

UNIT 5

IoT Physical Servers and Cloud Offering

(3 Hours) — Internet of Things (CACS460)

⚠ Please Note

These notes were prepared quickly and **may contain errors or omissions**. Please cross-check important definitions, formulas, code and diagrams with your textbook, class notes and the official TU syllabus before relying on them in exams.

5.1 Introduction

IoT devices are resource-limited, so heavy storage, processing and application hosting are moved to **servers** and **cloud platforms**. A **physical server** is a real machine (in a data centre or on premises) that runs these services; **cloud computing** offers the same capability on demand over the Internet, with pay-per-use pricing and elastic scaling.

Cloud Computing Basics

- **Essential features:** on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service.
- **Service models:** IaaS (virtual machines, storage, network — e.g., Amazon EC2), PaaS (runtime platform — e.g., Google App Engine, Heroku), SaaS (complete application — e.g., Gmail).
- **Deployment models:** public, private, hybrid and community cloud.

Model	You manage	Provider manages	IoT example
IaaS	OS, middleware, apps, data	Servers, storage, network, virtualisation	Run your own MQTT broker on a cloud VM
PaaS	Application and data	Runtime, OS, infrastructure	Deploy IoT web app on a platform service
SaaS	Just use it	Everything	Ready-made IoT dashboard service

IoT Cloud Platforms

Platform	Notable IoT services
AWS IoT Core	Device registry, MQTT/HTTP gateway, rules engine, device shadow; with Lambda, S3, DynamoDB
Microsoft Azure IoT Hub	Bi-directional device messaging, device twins, routing; with Stream Analytics
Google Cloud	Pub/Sub, Dataflow, BigQuery for IoT pipelines (Google's dedicated IoT Core service has been retired, so it is built from these)
ThingSpeak	Simple cloud with channels, MATLAB analysis and charts; popular for student projects
Others	Blynk, Adafruit IO, IBM Watson IoT, Xively, ThingsBoard (open source)

Note: Cloud vendors rename or discontinue services often, so verify current names and availability on the provider's website.

5.2 Cloud Storage Models

IoT systems must store telemetry, device metadata, files and results. Different data needs different storage models.

Storage model	Description	IoT use
Object storage	Data stored as objects with metadata in flat buckets, accessed via HTTP APIs; very scalable and cheap (Amazon S3, Azure Blob)	Images, video, raw logs, backups, firmware files
File storage	Hierarchical files and folders shared over network (NFS/SMB)	Shared configuration or datasets
Block storage	Raw volumes attached to virtual machines; low latency	Databases and VM disks
Relational (SQL) database	Tables with fixed schema and ACID transactions (MySQL, PostgreSQL)	Device registry, users, billing
NoSQL database	Flexible schema; key-value, document, column, graph (MongoDB, Cassandra, DynamoDB)	High-volume sensor messages
Time-series database	Optimised for timestamped data (InfluxDB, TimescaleDB)	Telemetry, metrics, dashboards
Data lake / warehouse	Central repository of raw (lake) or processed (warehouse) data for analytics	Historical analytics and ML training

Hot, Warm and Cold Data

- **Hot:** recent data used constantly — kept on fast storage.

- **Warm:** occasionally used — cheaper storage.
- **Cold / archive:** rarely accessed (compliance, history) — cheapest storage such as archive tier.

Data lifecycle policies automatically move or delete data (e.g., keep raw readings 30 days, hourly averages 2 years). Also consider replication, backup, encryption at rest and regional/legal data-residency rules.

5.3 Communication APIs of IoT Systems

An **API (Application Programming Interface)** defines how software components talk to each other. IoT systems expose and consume APIs so that devices, cloud services and applications can exchange data.

REST API

- Resources identified by URIs; operations use HTTP methods: **GET** (read), **POST** (create), **PUT/PATCH** (update), **DELETE** (remove).
- Stateless request–response; data typically in **JSON**.
- Easy and widely supported, but polling is needed for updates and headers add overhead.

```
POST /api/v1/devices/station1/readings HTTP/1.1
Host: iot.example.com
Authorization: Bearer <token>
Content-Type: application/json

{"temp": 24.5, "hum": 61, "ts": 1760000000}
```

WebSocket API

- Starts as HTTP then upgrades to a persistent **full-duplex** TCP connection.
- Either side can send at any time with very low overhead — ideal for live dashboards and real-time control.

MQTT API (Publish–Subscribe)

- Lightweight protocol on TCP (port 1883, 8883 with TLS) designed for constrained devices and unreliable networks.
- Uses a **broker**, **topics** (e.g., home/livingroom/temp) and QoS levels 0 (at most once), 1 (at least once), 2 (exactly once).
- Also provides retained messages and Last Will and Testament for device-offline detection.

CoAP API

- Constrained Application Protocol: a RESTful protocol over **UDP** for very small devices (port 5683).
- Supports GET/POST/PUT/DELETE, observe (subscribe to a resource) and multicast; secured with DTLS.

Basis	REST/HTTP	MQTT	CoAP	WebSocket
Model	Request–response	Publish–subscribe	Request–response (+observe)	Full-duplex
Transport	TCP	TCP	UDP	TCP
Overhead	High	Very low	Very low	Low
Best for	Web integration, simple calls	Telemetry from many devices	Constrained sensors/6LoWPAN	Real-time web UI

Example 5.1 — Choosing a protocol

A battery-powered soil sensor sends a 20-byte reading every 10 minutes over a lossy link. A web page needs live updates. Which protocols suit each part?

Solution

Sensor to cloud: **MQTT** (or CoAP) because of low overhead and tolerance of unreliable links.

Cloud to browser: **WebSocket** to push live updates without polling; REST for occasional queries of history.

Cloud API Security

- Authenticate devices and users (API keys, OAuth 2.0 tokens, X.509 certificates).
- Use TLS for all traffic; rate-limit requests; validate inputs; keep secrets out of firmware source code.

Key Takeaways

- ✓ Cloud gives elastic storage and compute for IoT: IaaS, PaaS, SaaS; examples AWS IoT, Azure IoT Hub, ThingSpeak.
- ✓ Pick storage by data type: object (files/images), SQL (structured), NoSQL and time-series (telemetry), data lake (analytics).
- ✓ Use lifecycle policies: hot → warm → cold.
- ✓ APIs: REST (HTTP, request–response), WebSocket (full-duplex), MQTT (pub–sub), CoAP (UDP, constrained).
- ✓ Secure all APIs with authentication, authorization and TLS.

— *End of Unit 5* —

bcalink.com