key wireless Internet technologies are WiFi, 3G, 4G, and satellite. WiFi is mainly used in smart homes, smart cities, smart transportation, smart industries, and smart building environments; whereas, 3G and 4G are mainly used in smart cities and smart grid environments. Satellites are used in smart transportation, smart cities, and smart grid environments. Table presents the comparative summary of the communication technologies used in IoT based smart environments.

Network Types

IoT-based smart environments rely on different types of networks to perform the collaborative tasks for making the lives of inhabitants more comfortable. The main networks are wireless local area networks (WLANs), wireless personal area networks (WPANs), wide area networks (WANs), metropolitan area networks (MANs), and wireless regional area networks (WRANs). These networks have different characteristics in terms of size, data transfer, and supported reach ability.

Technologies

IoT-based smart environments leverage various technologies to form a comfortable and suitable ecosystem. These technologies are include sensing, communication, data fusion, emerging computing, and information security. Sensing technologies are commonly used to acquire data from various locations and transmit it using communication technologies to a central location. The emerging computing technologies, such as cloud computing and fog computing, deployed in the central location, leverage the data fusion technologies for integrating the data coming from heterogeneous resources. In addition, smart environments also use information security technologies to ensure data integrity and user privacy.

Local Area Wireless Standards

The commonly used local area wireless standards in IoT-based smart environments are IEEE 802.11, IEEE 802.15.1, and IEEE 802.15.4. These standard technologies are used inside the smart environment to transfer the collected data among different devices. IEEE 802.11 is used in smart homes, smart buildings, and smart cities. IEEE 802.15.1 and IEEE 802.15.4 have relatively shorter coverage than IEEE 802.11 and are used mainly in sensors and other objects deployed in the smartenvironments.