Today's Class

- Part 1: Pervasive Computing
- Part 2: Multimedia computing

Pervasive Computing

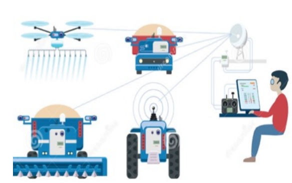
- Computing is becoming increasingly ubiquitous
- Sensing and computing "everywhere"
  - Increasingly part of physical environments
  - Enables many new application domains

Smart Health

Smart Buildings

Smart Transportation

Smart Agriculture



Internet of Things (IoT)

- Miniaturization of computing
  - Tiny sensors with computing and communication capability
  - MEMS: MicroElectroMechanical Systems
  - Expectation: Moore's law-like growth in MEMS
- Rise of internet of things
  - Network of Physical Devices
  - Ability to network devices and have them communicate
  - Large network of sensors

Smart Health

- Early Wearables devices
  - Fitness, exercise tracking
  - Sleep, heart rate, …
- New technologies emerging:

Smart Clothing

On-body monitoring

Smart Glasses

Gaze tracking, fatigue detection

Smart Buildings

- Proliferation of smart devices in homes

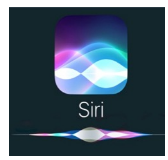
Thermostat

Smart Plug

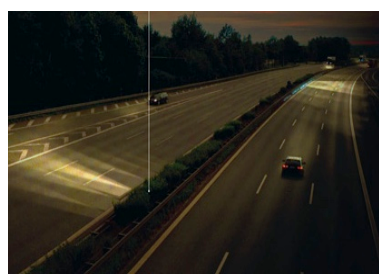
Smart Appliances

Smart Lock

- Phone and voice interfaces:



Smart Transportation



- Smart Roadways
  - Reactive Lights/Dynamic Lanes
  - Road Condition Monitoring
  - Traffic Management
- Connected Cars
  - Accident avoidance
  - Fleet Management
  - Real time public transport alerts

Typical smart app

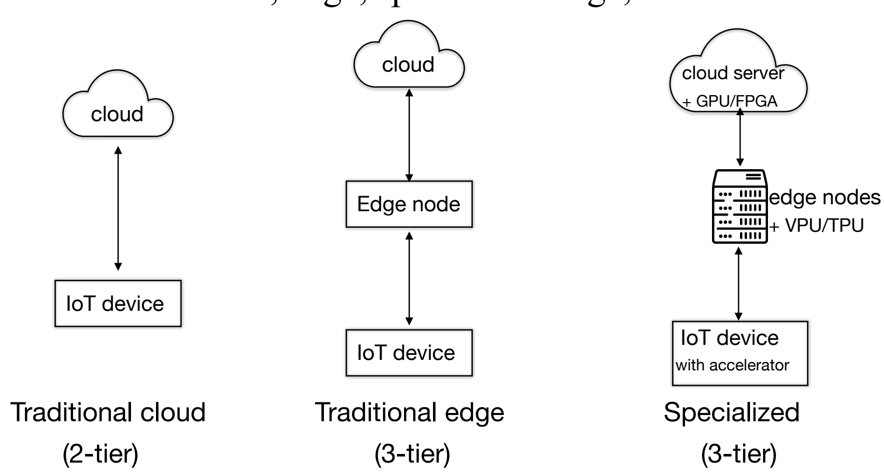
- Personal device to mobile phone to the cloud
  - Upload data to cloud via a mobile device (or directly)
  - Low-power communication to phone
  - Cloud provides analytics and provides feedback to phone

- Environmental sensors to internet to the cloud
  - Internet-enabled sensors
  - Upload to directly to servers / cloud through a router
  - Cloud provides analytics and provides dashboard

Computer Science

IoT Architectures

- Offload to cloud, edge, specialized edge,



Specialized Edge Computing

- Edge accelerators: special hardware to accelerate edge tasks on resource constrained edge servers
  - Nvidia Jetson GPU, Google edge Tensor processing Unit (TUP), Intel   Vision Processing Unit (VPU)
- Example: IoT ML inference on edge accelerators
  - Efficient inference on resource-constrained edge servers

Google Edge TPU

Nvidia Jetson Nano GPU

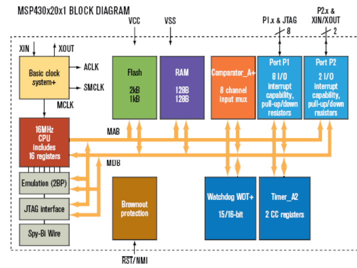
Apple Neural Engine

Sensor Platform

- Smart devices are a sensor node
- Resource-constrained distributed system
- Typical Sensor platform
  - Small CPUs
    - E.g. 8bit, 4k RAM
  - Low-power radios for communication
    - 10-200kbit/sec
  - Sensors
  - Battery driven or self-powered
  - Flash storage

Small CPUs

- Example: Atmel AVR
  - 8 bit
  - 4 KB RAM
  - 128 KB flash on-chip
  - ~8 mA
- Example: TI MSP430
  - 16 bit
  - 10 KB RAM
  - 48 KB flash
  - 2 mA



Higher-powered processors:

- ARM7 - 32 bit, 50 MHz, >>1MB RAM
- ARM9 - 32 bit, 400 MHz, >>16MB RAM

Low Power Radios

- Industrial, Scientific and Medical (ISM) Band
  - 900 MHz (33 cm), 2400 MHz (Bluetooth)
- Varying modulation and protocol
  - Zigbee (IEEE 802.15.4) – Modulating Phase
  - Bluetooth (IEEE 802.15.1) – Modulating Frequency
- Short range
  - Typically <100 m
- Low power. E.g. Chipcon CC2420:
  - 9-17 mA transmit (depending on output level)
  - 19 mA receive
- Listening can take more energy than transmitting

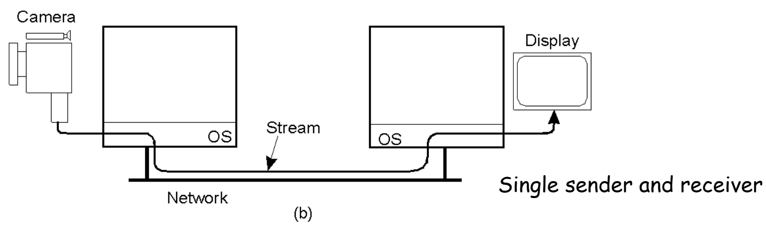
Sensors

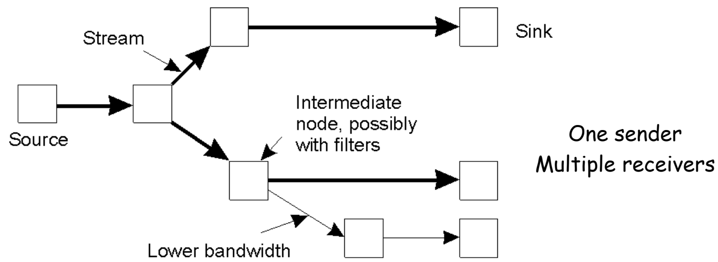
- Temperature
- Humidity
- Magnetometer
- Vibration
- Acoustic
- Light
- Motion (e.g. passive IR)
- Imaging (cameras)
- Accelerometer
- GPS
- Lots of others…

Multimedia Computing

- Message-oriented communication: request-response
  - When communication occurs and speed do not affect correctness
- Timing is crucial in certain forms of communication
  - Examples: audio and video ("continuous media")
  - 30 frames/s video => receive and display a frame every 33ms
- Characteristics of Video streaming
  - Isochronous communication
    - Data transfers have a maximum bound on end-end delay and jitter
  - Push mode: no explicit requests for individual data units beyond the first "play" request

Examples





Streams and Quality of Service

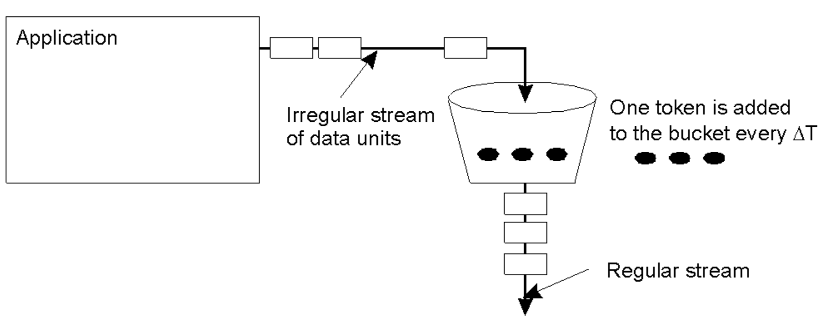
- Properties for Quality of Service:
- The required bit rate at which data should be transported.
- The maximum delay until a session has been set up
- The maximum end-to-end delay .
- The maximum delay variance, or jitter.
- The maximum round-trip delay.

Quality of Service (QoS)

- Time-dependent and other requirements are specified as *quality of service (QoS)*
  - Requirements/desired guarantees from the underlying systems
  - Application specifies workload and requests a certain service quality
  - Contract between the application and the system

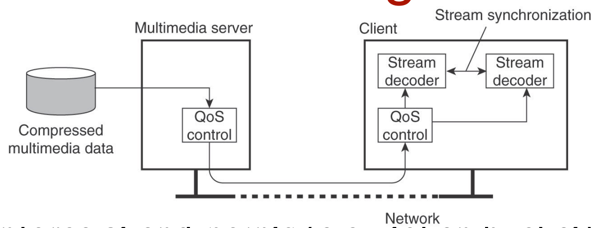
Characteristics of the Input	**Service Required**
•maximum data unit size (bytes) •Token bucket rate (bytes/sec) •Toke bucket size (bytes) •Maximum transmission rate (bytes/sec)	•Loss sensitivity (bytes) •Loss interval (μsec) •Burst loss sensitivity (data units) •Minimum delay noticed (μsec) •Maximum delay variation (μsec) •Quality of guarantee

Specifying QoS: Token bucket



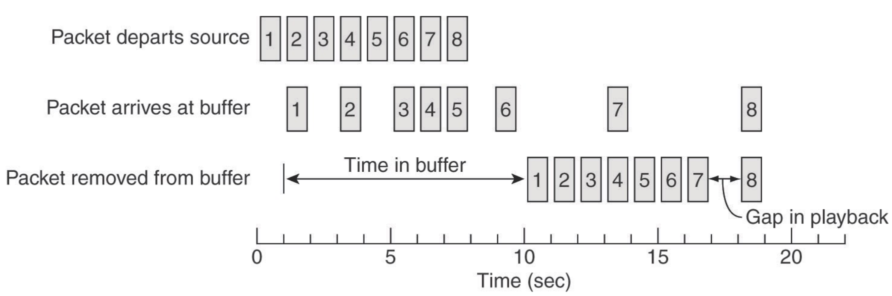
- The principle of a token bucket algorithm
  - Parameters (rate r, burst b)
  - Rate is the average rate, burst is the maximum number of packets that can arrive simultaneously

Enforcing QoS

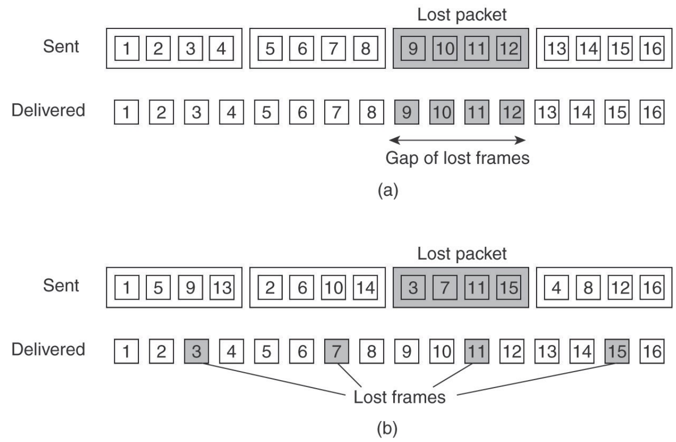


- Enforce at end-points (e.g., token bucket)
  - No network support needed
- Mark packets and use router support
  - Differentiated services: expedited & assured forwarding
- Use buffers at receiver to mask jitter
- Packet losses
  - Handle using forward error correction
  - Use interleaving to reduce impact

Enforcing QoS (1)



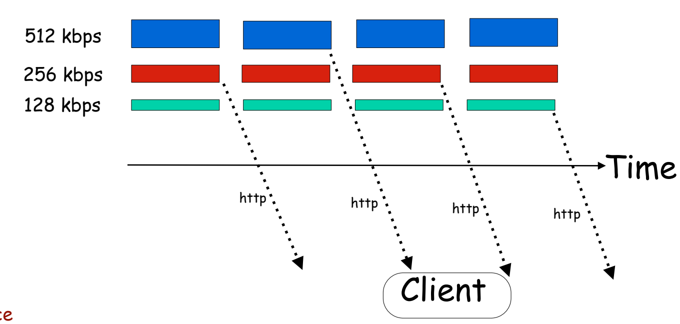
Enforcing QoS (2)



- Can also use forward error correction (FEC)

HTTP Streaming

- UDP is inherently better suited for streaming
  - Adaptive streaming, specialized streaming protocols
- Yet, almost all streaming occurs over HTTP (and TCP)
  - Universal availability of HTTP, no special protocol needed
- Direct Adaptive Streaming over HTTP (DASH)
  - Intelligence is placed at the client



Stream synchronization

- Multiple streams:
  - Audio and video; layered video
- Need to sync prior to playback
  - Timestamp each stream and sync up data units prior to playback
- Sender or receiver?
- App does low-level sync
  - 30 fps: image every 33ms, lip-sync with audio
- Use middleware and specify playback rates