Edge Computing

Edge Computing

- In a conventional cloud environment, computing, storage, and most communication occurs within each data center.
- Instead of concentrating cloud facilities in a single geographic location, an edge architecture places some computational facilities near the source of data.
- The chapter gives the motivation for edge computing, and explains why the Industrial Internet of Things (IIoT) works well with edge processing.

The Latency Disadvantage of Cloud

- Public data centers represent a larger move toward centralization because a given cloud data center houses computing facilities used by many organizations.
- Although the cloud approach has many advantages, it does have the disadvantage of introducing higher network latency because a data center is remote from the customers it serves.
- As we have seen, cloud providers attempt to minimize network latency in two ways:
  The use of multiple, geographically diverse sites
  Low-latency network connections

The use of multiple, geographically diverse sites.

- To reduce latency, a public cloud provider does not collect all its facilities into a single data center.
- Instead, a provider creates multiple data centers, and places them at geographic locations (sometimes called *zones*) near sets of customers.
- For example, a provider might spread multiple data centers across North America, Europe, and so on.

Low-latency network connections.

The second technique providers use to minimize latency involves low-latency network connections.

A large enterprise customer may choose to lease a direct connection from the customer site to a cloud data center.
To minimize latency for smaller customers, a major cloud provider connects directly to Tier-1 Internet backbone networks.
Despite the above optimizations, the latency between a user and a cloud data center can exceed 100 milliseconds, and may depend on network traffic.

Situations Where Latency Matters

When a corporation performs routine business (e.g., recording sales transactions or submitting monthly payroll information), a slight delay is unnoticeable.

In some situations, however, low delay can be crucial, either financially or otherwise.

In the financial industry, for example, a small delay in making stock trades can result in a huge loss.

In the health care industry, a small delay in receiving data from a patient monitor can delay activation of an implanted medical treatment device.

Industries That Need Low Latency

Consumers seldom worry about low latency. For example, when a user interacts with the controls in their smart home (e.g., to change the temperature), small delays go unnoticed.

However, industries that employ real-time control systems — sensors and actuators that monitor and control processing — rely on low delay.

Systems that provide fast responses to human users (e.g., responses to keystrokes a user enters) also need low latency.

Agriculture	Mass transit
Automotive	Oil and gas
Environmental monitoring	Retail
Health care	Transportation
Manufacturing	Utilities

Figure 16.1 Example industries in which applications require low latency.

- Of course, not all sensors require low latency.
- Even some medical devices do not require instant responses.
- For example, wearable medical devices exist that collect biometric data for long-term trend analysis, either by an AI program or a human medical professional.
- Such devices typically accumulate measurements over multiple hours or multiple days before uploading values to the cloud.

- **Moving Computing To The Edge**
  How can cloud computing be adapted to meet the requirements for low latency? The answer lies in an architecture known as *edge computing*.
- The idea is straightforward: place some of the computing facilities near each source of information, and perform initial processing locally.
- Simultaneously run applications in a cloud data center, and use the cloud applications to handle computational-intensive tasks. The term *edge* arises because cloud data centers typically connect to a centralized point of the Internet, whereas networking professionals say that users' devices connect to the "edge" of the Internet.
- Hence, computation performed near such devices occurs near the edge.
  To understand how local computing can help, consider a simplistic case: a sensing device. Suppose an app running on a user's cell phone uses Bluetooth technology to collect data from a sensor and then sends the data to an application running in the cloud for analysis.

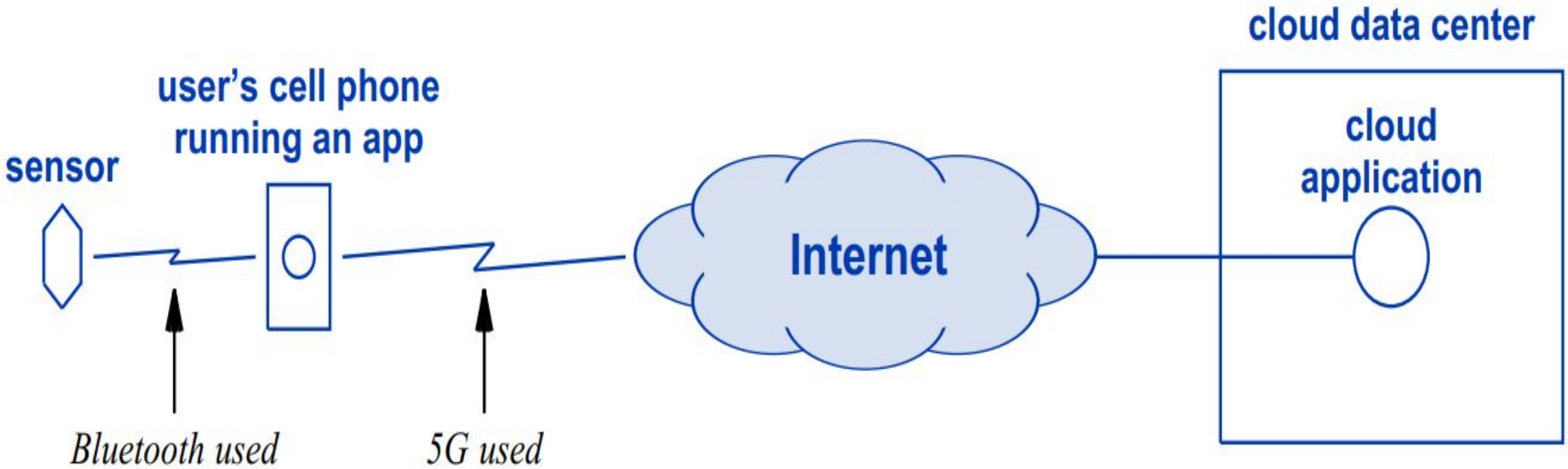

Figure 16.2 An app on a user's cell phone that gathers information from a sensor and sends the information to an application running in a cloud data center for analysis.

- The application running in the cloud has access to powerful computing facilities, and can perform computationally intensive processing, such as running AI software to analyze the sensor data.
- Meanwhile, the app on the user's cell phone handles local tasks.
- The app uses Bluetooth to communicate with the sensor and gather data.
- It can also perform basic data processing before sending the data on to the cloud.
- In such cases, the data being transferred from the sensor may be lost or corrupted.
- The app running in the cell phone can detect such problems, discard corrupted data values, and restart the transfer once the interference passes.
- In addition, the app can inform the user if the sensor becomes disconnected or the battery in the sensor needs to be recharged.
- To expand local computing capabilities, the edge computing approach places a miniature data center near locations that require low latency responses.
- Software running in the edge data center performs computation that requires low latency, and software running in a cloud data center performs computations that do not require rapid responses.

- **Extending Edge Computing to a Fog Hierarchy**
  Where should edge data centers be placed?
- The locations and sizes depend on the applications being supported and the latency requirements.
- To achieve the lowest possible latency, an edge facility must be as close to each user as possible (e.g., in each cell tower).
- For applications with less stringent requirements, an edge computing facility might serve a neighborhood of multiple cell towers or a geographic region with many neighborhoods.

Level	Computing Equipment	Connects To Multiple
1	Public cloud data center	Regional data centers
2	Regional data center	Town data centers
3	Town or neighborhood data center	Cell towers
4	Computers in a cell tower	Users' phones
5	User's phone	Sensoring devices

Figure 16.3 A possible edge computing hierarchy with computing facilities arranged into five levels.

To distinguish between edge facilities located adjacent to end users and edge facilities that serve larger geographic regions, industry sometimes uses the term *fog data center* to refer to an intermediate data center that serves a larger geographic area.

The name is intended to be humorous by referring to the idea that fog arises when a cloud comes down to earth. By analogy, a fog data center uses cloud technology, but places a smaller data center closer to endpoints.

Caching at Multiple Levels of a Hierarchy

Although the description above focuses on computation, a multi-level hierarchy of edge and fog data centers permits another optimization: data caching.

When an endpoint requests information, an application running in the nearest edge data center can obtain the requested information, store a copy locally, and return a copy to the requester.

If another endpoint subsequently requests the same information, a copy will be returned from the edge data center without any need to contact the original source.

Caching can be used at all levels of the hierarchy, and works for data flowing in either direction.

For example, suppose a hierarchy contains a cloud data center at the top, two fog data centers that each serve a region, and edge data centers in each region.

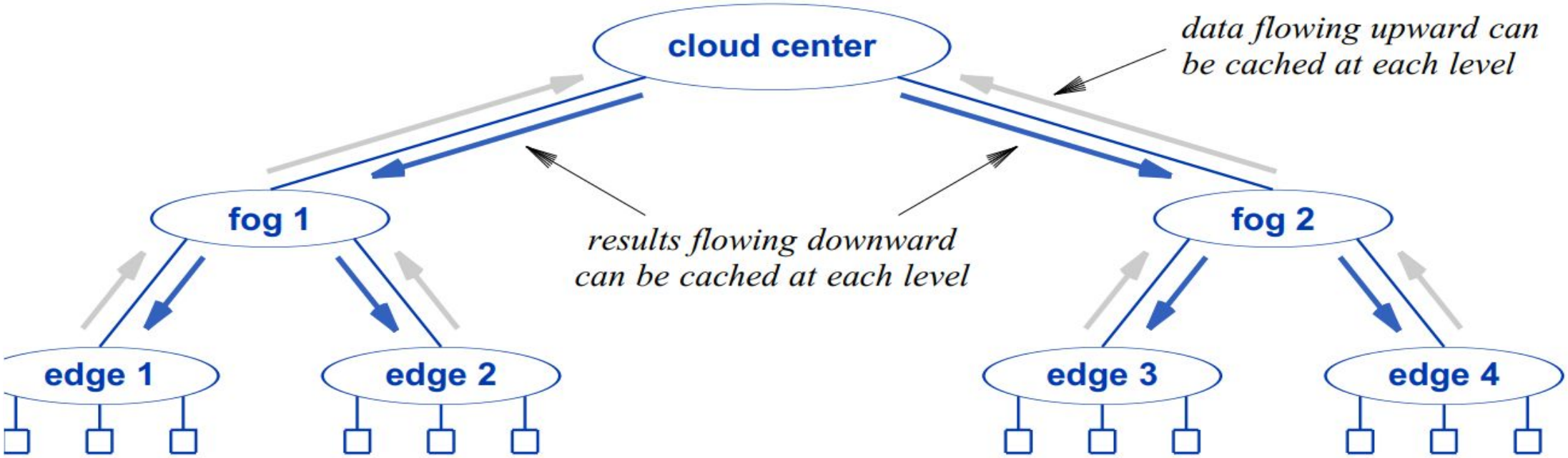

Figure 16.4 An example hierarchy of edge and fog data centers that each cache copies of data.

- As endpoints generate data, the data flows upward, and each data center keeps a cached copy.
- If an endpoint connected to edge data center *edge 1* generates data, *edge 1* caches a copy and forwards the data to *fog 1*, which keeps a cached copy and forwards a copy to the cloud, which also keeps a copy in its cache.
- If another endpoint connected to edge 1 requests the data, *edge 1* will return the values from its cache.
- Similarly, if an endpoint attached to *edge 2* requests a copy, *edge 2* will obtain the data from *fog 1*, cache a copy and return a copy to the endpoint that made the request.
- Caching also applies to data items flowing down the hierarchy, and can include items that span multiple applications.
- As an example, consider a supply chain management system for a retailer with multiple stores.
- As items are sold, information is propagated upward through the hierarchy, allowing regional managers to order replacement items from a supplier.
- The system also maintains information used to ship items between stores to obtain items from a nearby store when the supply temporarily runs low.
- Such information originates from a central company database and flows downward through the hierarchy, as needed.
- The caching system works equally well with both types of information.

An Automotive Example

A single application may benefit by using multiple levels of the hierarchy.

To see how, consider the automotive industry, which is working to create a system to support *connected vehicles*.

Once the system becomes operational, each vehicle, whether self driven or driven by a human, will communicate with nearby vehicles as well as with communication facilities permanently placed near roadways.

The system will handle safety as well as navigation.

For safety, mechanisms will ensure that a vehicle will not collide with another vehicle nor cause problems for others by maneuvering dangerously or slowing needlessly.

Three aspects of the connected vehicle system lend themselves to the edge computing approach.

Low latency / real-time requirements
Geographic locality and awareness
The wide scope needed for route planning and navigation

Low latency / real-time requirements.

Because vehicles travel at high speed, collision avoidance systems operate locally.

Such systems maintain information about the position and direction of surrounding vehicles, allowing them to calculate actions quickly when a problem occurs.

Geographic locality and awareness.

To calculate a safe speed and choose an action that will avoid a collision, a collision avoidance system must also have information about local road conditions.

For example, is the roadway rough or has it become covered with ice?

Similarly, a vehicle must be informed when an accident closes the road or when traffic ahead stops unexpectedly.

Because they must act quickly, vehicles closest to the problem must learn of the problem first.

That is, notification mechanisms must be aware of geographic positions.

The wide scope needed for route planning and navigation.

- Route planning systems use map data to perform a global optimization: the system chooses a path that minimizes the travel time over the entire trip.
- However, navigation systems must also adapt to local conditions, routing around temporary problems.
- Thus, route planning and navigation require information about local conditions.
- The items described above suggest how automotive systems might use a hierarchy of edge computing facilities.
- A vehicle can track adjacent vehicles directly.
- To report current conditions and to learn about traffic and the roadway ahead, a vehicle might communicate with edge facilities along the road (e.g., in each cell tower).
- The edge facilities can learn about changes and report them rapidly (in seconds).

- At the next level of the hierarchy, we can imagine a set of data centers that each serve a small area.
- The data center runs software that collects and correlates information from multiple edge facilities.
- For example, by receiving reports of rain on various roads, software can calculate the path of a rain storm as it moves through the area and warn vehicles before they encounter the storm.
- Information in such data centers tends to change slowly (e.g., in minutes instead of seconds).
- At the third level of the hierarchy, we can imagine a set of regional data centers
  that each collect and process information from areas within their region.
- For example, systems in a regional data center might use AI to analyze traffic patterns and discover that a particular route becomes congested between 4:00 PM and 7:00 PM on workdays.

Edge Computing and IIoT

The term *Industrial Internet of Things* (*IIoT*) refers to an enhanced, larger-scale version of the Internet of Things.

The primary difference between consumer IoT systems and Industrial IoT systems lies in the importance: a company depends on an IIoT system as part of a critical business function.

As an example of IIoT, consider automated manufacturing.

The assembly line in an automated factory consists of robots at each station plus conveyors that move items down the line.

Raw materials enter at one end, and finished products emerge at the other end.

Because such assembly lines form a critical aspect of the company's business, the company loses money if the line stops. Consequently, sensors at each stage of the assembly line monitor items entering the assembly line, robots along the line, the progress of the entire line, and the quality of the items being manufactured.

A control system gathers data from the sensors, analyzes the data to detect and predict problems, and responds either by resetting equipment or notifying a human when it cannot handle a problem.

- An automated assembly line illustrates the characteristics often found in IIoT applications that distinguish them from most consumer IoT applications:
  Specific latency requirements
  Geospatial knowledge
  Large volumes of data with various QoS requirements
  A need for data filtering
  High availability requirements
  Security requirements

- *Specific latency requirements*.

Instead of a general desire for high performance, IIoT applications have specific requirements (e.g., if a specific assembly line robot malfunctions, it must be shut down within 150 milliseconds after a problem is detected).

Geospatial knowledge.

An IIoT system must be aware of locations and spatial relationships (e.g., the location of a failure and the set of surrounding systems that will be affected).

Large volumes of data with various QoS requirements.

IIoT applications often employ many sensors and video cameras that each generate data continuously, resulting in large volumes of data; technologies such as 5G wireless enable especially high traffic volumes. Each type of data may have specific *Quality of Service* (*QoS*) requirements, such as data rates and bounds on latency.

- *A need for data filtering*.

It does not make sense to send all the raw data gathered from sensors to the cloud for processing because data transport and computational cycles incur expense.

More important, because a local edge data center can handle items that require immediate action, applications running in the cloud only need data that allows them to analyze long-term trends (e.g., whether a given factory has more failures than other factories).

Even local processing benefits from filtering (e.g., because an adaptive cruise control application only handles sensor readings for nearby objects, filtering avoids having the application waste cycles on other data).

High availability requirements

Because a company depends on IIoT systems to sustain their business, the systems must be reliable. Thus, IIoT may need to employ redundancy (e.g., have multiple sensors monitor a given piece of equipment in case one fails and reports incorrect values).

- *Security requirements*

IIoT systems must be secure from attack, and it must be possible to keep the data they gather confidential. For example, a robot should not accept a command that is not authenticated, and a biometric sensor should not send medical data over a network until the data has been encrypted.

- **Communication For IIoT**
  A typical IIoT application involves many sensors generating data and multiple applications running in a hierarchy of edge and fog centers processing the data and (possibly) issuing commands that control the underlying devices.
- As the previous section points out, the system must meet specific requirements for Quality of Service.
- The question arises, “How should communication be arranged among all the pieces to achieve the desired goal?”
- One answer comes from the *Object Management Group* (*OMG*).
- An OMG standard known as the *Distributed Data Service* (*DDS*) defines a mechanism that allows data from sensors to flow upward through a hierarchy of edge and fog centers to applications using the data.

- The DDS approach has the following characteristics:
  Completely decentralized
  d Suitable for industrial use
  d Publish-Subscribe interactions
  d Flexible data handling capabilities
  d Support for an edge hierarchy
-

- *Completely decentralized*.
- Unlike communication systems that rely on a process to distribute data to subscribers, DDS avoids a single point of failure while minimizing latency by using direct communication. DDS can also use multicast transmission over a network to reach multiple applications efficiently.

Suitable for industrial use.

- DDS offers the high reliability needed for IIoT applications. It can be configured to meet performance requirements and to prioritize specific types of data. In addition, DDS offers the ability to authenticate control messages and encrypt data traffic.

Publish-Subscribe interactions.

- DDS offers a *publish-subscribe* communication mechanism that allows each application to choose the data the application will receive. An application running in the edge may choose to receive all data from a given sensor, and an application running in a fog center may choose to ignore the raw data and only receive periodic summaries.

Flexible data handling capabilities.

DDS provides flexibility that allows users to
specify filtering and QoS on a per-interaction basis. For example, suppose a sensor system publishes data. One subscriber may choose to receive all data while another chooses to receive only values beyond a specified threshold. Each subscriber can also negotiate QoS requirements independently.

Support for an edge hierarchy. DDS can be configured to form a distributed system where some of the participating applications run in an edge data center and others run in fog centers at higher levels of a hierarchy. The hierarchy can extend to include public clouds as the highest level.

DDS uses the term *Data bus* to describe the mechanism used to support publish subscribe communication. Multiple applications connect to a Data bus, and each application can choose to publish data and subscribe to data published by other applications.

- As the name implies, a Data bus provides interconnections among publishers and subscribers analogous to the way a hardware bus in a computer provides interconnections among I/O devices, memories, and processors.
- Unlike the conventional bus in a computer, a Databus is not a hardware mechanism. Instead, one can think of a Databus as a communication abstraction implemented by software. Software modules arrange to send messages between publishers and subscribers to meet requirements and optimize communication.
- In practice, Data bus software sends messages across underlying computer network(s) and other computer communication channels.
- Databus technology includes a mechanism known as a *gateway* that allows a single Databus to span multiple levels of the hierarchy. The system allows a given application to subscribe to data published at any level of the hierarchy, as if all applications connect to the same Databus. Rather than blindly sending a copy of every message, gateways perform a filtering function, and only forward a message to another level if one or more applications have subscribed to receive the message.
- As a result, the system provides the illusion of a single, large communication system, but eliminates unnecessary message propagation.
- Figure 16.5 illustrates a Databus that uses gateways to span multiple levels of a hierarchy. In the figure, applications connect at each level. In addition, the figure shows databases that can be used to cache information.