IoT Processing Topologies and Types: Data Format, Importance of Processing in IoT, Processing

Topologies, IoT Device Design and Selection Considerations, Processing Offloading.

6.1 Data Format

The Internet hosts an enormous and continuously growing volume of data generated by billions of users and an increasing number of devices. This includes both human-generated data (like emails, social media posts, and multimedia files) and non-human data (like sensor data from automated systems). IoT contributes significantly to this data load, producing a wide range of information types. Data in IoT can generally be categorized into **two main types**: **Structured Data** and **Unstructured Data**.

6.1.1 Structured Data

- **Definition**: Structured data is highly organized and follows a specific format or schema, making it easily searchable and manageable.
- **Characteristics**:
  - Typically associated with relational database management systems (RDBMS).
  - Uses structured fields with length limits, such as phone numbers, social security numbers, and other standardized entries.
- **Examples**:
  - Commonly found in reservation systems, banking systems, inventory controls, and similar applications where data organization and retrieval are essential.
  - Accessed and managed through SQL (Structured Query Language).
- **In IoT**: Structured data makes up a smaller portion of IoT data as most IoT information lacks rigid structures.

6.1.2 Unstructured Data

- **Definition**: Unstructured data lacks a pre-defined format, making it more flexible but harder to organize and query.

- **Characteristics**:
  - Highly variable and application-dependent, with no fixed structure.
  - Commonly managed using NoSQL databases to handle its flexible formats.
- **Examples**:
  - **Human-Generated**: Text files, emails, videos, images, phone call recordings, chat logs, etc.
  - **Machine-Generated**: Sensor readings from traffic systems, building monitors, industrial equipment, satellite imagery, and surveillance videos.
- **In IoT**: Unstructured data dominates IoT as the diverse data sources and formats require flexible handling.

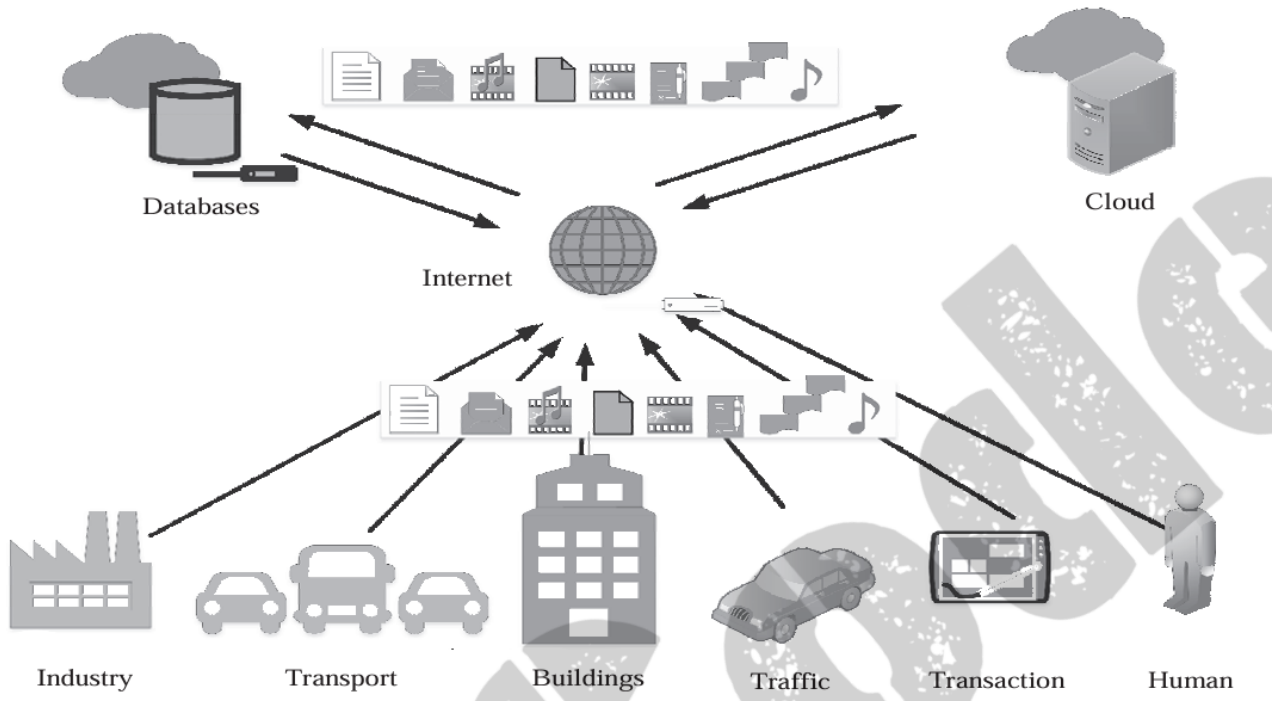


Figure 6.1 The various data generating and storage sources connected to the Internet and the plethora of data types contained within it

6.2 Importance of Processing in IoT

The massive data generated by IoT devices globally places immense pressure on network infrastructure, creating a demand for intelligent and efficient processing. Effective processing ensures that data is managed according to urgency, optimizing response times and resource use. In IoT, data is categorized based on processing urgency:

Data Categories by Urgency

1. **Very Time-Critical Data**:
    - **Definition**: Requires immediate processing due to the high risk or impact of delays.
    - **Examples**: Data from systems like flight control, healthcare, and emergency response, where decisions need to be made within milliseconds.
    - **Processing Needs**: High; data must be processed as close to the source as possible to minimize latency.
2. **Time-Critical Data**:
    - **Definition**: Important data that can tolerate minor delays, generally a few seconds.
    - **Examples**: Traffic monitoring, vehicle systems, surveillance, and smart home systems.
    - **Processing Needs**: Moderate; data can be transmitted for remote processing, like cloud or collaborative systems, if on-site processing is not feasible.
3. **Normal Data**:
    - **Definition**: Data that can tolerate longer delays, from minutes to hours, as immediate processing is not essential.
    - **Examples**: Agriculture, environmental monitoring, and other fields where data sensitivity is low.
    - **Processing Needs**: Low; can be processed remotely or at a leisurely pace without significant impact.

Considerations for Processing in IoT

- **Very Time-Critical Data**: Immediate, on-site processing is necessary to support rapid decision-making.
- **Time-Critical Data**: Processing may occur on-site or remotely, allowing flexibility in the deployment of processing resources.
- **Normal Data**: Minimal urgency for processing, allowing for cost-effective and delayed processing solutions.

6.3 Processing Topologies

Choosing an effective processing topology is essential for efficient IoT architecture, helping to reduce network bandwidth usage and conserve energy while meeting the required processing latencies. IoT processing topologies are broadly divided into two types: **On-site Processing** and **Off-site Processing**. Off-site processing can further be classified into **Remote Processing** and **Collaborative Processing**.

6.3.1 On-Site Processing

- **Definition**: In on-site processing, data is processed directly at the source, typically within the sensor node or device itself.
- **Use Cases**: Best suited for applications requiring extremely low latency, such as healthcare systems or flight control systems, where data changes rapidly and delays could lead to serious consequences.
- **Example**: A fire detection system using a temperature sensor processes data locally to quickly detect and alert users to potential fires. The sensor node may also send data to a remote system for further analysis or storage.
- **Advantages**: Minimizes delay and reduces reliance on network connectivity, which is crucial for real-time systems.

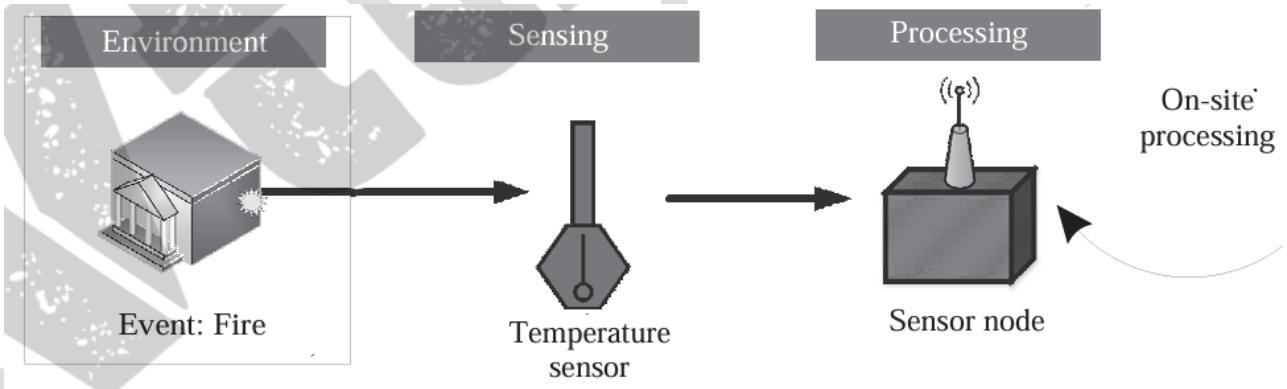


Figure 6.2 Event detection using an on-site processing topology

---

6.3.2 Off-Site Processing

- **Definition**: In off-site processing, data is collected by sensor nodes and transmitted to another location for processing, allowing for cost-effective setups with fewer processing demands on the source device.

1. **Remote Processing**:
    - **Definition**: Data is sent to a remote server or cloud-based system, where a powerful central processor handles analytics.
    - **Use Cases**: Common in large IoT deployments where scalability and resource-sharing are important.
    - **Example**: Data from multiple sensor nodes is transmitted to a cloud server for processing, allowing for simpler and smaller devices at the deployment site.
    - **Advantages**: Reduces on-site processing costs, provides massive scalability, and enables centralized resource management. However, it relies heavily on network connectivity and consumes significant bandwidth.

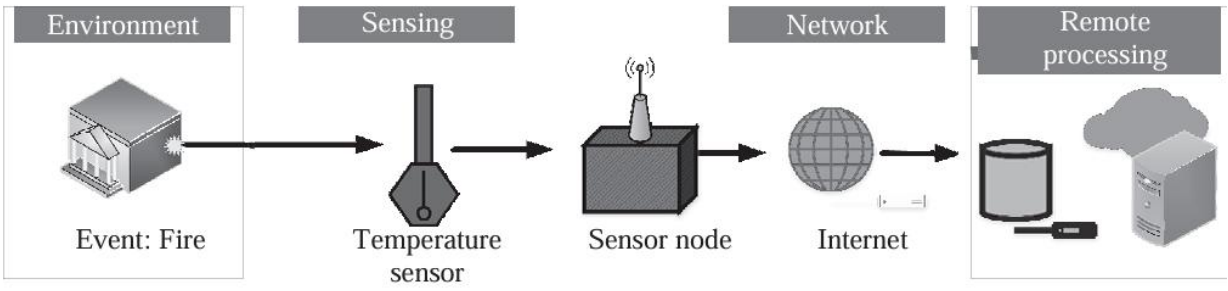


Figure 6.3 Event detection using an off-site remote processing topology

2. **Collaborative Processing**:
    - **Definition**: Nearby nodes combine processing power to handle data locally, without needing continuous access to a remote server.
    - **Use Cases**: Suitable for areas with limited or no network connectivity, especially in large-scale, remote deployments.
    - **Example**: In agricultural monitoring, sensors in close proximity work together to process data locally, reducing the need for constant remote access and saving network bandwidth.
    - **Advantages**: Reduces latency, conserves bandwidth, and is cost-effective for applications with infrequent data processing needs. Often implemented using mesh networks for easy connectivity among nodes.

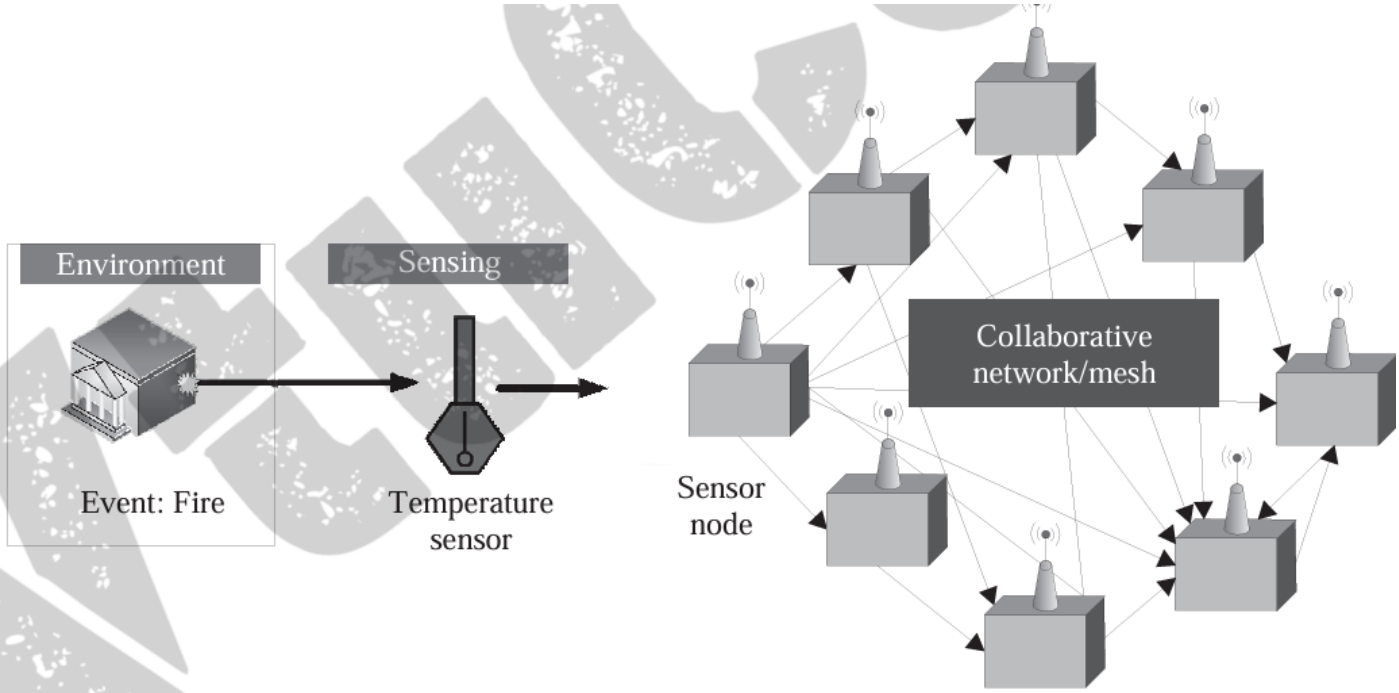


Figure 6.4 Event detection using a collaborative processing topology

6.4 IoT Device Design and Selection Considerations

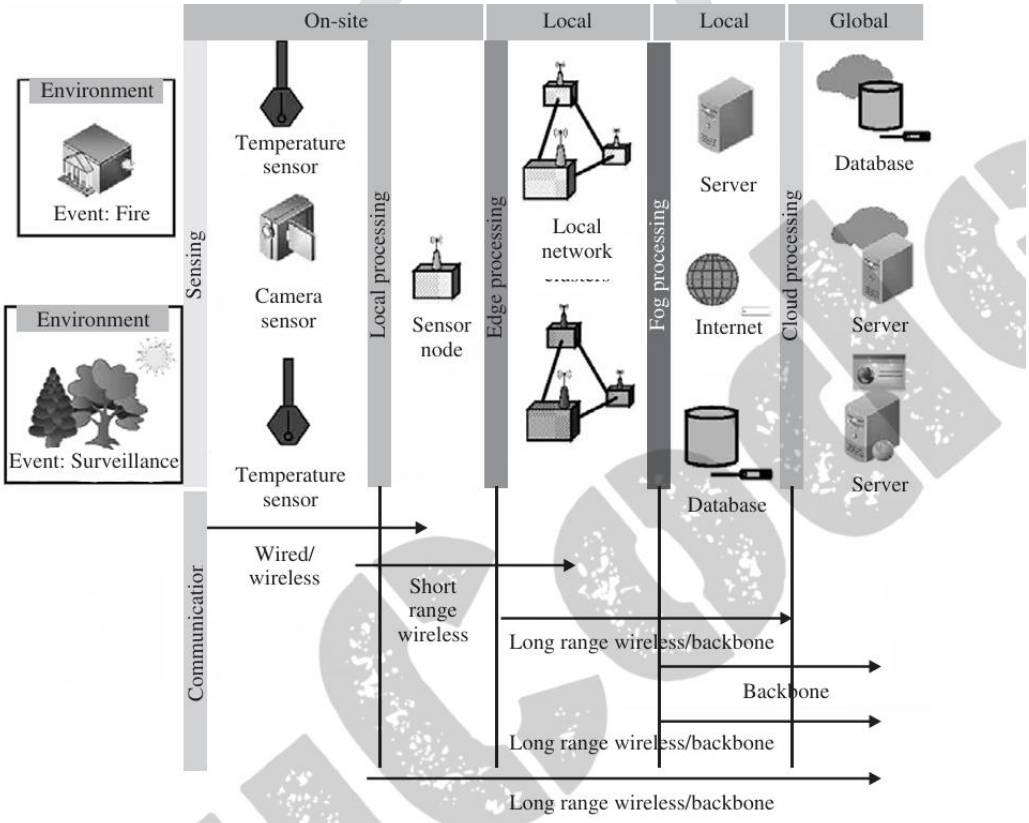
Designing an IoT device requires careful selection of a processor, as various factors influence the usability, design, and cost-effectiveness of the sensor node.

- **Size**: This is a crucial factor in determining the form factor and energy consumption of the sensor node. Larger form factors generally require more energy and may not be suitable for applications like wearables, where a minimal footprint and low power consumption are essential.
- **Energy**: The energy efficiency of a processor is particularly important for IoT devices, especially those deployed in remote or hard-to-reach locations. Devices with high energy demands need frequent battery replacements, which can decrease the long-term sustainability of the solution.
- **Cost**: The cost of the processor and sensors significantly impacts the feasibility of high-density sensor deployments in IoT solutions. Low-cost devices allow for greater sensor coverage, making large-scale deployments, such as environmental monitoring or safety systems, more affordable.
- **Memory**: The memory capacity, including both volatile and non-volatile memory, enables local processing and storage, data filtering, and formatting. Devices with greater memory can handle more complex tasks, although they may be more expensive.

- **Processing Power**: This defines the range of tasks the device can support. Applications that require video or image processing need higher processing power, while simpler tasks, like temperature monitoring, require less.
- **I/O Rating**: The input-output rating of the processor, including voltage requirements, influences the device's compatibility with various sensors and its overall circuit complexity. Newer processors often have a 3.3V I/O rating, compared to 5V for older processors, which means additional circuitry may be needed for compatibility with legacy sensors, impacting cost and complexity.
- **Add-ons**: Additional features such as ADC units, clock circuits, USB and ethernet connections, and wireless capabilities add versatility to the IoT device. Having these add-ons pre-integrated simplifies hardware development and makes the processor more adaptable for a range of applications.

6.5 Processing Offloading

Processing offloading is crucial in IoT for creating scalable, energy-efficient, and cost-effective solutions. By offloading data processing from the device to external locations, IoT systems maintain minimal complexity on-site while supporting diverse and dense deployments. Offloading is classified based on location, decision-making, and specific considerations.



re 6.5 The various data generating and storage sources connected to the Internet and the plethora of data types contained within it

6.5.1 Offload Location

The choice of offload location affects the feasibility, cost, and sustainability of IoT applications. Offloading can occur at four main locations:

- **Edge**: Processing occurs near or at the data source itself, helping with data aggregation, manipulation, and bandwidth reduction. This approach minimizes delays and processes data directly on the IoT device or within its immediate network.
- **Fog**: Fog computing provides a decentralized computing infrastructure between the data source and the cloud, reducing bandwidth usage, latency, and data flow across the Internet. Localized to a geographical area, fog nodes (often gateways) serve IoT nodes nearby without necessarily requiring Internet access.
- **Remote Server**: A remote server with substantial processing power can handle data from resource-constrained IoT devices. While generally cost-effective, this option may face scalability issues and is more difficult to maintain compared to cloud solutions.
- **Cloud**: Cloud computing provides scalable resources, storage, and platforms remotely over the Internet. The cloud supports extensive scalability, with resources dynamically allocated as needed without the user needing to manage hardware. This solution, though widely accessible, can incur high network bandwidth costs and introduce latency issues.

6.5.2 Offload Decision-Making

Deciding where and how much to offload is key to effective processing in IoT architectures. Decision-making methods vary based on data rate, network bandwidth, criticality, and available processing resources.

- **Naive Approach**: A simple, rule-based method that offloads data to the nearest processing location based on set criteria. While easy to implement, this approach lacks scalability and isn't ideal for high-density IoT deployments with complex data.
- **Bargaining-Based Approach**: This approach optimizes network congestion and quality of service (QoS) by balancing multiple parameters across the IoT deployment. Using techniques like game theory, the bargaining approach finds a trade-off that improves overall QoS, though individual parameters may be compromised to benefit the entire system.
- **Learning-Based Approach**: Utilizing past trends, learning-based methods use machine learning to optimize offloading based on historical data, enhancing performance over time. This approach, while

powerful, requires more memory and processing resources to make real-time decisions based on learned data.

6.5.3 Offloading Considerations

Several factors influence the choice of offloading type, depending on the IoT application and hardware. These include:

- **Bandwidth**: The data-carrying capacity of a network affects how much data can be transmitted between points. Higher bandwidth allows faster data transfer, which is essential for applications that generate large volumes of data.
- **Latency**: This is the delay between the start and completion of a task. Network latency and processing latency can hinder performance; thus, applications with critical time constraints require low-latency solutions.
- **Criticality**: This factor assesses the urgency of tasks. For instance, a fire detection system demands a faster response time than agricultural monitoring, as delay in critical applications could lead to severe consequences.
- **Resources**: The offload location's available resources, such as processing power and analytical capabilities, determine its suitability. Allocating high-energy, high-capacity resources to handle low-volume data is wasteful, so offloading should be matched to resource capabilities.
- **Data Volume**: The amount of data generated impacts the processing capacity required at the offload location. For dense deployments, the offloading site should be robust enough to manage high data volumes without bottlenecks.