URJC Escuela de Ingeniería de Fuenlabrada

RoboticsLabURJC
Programming Robot Intelligence

Sistemas Distribuidos y Concurrentes

Introducción

Grado en Ingeniería de Robótica Software

Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación

Roberto Calvo Palomino
roberto.calvo@urjc.es

¿Qué es un sistema distribuido?

- Un sistema distribuido es un conjunto de sistemas o nodos **autónomos** que actúan de forma **transparente** actuando como un **único sistema** para el usuario

  - Nodos autónomos capaces de ejecutar independientemente
  - Dispositivos hardware o procesos software pueden ser nodos
  - Transparencia total para el usuario

You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done.

– Leslie Lamport --

¿Qué es un sistema distribuido?

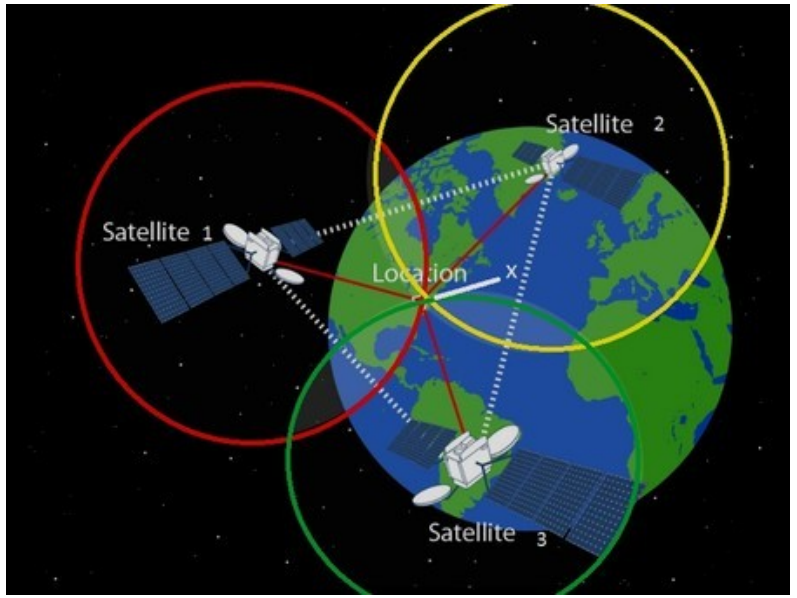
- Principal idea es para compartir servicios y recursos:
  - Datos
  - Hardware: CPU, memoria, almacenamiento
  - Aplicaciones software: procesadores de texto, visores
  - Servicios de localización
  - Robots: sensores y actuadores
- Eliminar cuellos de botella
- Reducir los puntos de error centrales del sistema

¿Qué es un sistema distribuido?

- Redes de telefonía
  - Antenas, torres de comunicación, cables, satélites, conmutadores
  - Orquestación
- Internet
  - WWW
  - red de pares (P2P)
- Aplicaciones
  - Gmail, Google Cloud

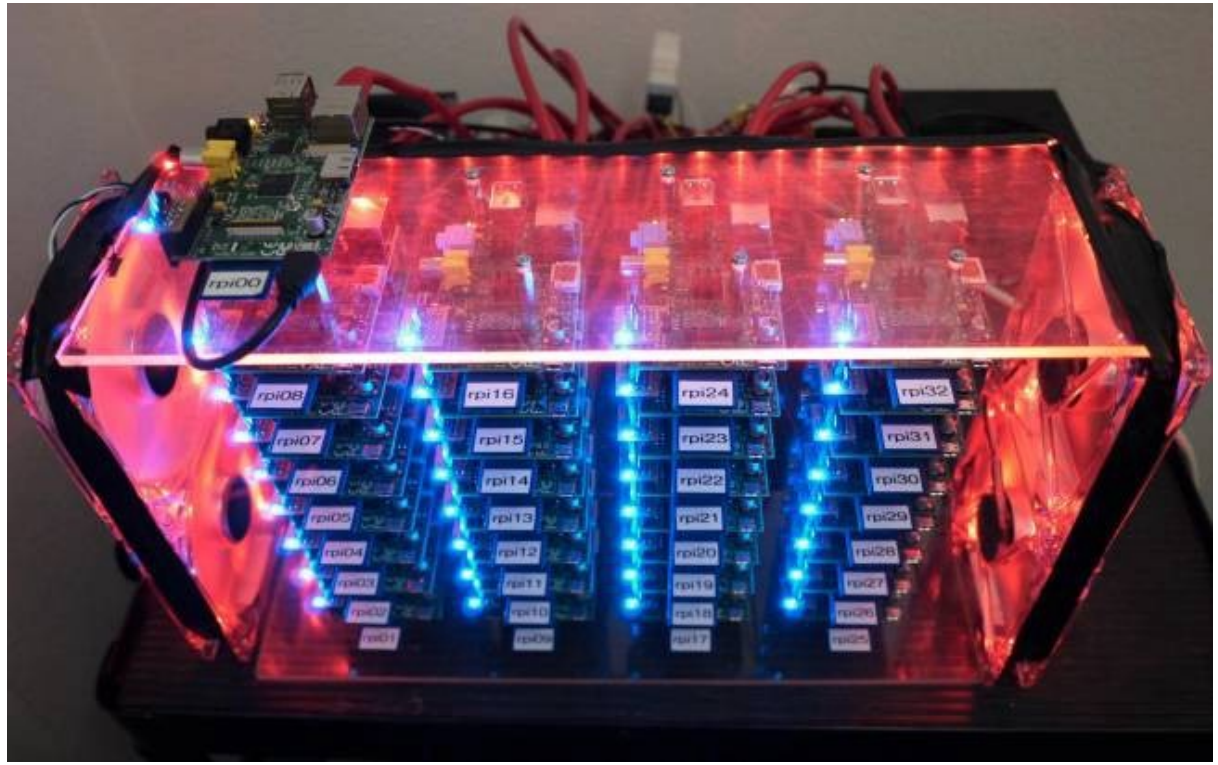
¿Qué es un sistema distribuido?

- GPS (Global Positioning System)
  - Actualmente en infinidad de dispositivos
    - Aviones, barcos, coches, smartphones, robots, etc ..
  - Constelación de 32 satélites orbitando a 20.000 km de altura
  - Velocidad media de 14.000 km/hora
  - Relojes atómicos en cada satélite
  - El usuario final ve todo el conjunto del sub-sistema como un único sistema,
    "Obtener posición geo-localizada"



¿Qué es un sistema distribuido?

- OctaPi
  - Cluster de computación basado en RaspberryPi
  - https://projects.raspberrypi.org/en/projects/build-an-octapi



¿Qué es un sistema distribuido?

- Robocup
  - Competición de fútbol con robots autónomos
  - Algoritmos distribuidos
  - Cada robot puede tener programado varios nodos

¿Qué es un sistema distribuido?

- Ciudad inteligente
  - Internet-of-Things (IoT) y 5G.
  - Sistemas distribuidos englobarán comunicaciones entre:
    - Satélites
    - Aviones
    - Coches
    - Dispositivos WiFi
    - ....

- El usuario final “solo” conduce su coche.

¿Qué es un sistema distribuido?

- Ventajas
  - **Económicas**: es más barato comprar muchos ordenadores pequeños que fabricar un super computador. Summit (IBM): 148,6 petaflops y 2,41 millones de nucleos.
  - **Fiabilidad**: algunos nodos pueden fallar, pero el sistema puede seguir funcionando.
  - **Crecimiento Incremental:** Se pueden añadir/actualizar recursos bajo demanda.
  - **Distribución Inherente**: algunas aplicaciones dependen de varias máquinas separadas por sus propias características (GPS).

Objetivo: Transparencia

- Transparencia de distribución: esconder el hecho de que el sistema está formado por distintos componentes. Si se consigue una transparencia total, se obtiene la ilusión de tener un único sistema (single-system image).
- Tipos:
  - Acceso
  - Localización
  - Migración
  - Relocalización
  - Replicación
    - Concurrencia
  - Fallo

Transparencia de Acceso

- Esconder los detalles sobre diferencias en la representación de los datos y sus mecanismos de acceso.

- Ejemplos:
  - Establecer un orden canónico para los bytes (Big Endian / Litte Endian)
  - Establecer una codificación de texto plano (UTF-8, ASCII)
  - Acceder a una interfaz homogénea para los datos, sin depender de que éstos procedan de sistema de ficheros, BBDD, sensores, etc.

Transparencia de Localización

- Esconder los detalles sobre la localización de los recursos mediante la asignación de nombres lógicos.

- Ejemplos:
  - Números de teléfono
  - */home/username/file*
  - http://www.urjc.es/etsit

Transparencia de migración y relocalización

- El usuario no se entera si el recurso se mueve de un componente a otro. Si esto pasa mientras que el usuario está usando el recurso, se habla de relocalización.

- Ejemplos:
  - El componente que sirve un sistema de ficheros cambia mientras que tenemos montado el volumen.
  - Los satélites GPS en visión directa cambian mientras obtenemos la localización.
  - La re-asignación de la torre de comunicaciones mientras hablamos con nuestro móvil.

Transparencia de replicación

- Se oculta que hay varias réplicas del mismo recurso. Requiere transparencia de localización.

- Ejemplo:
  - El usuario no es consciente que cada vez que actualiza una tabla de una BB.DD. se actualizan N réplicas.
  - El usuario no se da cuenta que sus peticiones van a la réplica del servicio que está más cercana en la red.

Transparencia de concurrencia

- Ocultar a los usuarios que componentes del sistema necesitan compartir ciertos objetos y deben cooperar para proporcionar el recurso conservando un estado coherente.

- Ejemplo:
  - El usuario no tiene que re-intentar una operación si la base de datos está siendo usada por otro componente en ese mismo instante.

Transparencia de Fallo

- El usuario no se entera si ciertos componentes del sistema han fallado. Enmascarar los fallos es complejo, a veces no es posible, y no siempre es apropiado.

- Ejemplo:
  - El sistema redirige una petición del cliente a un segundo servidor si el primero falla, el cliente recibe el resultado como si no hubiera fallado nada (no se retorna error).

Grado de transparencia

- ¿Qué grado de transparencia es deseable?
  - Problema: hay un compromiso entre transparencia y eficiencia.
  - Hay restricciones físicas (latencia, etc.)
  - No siempre conviene ofrecer un tipo de transparencia:
    - P. ej. para algunos servicios no se desea la transparencia de localización.
    - P. ej. la comprensibilidad del sistema puede verse reducida por la transparencia de fallo (ocultación de errores a las aplicaciones)

Objetivo: Sistema Abierto

Los recursos se sirven de una forma estándar, siguiendo una semántica y una sintaxis determinada para proporcionar **interoperabilidad**, **portabilidad**, y **extensibilidad**.

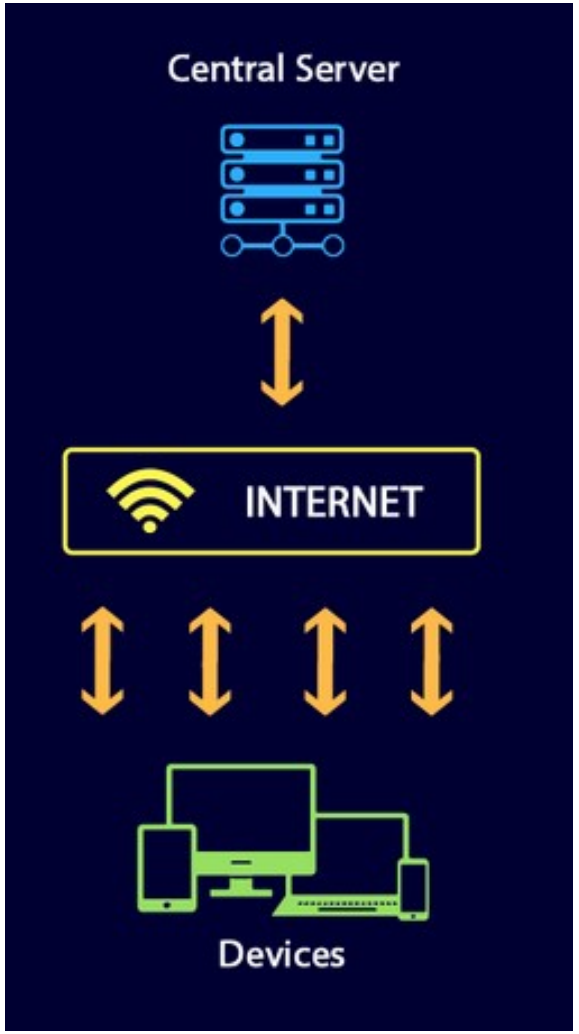
- Descripción de la interfaz del recurso.
- Formato y semántica de los mensajes.
- Convenios del sistema.
- Separación de políticas y mecanismos.

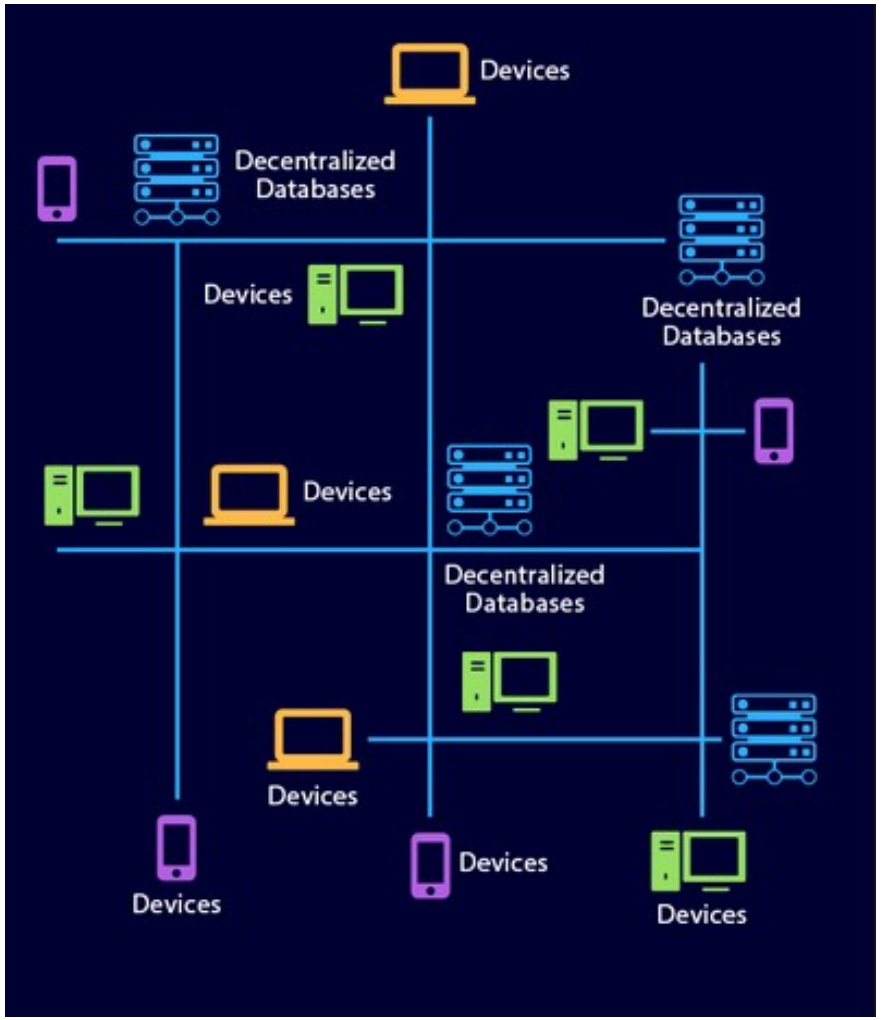
Objetivo: Escalabilidad

- El sistema distribuido debe poder soportar su crecimiento (usuarios, componentes, datos, etc.), su distribución **geográfica** y su **administración**.
- Un servicio centralizado tiende a convertirse en un cuello de botella que reduce la escalabilidad del sistema → replicación.
- Si los clientes del servicio están dispersos geográficamente, un servicio centralizado estará lejos de algunos clientes (p.ej. Akamai) → distribución.
- Un servicio centralizado es más sencillo de implementar. No se puede subestimar la simplicidad de un sistema.
- Un servicio distribuido es más difícil de asegurar.

Escalabilidad: datos

- Datos centralizados vs. Datos distribuidos





Escalabilidad: algoritmos

- **Algoritmo centralizado**: los nodos tienen que recolectar todos los datos sobre el estado del sistema para ejecutar el algoritmo. Esto puede no ser asumible.
- **Algoritmo distribuido**:
  - Ningún nodo tiene la información completa del sistema.
  - Los nodos toman decisiones en base a su estado.
  - El fallo de un nodo no arruina el algoritmo.
  - No se asume un reloj común exactamente sincronizado.

Ejemplos: consenso, commit atómico de transacciones, ordenación de eventos, elección de lider, exclusión mutua distribuida, etc.

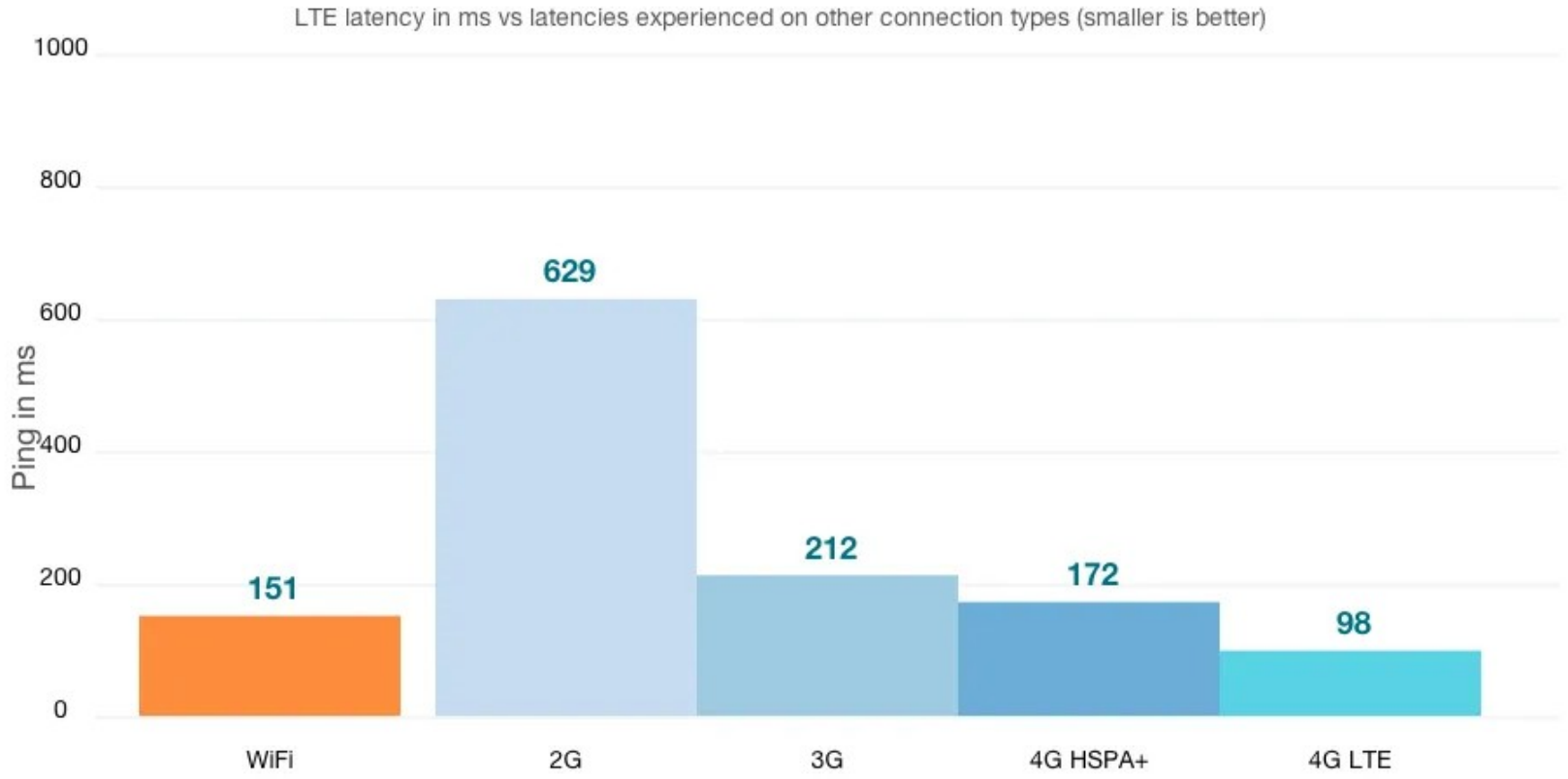
Escalabilidad: geográfica

- Principal problema: latencia*.
  - L1 cache reference : 0.5 ns
  - L2 cache reference : 7 ns
  - Mutex lock/unlock : 25 ns
  - Main memory reference : 100 ns
  - SSD random read : 150,000 ns = 150 us
  - Read 1 MB sequentially from memory : 250,000 ns = 250 us
  - Round trip within same datacenter : 500,000 ns = 0.5 ms
  - Read 1 MB sequentially from SSD* : 1,000,000 ns = 1 ms
  - Read 1 MB sequentially from disk : 20,000,000 ns = 20 ms
  - Send packet CA-Netherlands-CA : 150,000,000 ns = 150 ms
- Latencia entre nodos lejanos tiene un límite fijo, velocidad de la luz (300.000 km/s)
- Comunicaciones a larga distancia menos fiables

* https://gist.github.com/jboner/2841832

Escalabilidad: geográfica

- Latencia en sistemas telefonía móvil



- 5G: 1 ms teórico, ~20 ms en el mundo real

* https://www.cablefree.net/wirelesstechnology/4glte/lte-network-latency/

Escalabilidad: técnicas

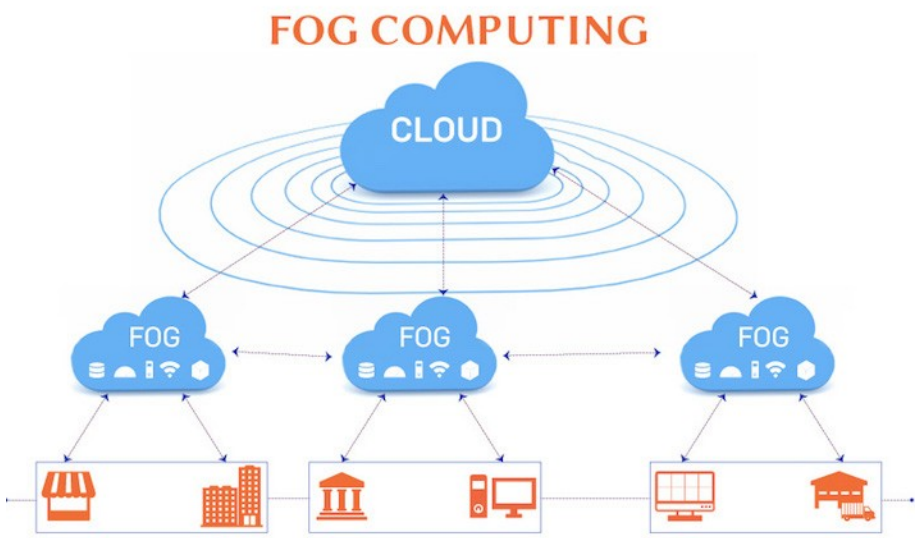
- Comunicación asíncrona en servicios no interactivos.
- Agrupación de operaciones (batching).
- Protocolos con pocos round-trips.
- Preprocesado en el cliente.
- Caching → coherencia de cache.

Edge Computing

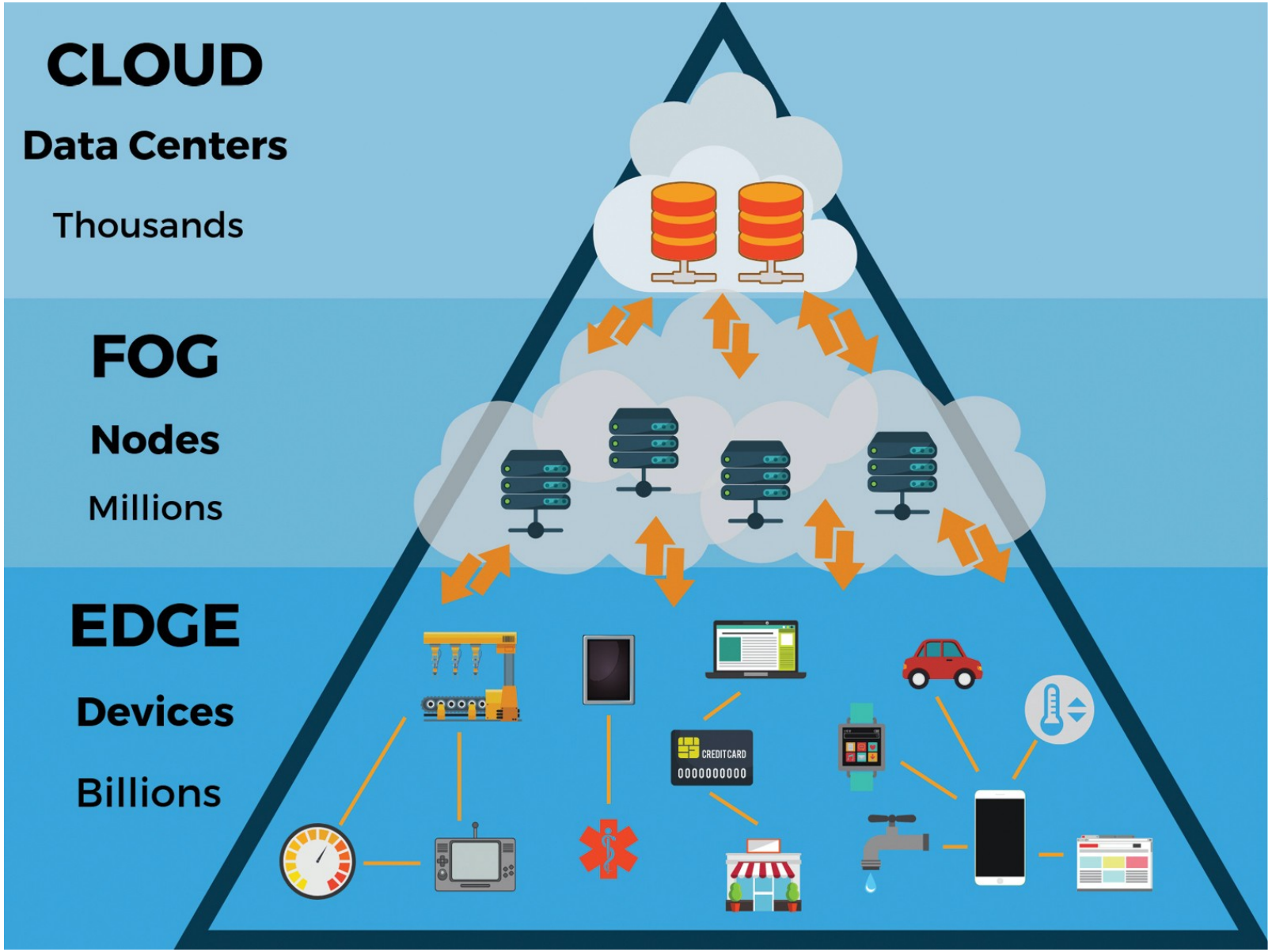
- Arquitectura tradicional propone que los datos recogidos por sensores se envíen a través de largos recorridos para que lleguen a **centros de datos y nubes de computación**.
- Dispositivos tienen un funcionamiento **pasivo**
- El concepto ***edge*** propone que los datos sean analizados por los propios sensores o por componentes que se encuentran cerca de los propios sensores (evitando el paso por la nube)
- Proporciona la ventaja de analizar datos en tiempo real.
- **Edge computing** se basa en cómo los procesos computacionales se realizan en (o cerca de) los dispositivos IoT que generan la información

Fog Computing

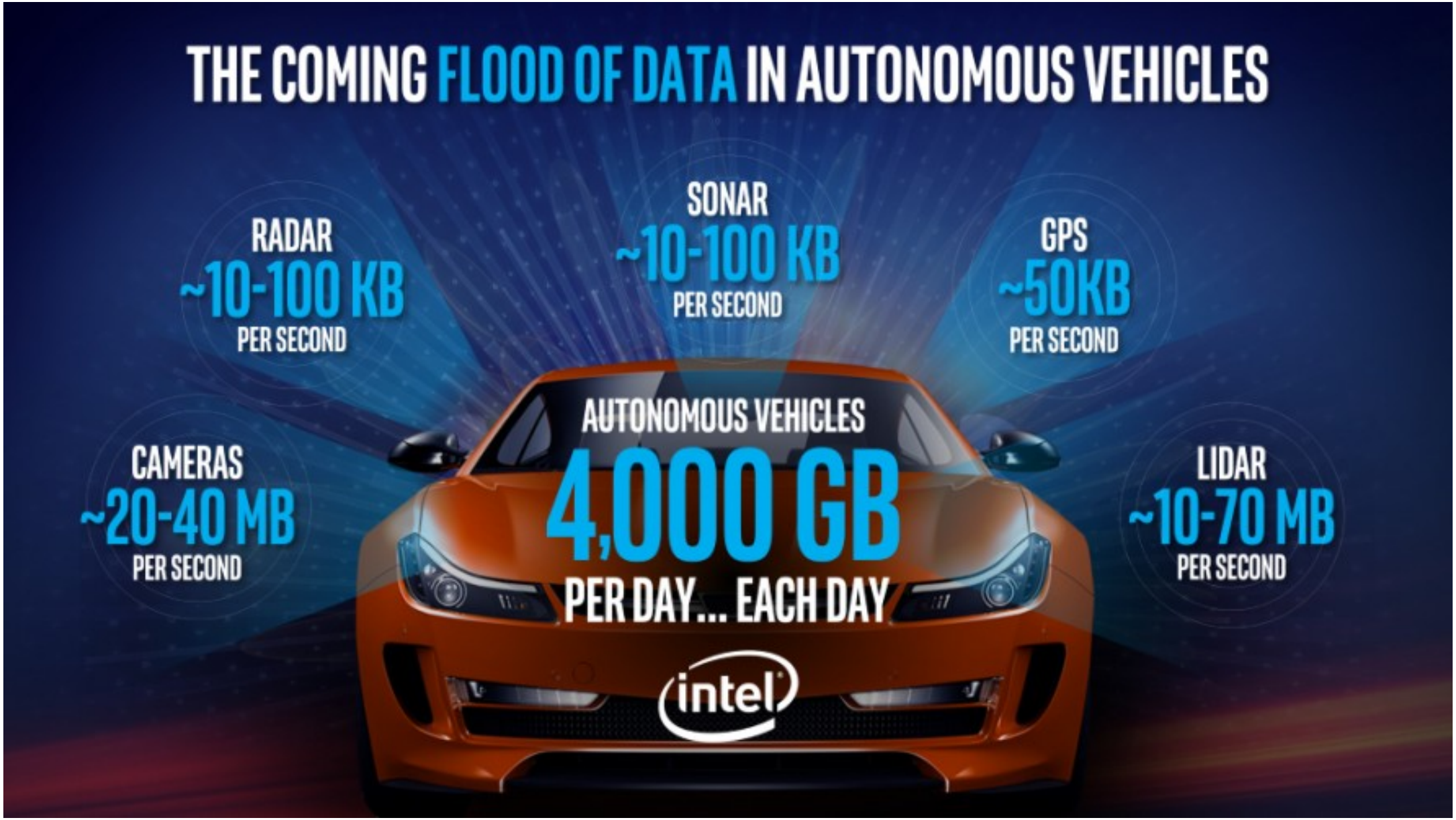
- El concepto Fog Computing se refiere a una estructura de **red descentralizada** en la que los recursos, incluyendo datos y aplicaciones, se sitúan en algún lugar lógico entre el Cloud y la fuente que genera los datos.
- Lleva la inteligencia al nivel de **red de área local** de la arquitectura de red.



Fog / Edge Computing



Coche autónomo



Middleware Distribuidos

- En un sistema software que permite comunicación entre procesos abstrayendo de toda la complejidad de comunicaciones, localización, hardware, etc.
- ONC RPC (Sun)
- CORBA
- Java (RMI)
- ICE (ZeroC)
- DDS

URJC Escuela de Ingeniería de Fuenlabrada