IoT Concepts

Verified 2026-08-08

The concepts behind any IoT system — constrained devices, link layers, protocols, device state, edge, identity, security, and OTA updates.

Contents

Mental model

IoT is distributed systems where the binding constraints are physical rather than economic. A node may hold tens of kilobytes of RAM, run a decade on one battery, and reach the network over a link that is slow, lossy, and asleep most of the time. Those three limits — memory, energy, connectivity — decide the protocol, the data model, and the security design, in that order. Assume the device is offline, the message was lost, and the firmware in the field is old; anything better is a bonus.

Constrained devices

Class	RAM	Flash
C0	`<< 10 KiB`	`<< 100 KiB`
C1	`~ 10 KiB`	`~ 100 KiB`
C2	`~ 50 KiB`	`~ 250 KiB`

RFC 7228 grades constrained nodes by memory, because memory decides whether an IP stack, a TLS session, or a JSON parser fits at all. Power is graded separately: E-

classes describe the energy budget — from harvested to mains — and P-classes the duty cycle, running from P0 normally-off through P1 low-power to P9 always-on.

```
C0  gateway-dependent, no usable IP stack
C1  CoAP + DTLS fits, HTTP + TLS does not
C2  IP stack fits, still budget every byte
```

> **Gotcha:** the class is set by memory, but the behavior you must design around is set by the P-class. A normally-off sensor is unreachable for most of its life, so anything assuming you can call the device is wrong however much RAM it has.

Connectivity and link layers

Link	Trade
BLE, Zigbee, Thread	Metres, mesh, needs a gateway
Wi-Fi, Ethernet	LAN speed, mains power assumed

Link	Trade
LoRaWAN, Sigfox	Kilometres, bytes per message
NB-IoT, LTE-M	Cellular, licensed spectrum

Choose the link from the power and range budget, not the data rate. Short-range radios form star or mesh networks needing a gateway to reach IP. Low-power wide-area networks trade throughput for kilometres and a decade of battery — LoRaWAN's smallest frame leaves 11 octets of payload, a Sigfox uplink at most 12 bytes. NB-IoT alone has no duty-cycle cap, because it runs in licensed spectrum.

```
LoRaWAN   250 bps - 50 kbps, 1% duty (EU868)
Sigfox    up to 140 uplinks per device per day
NB-IoT    ~60 kbps up, 100% duty, ~10 yr target
```

> **Gotcha:** unlicensed sub-GHz duty-cycle caps are regulatory, not technical. At 1% airtime a device cannot be commanded on demand and cannot receive a firmware image in any reasonable time — no amount of engineering buys past that ceiling.

Application protocols

Protocol	Shape
MQTT (1883 / 8883)	Broker pub/sub over TCP
CoAP (5683 / 5684)	REST over UDP, DTLS-secured
HTTP	Request/response, costly per message
LwM2M	Device management objects on CoAP

MQTT is broker-mediated publish/subscribe and the default for telemetry into a backend. CoAP is REST over UDP, small enough for a C1 node, and cross-proxies to HTTP so one resource model reaches the web. HTTP

works but pays for headers and a handshake every exchange. LwM2M adds a management object model on top of CoAP.

```
MQTT: device --publish--> broker --> consumers
CoAP: device <--GET/PUT--> server, no broker
```

> **Note:** CoAP's confirmable messages retransmit with exponential backoff, giving reliability without TCP's connection state. That is the point — a datagram costs a constrained node far less RAM than a socket plus a TLS session.

Telemetry, commands, and device state

Flow	Direction and shape
Telemetry	Device to cloud, high volume
Command	Cloud to device, needs an ack
Shadow / twin	Cloud-held desired and reported

These are three different problems. Telemetry is append-only and tolerates loss. A command needs an acknowledgement path, because sent is not applied. Long-lived state belongs in a *shadow* (or twin): the cloud holds a `desired` document written by apps and a `reported` one written by the device, publishes the difference as a delta, and the device reconciles when it next wakes.

```
{
  "state": {
    "desired":  { "interval": 60 },
    "reported": { "interval": 300 }
  },
  "version": 12
}
```

> **Gotcha:** deltas can arrive out of order, so a device must compare the document `version` and discard anything older than what it has already applied. Trusting the newest message received rather than the highest version silently reverts settings.

Edge and gateways

Role	Why it exists
Protocol translation	Non-IP radios reach IP
Store and forward	Survive a backhaul outage
Filter and aggregate	Cut bandwidth and egress cost
Local control loop	Act without cloud round-trip

A gateway terminates the constrained network, speaks IP upstream, and does the work the node cannot: buffering through an outage, collapsing a thousand readings into one summary, closing latency-critical loops locally. Edge computing generalises that — process where the data is produced whenever bandwidth,

latency, privacy, or cost make a cloud round-trip the wrong trade.

```
sensors -> gateway -> backhaul -> cloud
             |
             +-- buffer, filter, local rules
```

> **Gotcha:** a gateway that buffers through an outage will flood the backend the moment it reconnects. Rate-limit the drain and make ingest idempotent, or the recovery becomes a second, larger outage.

Identity and provisioning

Mechanism	Strength
Shared symmetric key	Weak; extractable from flash
Per-device X.509	Strong; needs a PKI to run
Secure element or TPM	Private key never leaves the chip

Every device needs an identity that is unique, immutable, and not extractable — NIST's baseline lists device identification first for that reason. Provisioning is how it reaches the device and the registry: injected on the production line, or claimed at first boot by trading a bootstrap credential for an operational one. Keep the private key in a secure element, so a device pulled off a wall yields no usable fleet credential.

```
factory  key + cert burned in, registered
bootstrap  claim once, swap for operational
```

> **Warning:** one shared key across a product line means a single extracted key compromises every unit ever shipped, with no revocation short of a recall. Per-device credentials are the difference between an incident and a recall.

Securing the fleet

Capability	What it requires
Device identification	Unique logical and physical ID
Device configuration	Authorized changes only
Data protection	Crypto at rest and in flight
Logical access	Restrict interfaces and services
Software update	Verified, authorized, revertible
State awareness	Report its own security state

NIST IR 8259A defines those six capabilities as the baseline that makes a device *securable*. Treat them as hardware acceptance criteria — none retrofits after the silicon ships. On the network side, MUD lets a manufacturer publish the traffic a device is supposed to send as an ACL the network enforces, so a compromised thermostat that only ever needed one host is denied everything else.

```
device --DHCP/802.1AR--> MUD URL
manager --fetch--> MUD file --> switch ACL
```

> **Gotcha:** transport encryption is not device authentication. A DTLS or TLS session that validates only the server proves the device reached the right backend, not that the backend reached the right device. Mutual authentication is what makes a cloned credential fail.

Updates and lifecycle

Stage	Concern
Roll out	Staged, canaried, resumable
Verify	Signature checked before execution
Fall back	A/B slots, automatic revert
Decommission	Revoke credentials, wipe data

A device shipped today may run for a decade, so an unattended, secure update path is the most important

thing you build. RFC 9019 frames it around a signed *manifest*: the signature, the vendor and class IDs deciding whether an image applies, a sequence number, and where to fetch the payload. A bootloader verifies against a stored trust anchor before executing anything new.

```
manifest: signature, vendor + class id,
          sequence number, image URI
boot: verify -> install -> revert on failure
```

> **Warning:** without a monotonic sequence number an attacker can replay a correctly signed *older* image and reopen a patched vulnerability. A valid signature establishes authenticity, never freshness.

Interoperability and data models

Standard	Domain
Matter	Consumer smart home devices

Standard	Domain
LwM2M	Device management objects
OPC UA	Industrial automation
Sparkplug B	MQTT payload and state contract

Moving bytes is solved; agreeing what they mean is not. A topic carrying a bare float is a private contract no other system can read. Object models supply the semantics — Matter runs over Thread, Wi-Fi and Ethernet, commissioned over BLE; OPC UA covers plant equipment; Sparkplug B pins MQTT payloads plus birth and death state. Choose early: retrofitting a schema across deployed firmware is an OTA campaign, not a refactor.

```
{
  "schema": 2,
  "ts": 1786200000,
  "temp_c": 21.4
}
```

> **Tip:** if no standard model fits, at least version your own and carry that version in every message. Devices you shipped years ago will keep emitting the old shape long after the backend moved on.

Further reading

- RFC 7228 — Terminology for Constrained-Node Networks
- RFC 7252 — The Constrained Application Protocol (CoAP)
- RFC 8376 — Low-Power Wide Area Network (LPWAN) Overview
- RFC 8520 — Manufacturer Usage Description Specification
- RFC 9019 — A Firmware Update Architecture for IoT Devices
- NIST IR 8259A — IoT Device Cybersecurity Capability Core Baseline