Internet of Things and Fog Computing

Satish Srirama
satish.srirama@ut.ee
Outline

• Internet of Things
• Fog computing
• Fog computing challenges
• Fog placement strategies
• Future research directions
Internet of Things (IoT)

- IoT allows people and things to be connected
  - Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service

[European Research Cluster on IoT]

Image source: https://dyn.com/blog/what-needs-to-happen-before-iot-can-change-the-world/
History of IoT

- History: Internet-connected appliance - “The Only Coke Machine on the Internet” (Carnegie Mellon University, 1982)
- History: Internet for Things (Kevin Ashton, MIT Auto-ID Center, 1999)
  - RFID (Radio-frequency identification)
    - EPC (Electronic Product Code) Global Network
EPC Global Network

• Electronic Product Code (EPC)
Related Terms

• Cyber-Physical Systems (2006-)
  – Coined by National Science Foundation (U.S.)
  – Integrated embedded systems
  – Every physical thing embeds a networked ‘computer’

• Machine-to-Machine (M2M)
  – The root is much older than IoT (1970s)
  – Technologies to accelerate IoT
What is a *thing*?

- Can be a person with a heart monitor implant
- A farm animal with a biochip transponder
- An automobile that has built-in sensors
- Other natural or man-made objects
- With a **unique identifier** and the ability to communicate over the internet, without requiring human interaction.
Why it is so important?

- More connected devices than people
- Recent predictions go up to 500 billion devices by 2030
- Cisco believes the market size will be $19 trillion by 2025


05/05/2020
Layers of Cloud-centric IoT

Remote Cloud-based processing

Connectivity nodes & Embedded processing

Sensing and smart devices

[Srirama, CSIICT 2017]
Sensing and Smart Devices

• **IoT Devices**
  – Sensors and actuators
  – Motion, Temp, Light, Open/Close, Video, Reading, Power on/off/dimm etc.

• **Communication protocols**
  – Wireless and wired
  – Protocols such as ZigBee, Z-Wave, Wi-Fi/Wi-Fi Direct, Bluetooth etc.
  – Network congestion is a challenge

• **Arduino & Raspberry PI**
  – For rapid prototyping
Gateway/Connectivity Nodes

• Primarily deals with the sensor data acquisition and provisioning
• Embedded processing saves the communication latencies
• Predictive analytics
  – Collect data only occasionally
• Mobiles can also participate as Gateways
  – This brings in the scope of mobile web services and mobile cloud services for IoT
Mobile Cloud

• Harness cloud computing resources from mobile devices
• Mobile has significant advantage by going cloud-aware
  – Increased data storage capacity
  – Availability of unlimited processing power
  – PC-like functionality for mobile applications
  – Extended battery life (energy efficiency)

• Binding models [Flores and Srirama, JSS 2014]
  – Task delegation
    • Follows traditional SOA (Service Oriented Architecture) model to
      invoke cloud services
  – Mobile code offloading
    • Offload to a cloud based surrogate based on system and code profilers
      input
IoT Data Processing on Cloud

• Enormous amounts of unstructured data
  – In Zetabytes \((10^{21} \text{ bytes})\) by 2020 [TelecomEngine]
  – Has to be properly stored, analysed and interpreted and presented
• Big data acquisition and analytics e.g. with MapReduce and Spark
• In addition to big data, IoT mostly deals with big streaming data
  – Message queues such as Apache Kafka to buffer and feed the data into stream processing systems such as Apache Storm
  – Apache Spark streaming
IoT platform

• Middleware and the infrastructure that enables end-users to interact with smart objects
• Huge number of platforms are already available
• Features
  – Data storage
  – Data processing/Analytics
  – Visualization of data
  – Device management and Discovery
  – Support different communication protocols
  – Support heterogeneity
• Based on different architectures [Mineraud et al, CC 2016]
  – Mostly cloud-based
  – Centralized
  – Decentralized etc.
• E.g. Node-RED, FP7 OpenIoT, ThingSpeak, AWS IoT, ThingWorx, Cumulocity etc.
Issues with Cloud-centric IoT

- The cost of a cloud-only solution is too high
  - To run a large scale IoT system with >1000 geo-distributed devices [FogFlow]
- Latency issues for applications with sub-second response requirements
  - Health care scenarios
  - Smart cities and tasks such as surveillance need real-time analysis with strict deadlines
  - Many IoT services require <10ms end-to-end latency
- Network load
- Certain scenarios do not let the data move to cloud
  - Better security and deeper insights with privacy control
Fog Computing

- Processing across all the layers, including network switches/routers

[Chang et al, AINA 2017; FEC 2019; Mass et al, SCC 2016; Liyanage et al, PDCAT 2016]
Fog applications

- Multi-media applications
  - Finding best data streaming bit rate in different scenarios
    - Such as in video surveillance applications
    - Adjusting video encoding rate (video processing speed) based on network load
- Sensor data filtering and preprocessing
- Sensing frequency calibration
- Sensor data prediction
- Interactive games
  - Electroencephalogram (EEG) Tractor Beam game
  - Real-time EEG signal analysis and brain state (concentration) prediction

[Zao et al, Frontiers in Human Neuroscience 8:370]
Fog Architecture

Large-scale distributed data analytics
- e.g. MapReduce, Flink, Spark

Edge Analytics
- Filtering, consolidation, error detection, anonymization

Core Network
- Fog Nodes
  - Private Clouds, Cisco IOx, Switches, Cloudlets
- Gateways
  - Access points, Mobile Phones, Single Board Computers, CPE
- Edge Nodes
  - Sensors, Actuators, Mobile Phones, Smart Vehicles, ...

Cloud
Advantages of Fog Computing

• **Security**
  – Supports additional security to IoT devices to ensure safety and trustworthiness in transactions

• **Cognition**
  – Enables fog providers the awareness of the objectives of their clients toward supporting autonomous decision-making
    • E.g. where and when to deploy computing, storage, and control functions

• **Agility**
  – Brings the opportunity to individual and small businesses to participate in providing FEC services

• **Latency**
  – Provide rapid responses for the applications that require ultra-low latency

• **Efficiency**
  – Reduces the unnecessary cost of outgoing communication bandwidth
  – Consume minimum power for data offloading and processing when compared to CIoT model

[Chang et al, FEC 2019]
Types of fog computing applications

• Top-down applications
  – Applications that are managed by a cloud provider
  – Cloud provider deploys the necessary geo-distributed resources
  – Cloud providers manage the IoT application execution across the fog topology
  – Consider proximity and QoS parameters such as user load, latency, cost models etc.

• Bottom up applications
  – Applications that are managed by individual service providers
  – Fog resources are provided by different vendors
    • E.g. General public, private clouds, network operators
  – Gateways decide how to schedule the components across the fog topology
Fog Service Types

- **Storage (caching)**
  - Keep tenant-side client data locally
  - Temporary caching periodical data
  - E.g. Static local healthcare and eldercare

- **Compute (VM/ containers; IaaS/PaaS/FaaS / SaaS/ CaaS/ )**
  - Pre-processing data before sending to the cloud
  - Predefined rules deployed at the fog nodes

- **Acceleration (GPU/FPGA/APU etc.)**
  - Fog nodes with Tensor Processing Unit (TPU), AI Processing Unit (APU) etc.

- **Networking**
  - Support for Software-defined networking (SDN) / Network functions virtualization (NFV)

- **Control**
  - Deploy, actuating, mediating, security etc.
Research Challenges in Fog Computing

• Frameworks for establishing fog setups
• Fog resource provisioning
• Dynamic fog computing service discovery and accessing
• Fog execution frameworks
• Fog application placement
• Mobility of fog devices
• Etc.
IndieFog

- **Indie Fog** [Chang et al, IEEE Computer 2017]
  - System architecture for enabling fog computing with customer premise equipment

- **Proactive fog computing using resource-aware work-stealing** [Soo et al, IJMCMC 2017]

- **Dynamic fog computing service discovery and accessing**
Fog execution frameworks – Actor programming model

• Distributed and fault-tolerant execution of Fog computing applications
• Based on Actor programming model
• Actor model is conceived as a universal paradigm for concurrent computation [Carl Hewitt, 1973]
  – Basic unit of computation is an Actor
  – Actors communicate with other actors through messages
  – Actor can also create other actors (Parent-child hierarchy)
  – Designate the behavior to be used for the next message it receives
• Have implemented applications using the Akka framework [Srirama et al, FGCS 2020 (Under review)]
Fog execution frameworks – Serverless computing

• IoT workloads are a better fit for event driven programming
  – Execute app logic in response to sensor data
  – Execute app logic in response to scheduled tasks etc.
• Event-action platforms to execute code in response to events
• Applications are charged by compute time (millisecond) rather than by reserved resources
• Serverless computing is ideal solution for fog processing
  – OpenFaaS, light-weight enough to place on Raspberry Pi
Fog application placement strategies

• Resource intensive tasks of IoT applications can be placed across the Fog topology
• How to decide to which node the task to be offloaded?
  – Based on quality of service (QoS) parameters such as latency, resources, cost etc.
  – Latency and cost-aware [Mahmud et al, JPDC 2020] application module management
• The problem can also be formulated as multi-objective offloading strategy [Adhikari et al, IEEE IoTJ 2020]
  – Latency and resource management
  – Need to consider the types of the jobs and their priority
  – Need to find ideal heuristics, metaheuristics etc.
  – Also have to consider the graph topology of the Fog nodes
Application Offloading Strategy for Hierarchical Fog Environment through Swarm Optimization

- Multi-tier Fog-Cloud Model
- IoT layer
- Low-capacity fog layer
  - Resource constrained low capacity fog devices
  - Mainly process delay-sensitive applications
- High-capacity fog layer
  - Resource constrained high capacity fog devices
  - Process larger delay-sensitive applications or smaller computation-intensive applications
- Cloud layer
  - Can process any kind of computation-intensive applications

[Adhikari et al, IEEE IoTJ 2020]
QoS parameters – Cost and resource utilization

- Calculate cost of CPU and memory usage:
  
  \[ CO_{kl}^C = \frac{P_{kl} \times CO_{C}}{\tau_1}, \forall(k,l) \]
  
  \[ CO_{kl}^{mm} = \frac{P_{kl} \times CO_{mm}}{\tau_2}, \forall(k,l) \]
  
  \[ CO_{kl} = CO_{kl}^C + CO_{kl}^{mm}, \forall(k,l) \]

- Calculate communication cost:
  
  \[ CO_{kl}^{LD} = \frac{LD_{kl} \times CO_{LD}}{\tau_3}, \forall(k,l) \]

- Calculate total cost:
  
  \[ TCO_{kl} = CO_{kl} + CO_{kl}^{LD}, \forall(k,l) \]

- Calculate resource utilization:
  
  \[ RU_{kl}(t) = \frac{L_k(t)}{W_i(t)} \times 100, \forall(k,l) \]

- Formulate multi-objective function:
  
  \[ MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl}, \forall(k,l) \]

---

Algorithm 1 Representation of multi-objective Function

**INPUT:** \( U_i^C, U_i^{mm}, R_k^C, R_k^{mm}, S_k, CP_k^C \)

**OUTPUT:** \( MO_{kl} \)

1. for \( i: 1 \) to \( n \) do
   2. \( W_i(t) = \alpha \sum_{t_1}^{t_2} U_i^C(t) + (1 - \alpha) \sum_{t_1}^{t_2} U_i^{mm}(t) \)
3. end for
4. for \( k: 1 \) to \( K \) do
   5. \( L_k(t) = \alpha \sum_{t_1}^{t_2} R_k^C(t) + (1 - \alpha) \sum_{t_1}^{t_2} R_k^{mm}(t) \)
6. end for
7. for \( k: 1 \) to \( K \) do
   8. for \( l: 1 \) to \( n \) do
     9. if \( \sum_{k=1}^{K} L_k(t), x_{kl} \leq W_i(t) \) then
       10. \( RU_{kl} = \frac{L_k(t)}{W_i(t)} \times 100 \)
     end if
   12. \( CO_{kl} = CO_{kl}^C + CO_{kl}^{mm} \)
   13. \( CO_{kl}^{LD} = \frac{LD_{kl} \times CO_{LD}}{\tau_3} \)
   14. Total Cost \( TCO_{kl} = CO_{kl} + CO_{kl}^{LD} \)
   15. Function \( MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl} \)
16. end for
17. end for
Metaheuristics – Accelerated Particle Swarm Optimization

• **PSO**
  – Designed based on the behavior of bird or fish schooling in nature
  – Searches the space of an objective function by adjusting the position of the particles in a quasi-stochastic manner
  – Finds an optimal particle based on two parameters: Velocity and Position
  \[
  v_i(t+1) = v_i(t) + \alpha \varepsilon_1 (G^* - X_i(t)) + \beta \varepsilon_2 (X^*(t) - X_i(t))
  \]
  \[
  X_i(t+1) = X_i(t) + v_i(t+1)
  \]

• **APSO**
  – No compelling to use the individual best unless the optimization problem is highly multimodal or nonlinear
  – APSO uses the global best location of a particle to accelerate the convergence speed of the problem
  \[
  v_i(t+1) = v_i(t) + \alpha \varepsilon + \beta (G^* - X_i(t))
  \]
  \[
  X_i(t+1) = (1 - \beta) X_i(t) + \beta G^* + \alpha \varepsilon
  \]
Quality of Service (QoS) vs Experience (QoE)

- QoS - refers to the overall features of system services which help to meet the stated and implied needs of the end users [ITU]
  - QoS drives through an agreement between user and provider that strongly monitors technical attributes of system services
  - Cost, service delivery deadline, packet loss ratio, jitter, throughput, etc.
  - E.g. Downloading a particular file in max 5 min time
- QoE is the total acceptability of a service that is determined by subjective perception of the end users [ITU]
  - Encapsulates user’s requirement, intentions and perceptions while provisioning system services
  - E.g. 2 users require that file in 3 min and 7 min respectively
- End users perceived QoE can degrade the acceptability of a service greatly even when the proper QoS is maintained

ITU - International Telecommunication Union
Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments

- Fuzzy logic based approach that prioritizes different application placement requests
  - Based on the user expectations (*Rating of Expectation*)
- Fuzzy logic based approach that classifies Fog computational instances
  - Based on current status of the instances (*Capacity Class Score*)
- A linearly optimized mapping of application placement requests to Fog computing instances
  - To ensure maximized QoE-gain of the user

[Mahmud et al, JPDC 2019]
System model

• Fog-enabled IoT applications are divided into multiple interconnected Application Modules
  – E.g. Client Module and Main Application Module

• Fog nodes are organized in layers
  – Fog Gateway Nodes (FGN) and Fog Computational Nodes (FCN)
  – Provide RESTful services for querying and provisioning computational facilities
Calculating Rating of Expectation (ROE)

- Each application has certain parameters
- Range and unit of the parameters are not the same
  - So normalized to the interval \([-1, 1]\)
  - Parameter \(x\) values in range \([\alpha_x, \beta_x]\).
- Parameters and fuzzy sets
  - Access rate: \(Ar \in \{\text{Slow, Normal, Fast}\}\)
  - Required resources: \(Rr \in \{\text{Small, Regular, Large}\}\)
  - Processing time: \(Pt \in \{\text{Stringent, Moderate, Flexible}\}\).
- Fuzzy rules for ROE calculation
- Defuzzification using discrete center of gravity
Calculating Capacity Class Score (CCS)

- Parameters and fuzzy sets
  - Round trip time: $\text{Rtt} \in \{\text{Short, Typical, Lengthy}\}$
  - Resource availability: $\text{Ra} \in \{\text{Poor, Standard, Rich}\}$
  - Processing speed: $\text{Ps} \in \{\text{Least, Average, Intense}\}$.

- Discrete center of gravity

\[
\tau_{in} = \frac{\sum_{k=1}^{k=j} \mu'_c(f_{in}^k) \times \Phi_k}{\sum_{k=1}^{k=j} \mu'_c(f_{in}^k)}. 
\]
Mapping of applications to Fog instances

- Rating gain = ROE X CCS
- Total Rating Gain of the applications should be maximized

\[
\max \sum_{a_m \in A_m} \sum_{n \in N} \sum_{i_n \in I_n} z_{i_n}^{a_m} (\eta_{a_m} \times \tau_{i_n})
\]

subject to,

\[
\sum_{a_m \in A_m} z_{i_n}^{a_m} = 1; \forall n \in N, \forall i_n \in I_n
\]

\[\delta_{a_m} \leq Q_{\delta}\]

\[\zeta_{a_m} \leq Q_{\zeta}\]

\[\rho_{a_m} \leq Q_{\rho}\]

Maintain QoS - Service delivery time, service cost, and packet loss rate

Simulation parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation metrics:</td>
<td></td>
</tr>
<tr>
<td>Access rate</td>
<td>2-10 per sec</td>
</tr>
<tr>
<td>Resource requirement</td>
<td>1-8 CPU cores</td>
</tr>
<tr>
<td>Processing time</td>
<td>30-120 ms</td>
</tr>
<tr>
<td>Status metrics:</td>
<td></td>
</tr>
<tr>
<td>Round trip time</td>
<td>100-600 ms</td>
</tr>
<tr>
<td>Resource availability</td>
<td>1-10 CPU cores</td>
</tr>
<tr>
<td>Processing speed</td>
<td>10-70 TIPS</td>
</tr>
<tr>
<td>Applications service delivery deadline</td>
<td>250-750 ms</td>
</tr>
<tr>
<td>Data signal loss rate of the network</td>
<td>3%-5%</td>
</tr>
<tr>
<td>Service cost</td>
<td>0.1-0.15 $ per min</td>
</tr>
<tr>
<td>Number of accessible FCN per FGN</td>
<td>4-10</td>
</tr>
<tr>
<td>Data signal size</td>
<td>1000-2000 instructions</td>
</tr>
</tbody>
</table>
DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing

• Task Priority 1:
  – Class that supports the delay-sensitive tasks with hard deadlines
  – No negotiation on deadline
• Task Priority 2:
  – Tasks have intermediate priority with soft-deadline
  – Tasks meet their deadline with a negotiation and penalty
• Task Priority 3:
  – Lowest priority class that aims to support the resource-intensive tasks
  – No deadline
  – Mostly offloaded to the cloud

• Goal: To minimize the overall queueing waiting time and offloading time of the real-time tasks while meeting the deadline and resource

[Adhikari et al, IEEE IoTJ 2020]
HeRAFC: Heuristic Resource Allocation and Optimization in MultiFog-Cloud Environment

- The infrastructure and applications are modelled using graph theory
- FCI (Fog-Cloud Interface) is introduced, where a fog node can communicate with cloud and other fog nodes as well
- A task can be assigned to other fog nodes at multi-hop distance
- Heuristic approach is followed to achieve the objective of resource utilization maximization

[Dehury and Srirama, TSC 2020 (Under review)]
Fog Computing – Research Challenges
- continued

• Process-driven Edge Computing in Mobile IoT
  [Mass et al, IoTJ 2019; CASA 2018; Chang et al, CSUR 2016]
Fog Computing – Research Challenges - continued

- Mobility also becomes critical in Fog computing [Mass et al, IoTJ 2019]

  - Extended the ONE simulator to simulate the Fog computing mobility aspects
  - Process execution based on Flowable BPMS
STEP-ONE

[Mass et al, JSS 2020]
Future Research Scope

• Standards-based dynamic deployment and management of fog applications
  – Deployment should consider heterogeneity, mobility aspects, multiple ownership (interoperability) etc.
  – Cloud computing community addressed similar problem through OASIS TOSCA
  – Extending TOSCA to support fog-based applications
  – Develop relevant orchestrators

• Efficient algorithms for self-adaptive resource provisioning and QoS-aware application module scheduling in fog topology

• Reliable and data-locality preserving real-time data processing in fog environments
  – Controlling the dataflow of distributed data processing applications across the multi-layer fog
  – Restructure the IoT applications into dynamically composable multi-stage data pipelines
  – Actor programming model and serverless functions edge analytics tasks
  – Apache Nifi data pipelines for task composition

• IoT case studies in domains such as smart city, smart healthcare, smart agriculture etc.
Next lecture

• A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade
References