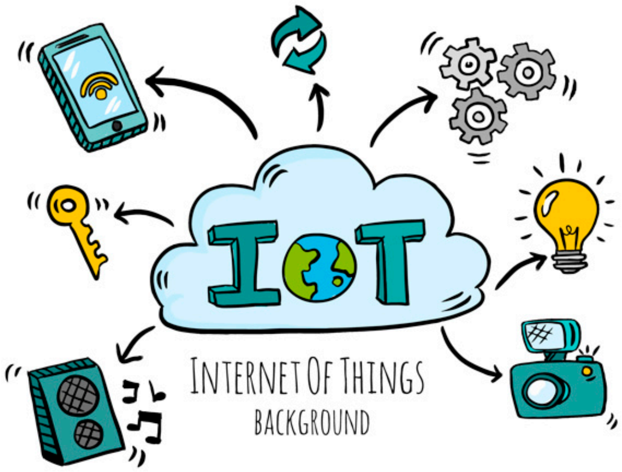
THE FOUNDATION OF FUTURE INTERNET OF THINGS SYSTEMS

Presented by : Michael Zhang

March, 1st, 2018

What is “Internet of Things”?

- “**Things**” refer to a wide variety of physical devices, including **sensors, controllers and actuators.**
- “**Internet**” connects these physical devices and enables the **exchange of data.**
- **Cloud computing** provides the backbone infrastructure of Internet of Things.
  - *Remote & Edge Computing Resources*
  - *Optimization by Provisioning & Offloading*
  - *Large-scale Deployment & Data Analytics*
  - *Identity Management & Security*



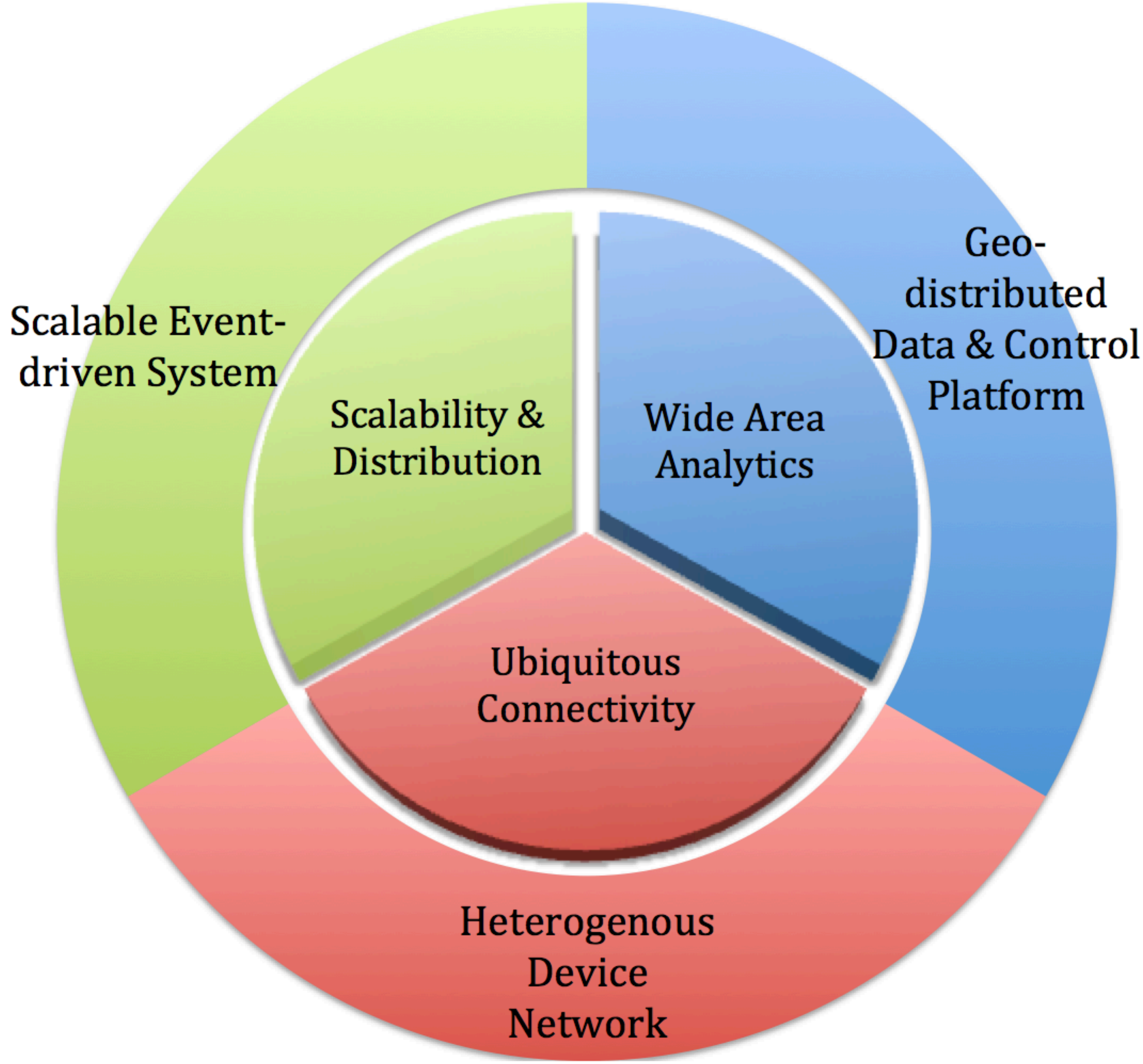
Challenges

- **Scalability & Distribution**
  - Overhead of thread-per-request currency model
  - Unsuitable monolithic deployment for IoT system
  - Optimization and debugging of functional programming language on Event-driven Architecture

- **Ubiquitous Connectivity**
  - Heterogeneity of devices in software and hardware
  - The demand of seamless and spontaneous interaction across IoT system

- **Wide Area Analytics**
  - Latency from round-trip time among geo-distributed edge cloud clusters and data centers
  - Large-scale deployment in the real world and socioeconomic acceptance

The foundation of future IoT System

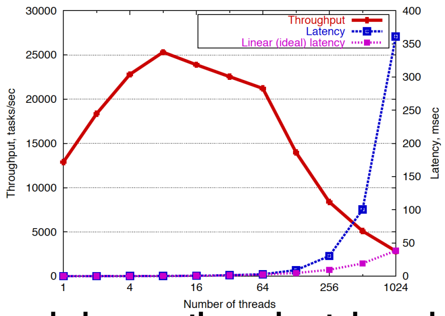


Challenges vs Opportunity

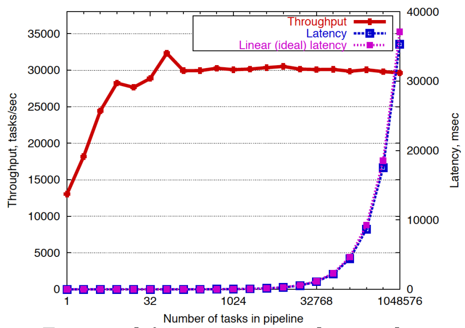
Challenges	Research Opportunity
Scalability & Distribution	**Scalable Event-driven System**
Ubiquitous Connectivity	Heterogeneous Device Network
Wide Area Analytics	Geo-distributed Data & Control Platform

SEDA: An Architecture for Well-Conditioned Scalable Internet Services

- **Problem**
  - Massive concurrent requests from Internet
  - Thread-based concurrency latency
  - Complex services with dynamic contents
- **Contribution**
  - Staged Event-Driven Architecture (SEDA) provides a scalable event-driven programming model
  - Control mechanisms for automatic tuning and load conditioning

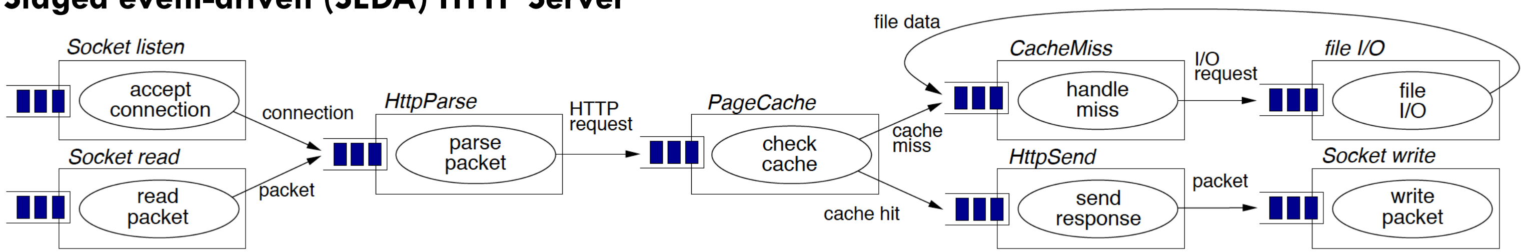


Threaded server throughput degradation



Event-driven server throughput

Staged event-driven (SEDA) HTTP Server



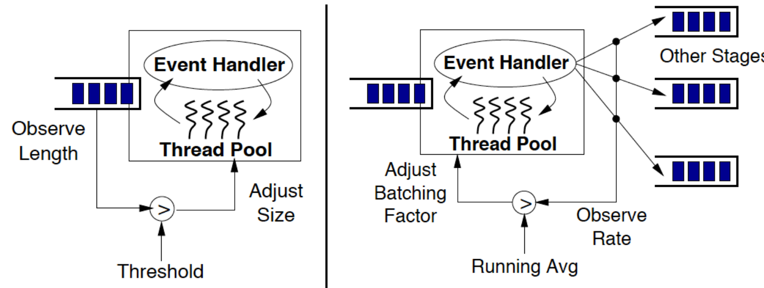
Welsh, Matt, David Culler, and Eric Brewer. "SEDA: an architecture for well-conditioned, scalable internet services." *ACM SIGOPS Operating Systems Review*. Vol. 35. No. 5. ACM, 2001.

SEDA: An Architecture for Well-Conditioned Scalable Internet Services

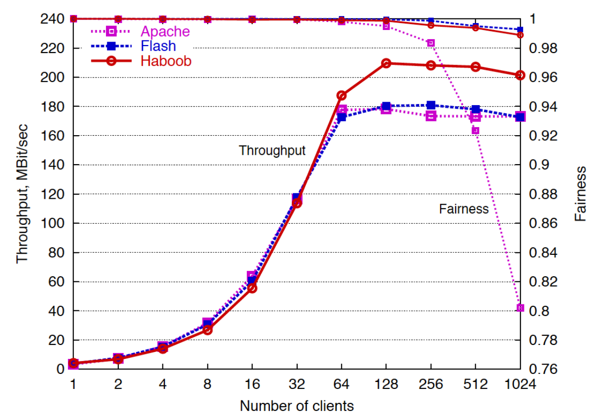
- **Dynamic Resource Controller**
  - Thread pool controller
  - Batching controller
- **Application & Evaluation**
  - Sandstorm
    - A SEDA-based Internet service platform
  - Gnutella packet router
  - Haboob
    - A high-performance HTTP server
    - Fairness:

$$f(x) = \frac{(\sum x_i)^2}{N \sum x_i^2}$$

Xi – The number of fulfilled request for each of N clients



(a) Thread pool controller (b) Batching controller



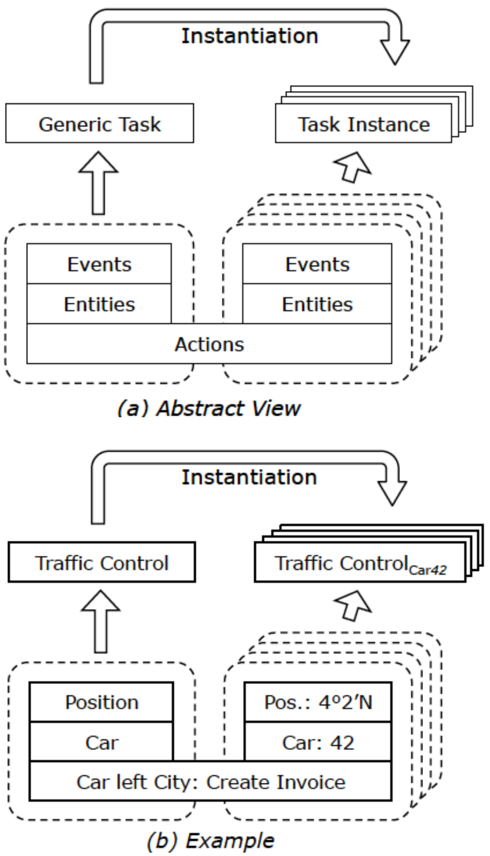
(a) Throughput vs. number of clients

Haboob performance of throughput and fairness (cf. Apache & Flash)

Welsh, Matt, David Culler, and Eric Brewer. "SEDA: an architecture for well-conditioned, scalable internet services." *ACM SIGOPS Operating Systems Review*. Vol. 35. No. 5. ACM, 2001.

Eventlets: Components for the Integration of Event Streams with Service-Oriented Architecture (SOA)

- **Problem**
  - Most Internet of Things systems lack power of processing event streams.
  - Push-based (Asnyc) event stream in IoT system differs from pull-based (Sync) in traditional SOA.
    - A service is an independent functionality that can be acted upon and updated remotely
    - SOA is an design where services are provided by application components through protocol over a network.
- **Contribution**
  - Eventlet - A generic and reusable component model for encapsulating event stream processing logic.
  - A distributed architecture of Eventlet middleware.



Generic Task Model

Appel, Stefan, et al. "Eventlets: Components for the integration of event streams with SOA." *Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference on.* IEEE, 2012.

Eventlets: Components for the Integration of Event Streams with Service-Oriented Architecture (SOA)

- **Structure**
  - Eventlet Prototype
    - Eventlet Metadata
    - Eventlet runtime code
  - Eventlet Middleware

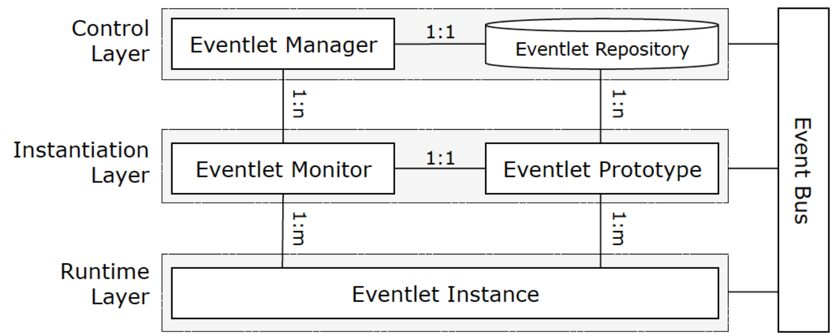
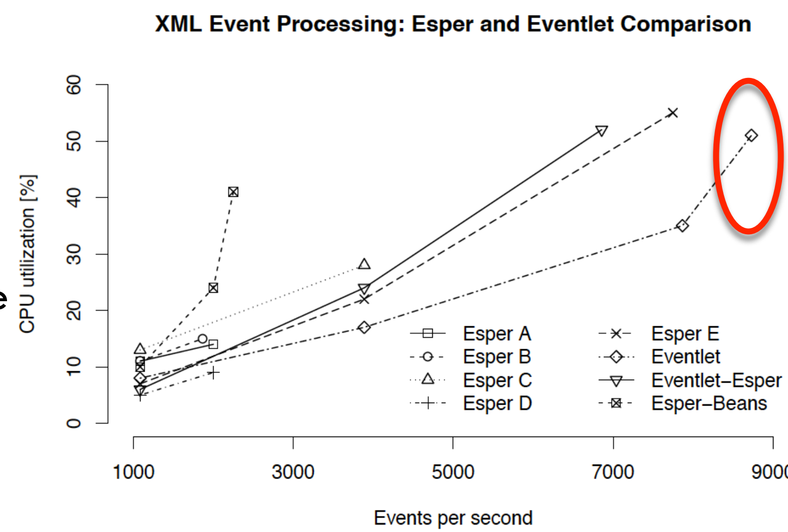


Fig. 3: Eventlet Middleware Components

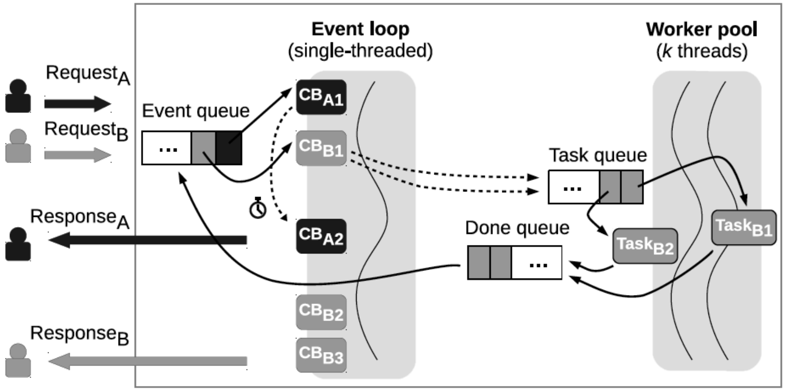
- **Implementation & Evaluation**
  - Java Message Service (JMS) as Event Bus
  - Java Reflection for instantiation
  - Esper for Complex Event Processing (CEP)
  - Measurements
    - CPU Utilization – Eventlet reaches the highest event rate given a certain level of CPU usage
    - Program complexity
      - Eventlet-Esper (2 classes, 41 LOC)
      - Esper-Beans (21 classes, 157 LOC)



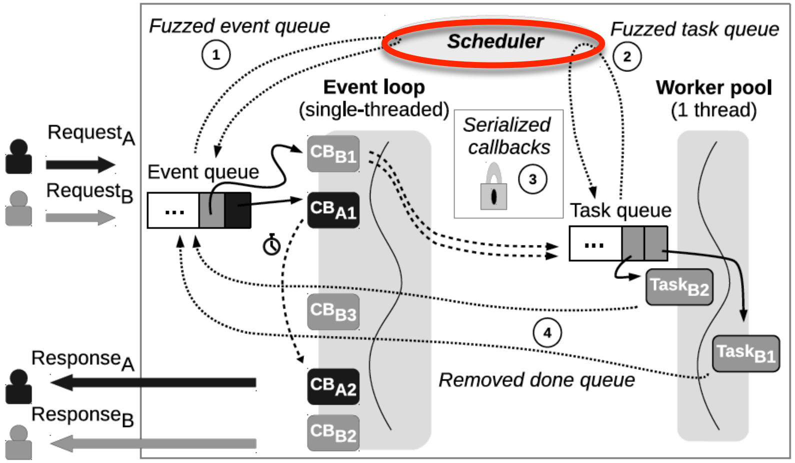
Appel, Stefan, et al. "Eventlets: Components for the integration of event streams with SOA." *Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference on*. IEEE, 2012.

Node.fz: Fuzzing the Server-side Event-Driven Architecture

- **Problem**
  - Concurrency bugs in server-side event-driven applications (EDA)
  - Limitations of existing concurrency error detection tools
- **Contribution**
  - Concurrency bug characteristics study
  - Node.fz – A fuzz testing tool for server-side EDA
    - Fuzzing is a testing technique of injecting random and unexpected data to a program that takes structured input.
  - Evaluation of Node.fz with real-world Node.js applications



Event-Driven Architecture event loop and worker pool



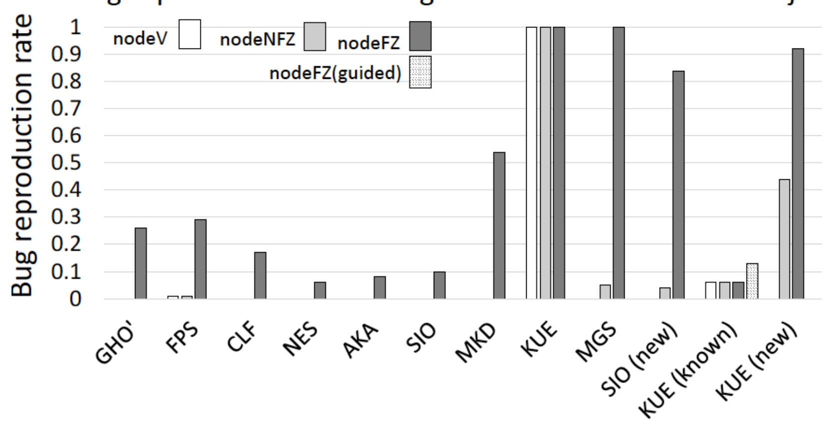
Callback orderings fuzzed by the scheduler

Davis, James, Arun Thekumparampil, and Dongyoon Lee. "Node. fz: Fuzzing the server-side event-driven architecture." *Proceedings of the Twelfth European Conference on Computer Systems.* ACM, 2017.

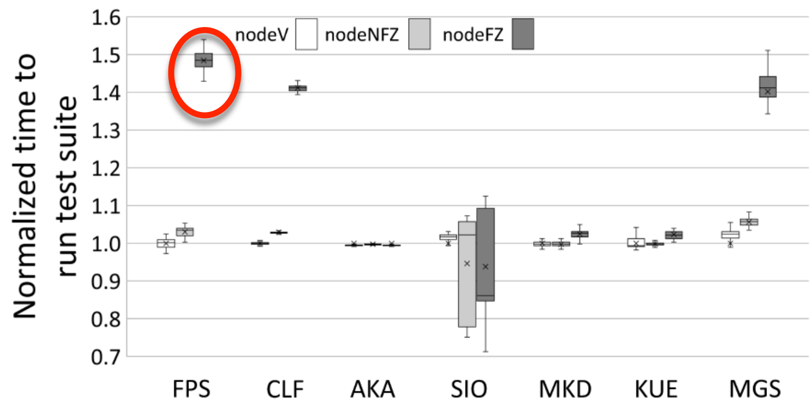
Node.fz: Fuzzing the Server-side Event-Driven Architecture

- **Concurrency bug study**
  - Atomicity violations
    - Two consecutive operations interleaved by another operation that affects the result
  - Ordering violations
    - The order of a series of consecutive operations are not enforced
- **Node.fz Design**
  - Control of three queues
    - Event, Task and Done
  - Shuffle the entries to yield various event arrival orders, not the events themselves
- **Evaluation**
  - Reproducing bug rate – Bug / Trials
  - The highest overhead ~ 50%

Bug reproduction rate using different versions of Node.js



Normalized test time using different versions of Node.js



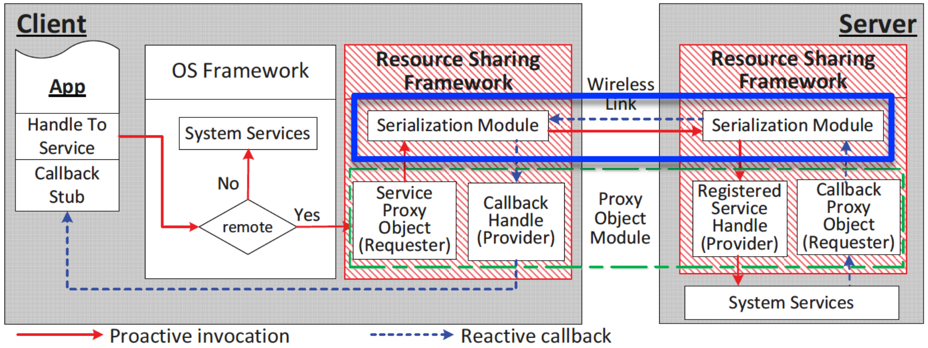
Davis, James, Arun Thekumparampil, and Dongyoon Lee. "Node. fz: Fuzzing the server-side event-driven architecture." *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, 2017.

Challenges vs Opportunity

Challenges	Research Opportunity
Scalability & Distribution	Scalable Event-driven System
Ubiquitous Connectivity	**Heterogeneous Device Network**
Wide Area Analytics	Geo-distributed Data & Control Platform

Interconnecting Heterogeneous Devices in the Personal Mobile Cloud

- **Problem**
  - Restrictions on resources of mobile devices
    - Computational power
    - Energy consumption
  - Heterogeneity of mobile devices in both software and hardware aspects
- **Contribution**
  - Develops a resource sharing framework that interconnects heterogeneous devices
    - i.e. Computational offloading of personal devices sharing GPS information
  - Implements with diverse characteristics and resource limits
  - Demonstrates the low overhead and small modification to underlying OS



Design of resource sharing framework

Li, Yong, and Wei Gao. "Interconnecting heterogeneous devices in the personal mobile cloud." *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017.

Interconnecting Heterogeneous Devices in the Personal Mobile Cloud

- **Design**
  - Resource sharing framework – Interacts with local and remote OS services
  - Application interface – Masks software and hardware heterogeneity through metadata file and service handle
- **Implementation & Evaluation**
  - Various supported services
    - GPS, Sensor, Audio, Camera
  - Measurements
    - Latency, power consumption, throughputs

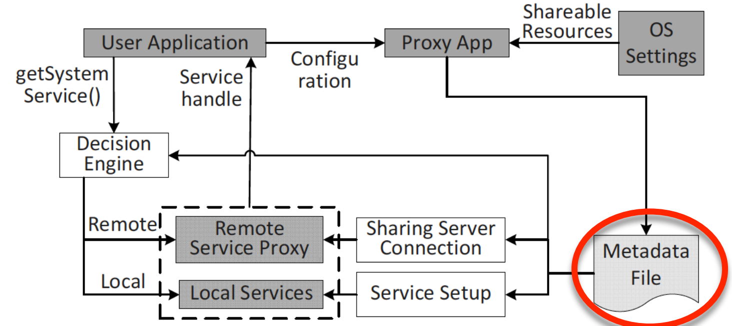


Fig. 4. Design of application interface

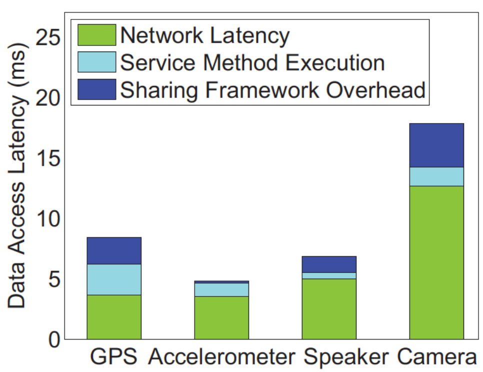


Fig. 9. Latency of accessing shared resources

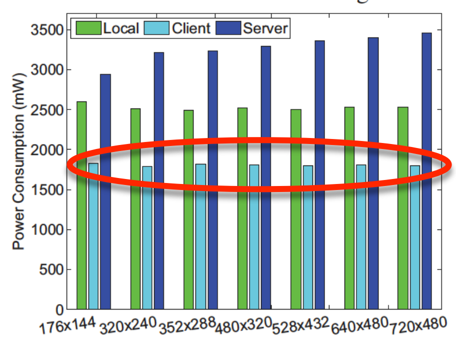


Fig. 13. Power consumption for camera sharing

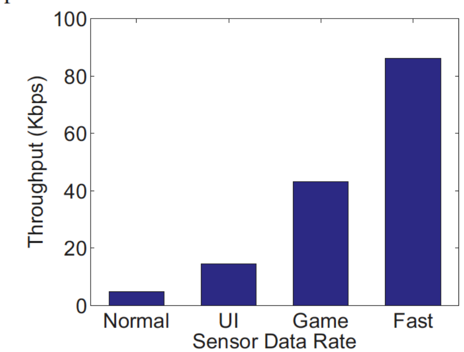


Fig. 15. Throughput for sensor sharing

Li, Yong, and Wei Gao. "Interconnecting heterogeneous devices in the personal mobile cloud." *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017.

Enabling Synergy in IoT: Platform to Service and Beyond

- **Problem**
  - IoT system needs seamless and spontaneous interactions among heterogeneous "things"
- **Contribution**
  - A system architecture model to explore synergy in IoT
    - Synergy – The aggregation of functionalities from self-assembling IoT devices
    - Hardware, Firmware, Person-where, Local-where, Wide-where
  - A practical IoT platform based on such system architecture

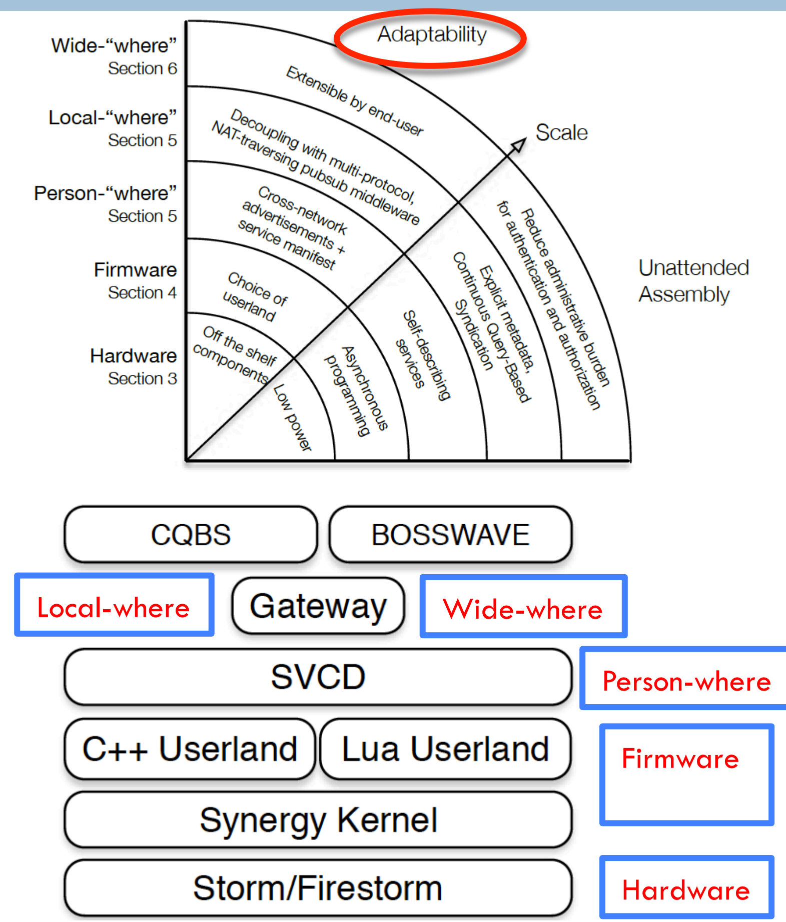


Figure 2: An IoT system architecture

Andersen, Michael P., Gabe Fierro, and David E. Culler. "Enabling synergy in iot: Platform to service and beyond." *Journal of Network and Computer Applications* 81 (2017): 96-110.

Enabling Synergy in IoT: Platform to Service and Beyond

- **Hardware**
  - Firestorm Platform
    - Device-to-Device, Device-to-Internet, Device-to-Person
    - Bluetooth Low Energy protocol (BLE) & Wi-Fi
- **Firmware**
  - Handles disjunction between the **sequential** program logic and **event-based** hardware behavior by closures
    - A closure is a record storing a function together with an environment that allows the function to access other captured variables.
    - Closures can be defined in the place as async operation initiation, with sequential code.
  - C++ and Lua userland
    - Lua – Dynamic-loading language
    - Programs can be modified at runtime to adapt to devices appearing in the ensembles.

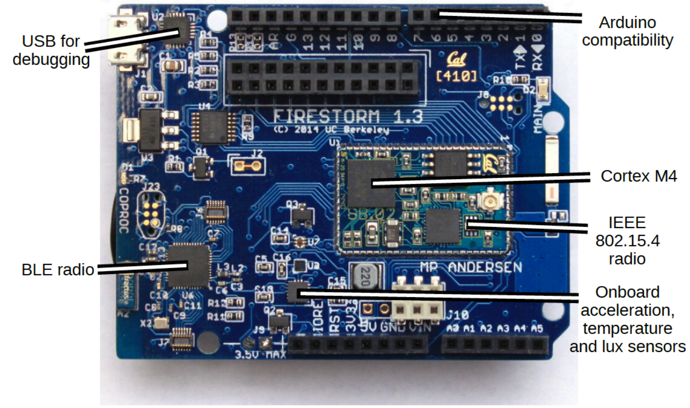
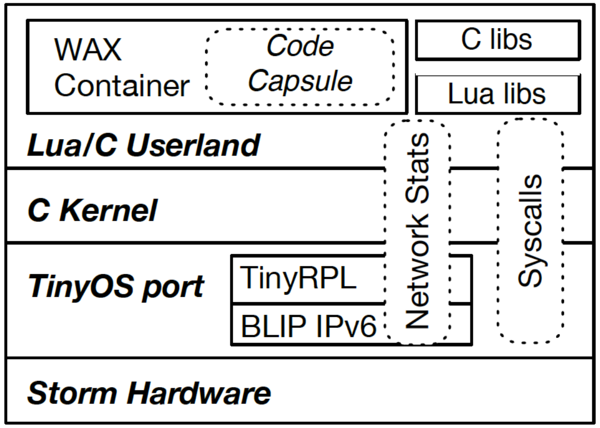


Figure 3: The Firestorm platform



The full "Wireless Active Network" (WAX) hardware/software stack for network experimentation

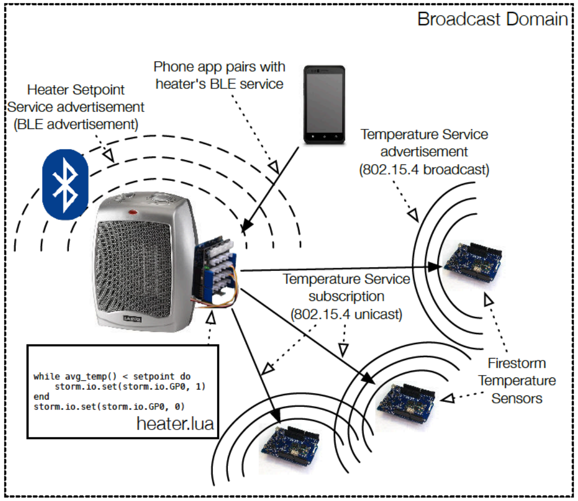
Andersen, Michael P., Gabe Fierro, and David E. Culler. "Enabling synergy in iot: Platform to service and beyond." *Journal of Network and Computer Applications* 81 (2017): 96-110.

Enabling Synergy in IoT: Platform to Service and Beyond

- **Person-where**
  - Enables devices in broadcasting domain to discover nearby self-describing services.
  - Synergistic service discovery (SVCD) – Async API to service
- **Local-where**
  - Deals with devices in an area larger than broadcasting domain.
  - Continuous query-based syndication (CQBS) – Registry and Coordinator
- **Wide-where**
  - Manages large-scale ensembles owned by multiple parties
  - Building Operating System Services Wide Area Verified Exchange (BOSSWAVE) – Identity, Namespace, Permissions, etc.

SVCD Function
`init()`
`add_service(svc_id)`
`add_attr(svc_id, attr_id, write_fn)`
`notify(svc_id, attr_id, value)`
`subscribe(targetip, svc_id, attr_id, on_notify) -> subscription_id`
`unsubscribe(subscription_id)`
`advert_received(payload, src_ip)`

API for cross-network service description and utilization



Composing locally-discovered services into an ensemble

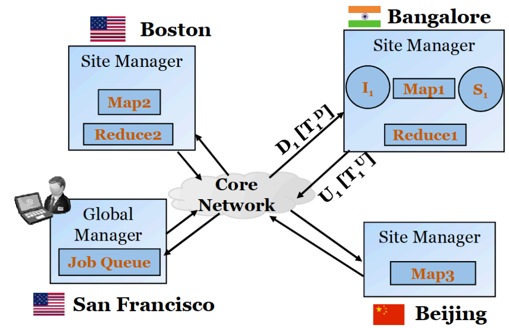
Andersen, Michael P., Gabe Fierro, and David E. Culler. "Enabling synergy in iot: Platform to service and beyond." *Journal of Network and Computer Applications* 81 (2017): 96-110.

Challenges vs Opportunity

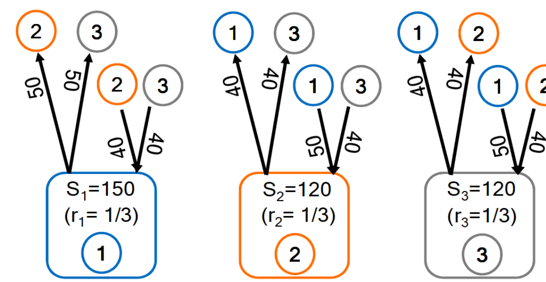
Challenges	Research Opportunity
Scalability & Distribution	Scalable Event-driven System
Ubiquitous Connectivity	Heterogeneous Device Network
Wide Area Analytics	**Geo-distributed Data & Control Platform**

Low Latency Geo-distributed Data Analytics

- **Problem**
  - Dataset queries across geo-distributed datacenters and edge clusters lead to high response time
  - Aggregating all data to one datacenter inflates timeliness of analytics
  - Unconstrained WAN link usage cost
- **Contribution**
  - Iridium – A system for low latency geo-distributed data analytics
    - **Intuition:** The duration of intermediate communications depends on the site with **slowest data transfer.** (Bottleneck)
    - **Key idea:** Iridium is to balance the transfer times among the WAN links to avoid outliers.



Geo-distributed map-reduce query

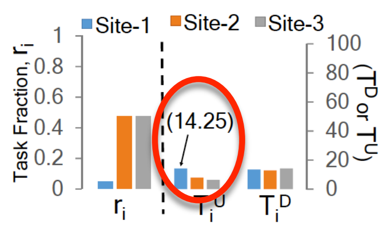


1/3-1/3-1/3 Equally spread scenario 5%-45%-45%

	Site-1	Site-2	Site-3
Input Data (MB), *I*	300	240	240
Intermediate Data (MB), *S*	150	120	120
Uplink (MB/s), *U*	10	10	10
Downlink (MB/s), *D*	1	10	10

(a) Setup of three sites.

(c) Equal Spread

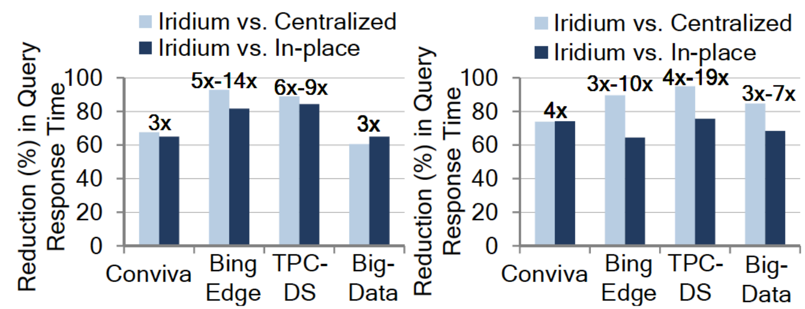


(d) Better Alternative

Pu, Qifan, et al. "Low latency geo-distributed data analytics." *ACM SIGCOMM Computer Communication Review*. Vol. 45. No. 4. ACM, 2015.

Low Latency Geo-distributed Data Analytics

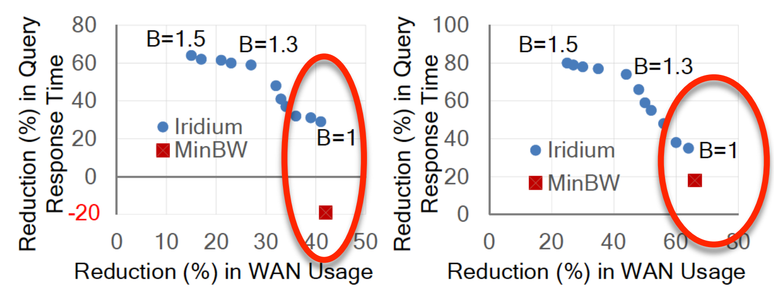
- **Iridium design**
  - Online heuristic - Iteratively identify bottlenecks and move data out to reduce intermediate data transfer time
  - Places the tasks to reduce the longest finish time depending on uplink and downlink bandwidth
  - Budgets the WAN link usage



(a) Inter-Region (b) 30 sites

EC2 Results across eight worldwide regions and 30 sites. Iridium is 3x-19x faster compared to two baselines.

- **Implementation & Evaluation**
  - Spark and EC2 deployment
    - Iridium is built upon Spark
    - Parse SQL queries and run streaming jobs
  - Iridium speeds up popular workloads benchmarks by 3x to 19x across 8 EC2 regions
  - Iridium saves WAN bandwidth usage by 15% to 64%. Even with WAN-usage optimal policy, the query speeds up 2x more.



(a) In-place baseline (b) Centralized baseline

WAN bandwidth Usage knob, B. MinBW is the scheme that optimizes for WAN usage. It **slows down** queries against the in-place baseline.

Pu, Qifan, et al. "Low latency geo-distributed data analytics." *ACM SIGCOMM Computer Communication Review*. Vol. 45. No. 4. ACM, 2015.

Large-scale Mobile Sensing Enabled Internet-of-Things Testbed for smart City Services

- **Problem**
  - Widespread deployment of spatially distributed IoT devices into physical realm of cities
  - Metrics of large-scale IoT systems
  - Socioeconomic acceptance
  - Quantification of service usability
  - Performance with end user in the loop
- **Contribution**
  - Presents the deployment of a large-scale IoT smart city test bed in Santander, Spain
  - Introduces three mobile sensing network strategies for distributing data
  - Sketches smart city services supported by data from mobile IoT

An aggregated location view of 30,000 observations taken by mobile IoT devices.
Orange(>100), Blue(15-100), Pink(<15)

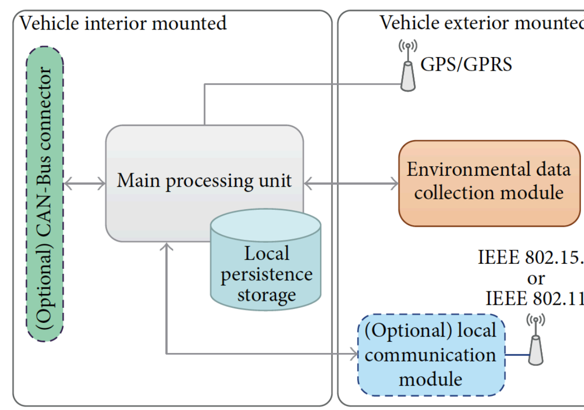
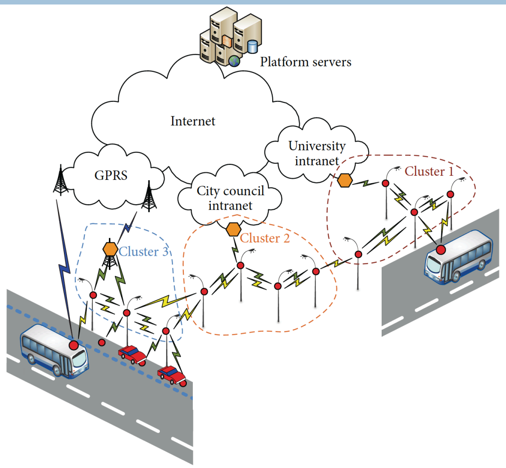


FIGURE 4: Mobile IoT device schema.

Lanza, Jorge, et al. "Large-Scale mobile sensing enabled internet-of-things testbed for smart city services." *International Journal of Distributed Sensor Networks* 11.8 (2015): 785061.

Large-scale Mobile Sensing Enabled Internet-of-Things Testbed for smart City Services

- SmartSantander IoT Platform
  - IoT device tier
  - IoT gateway tier
  - Server tier
- Mobile sensing data distributing strategies
  - WAN-based reporting
  - V2I-based opportunistic reporting (selected)
  - Batched reporting
- Smart City Applications
  - Environmental Monitoring
  - Traffic Conditions Assessment



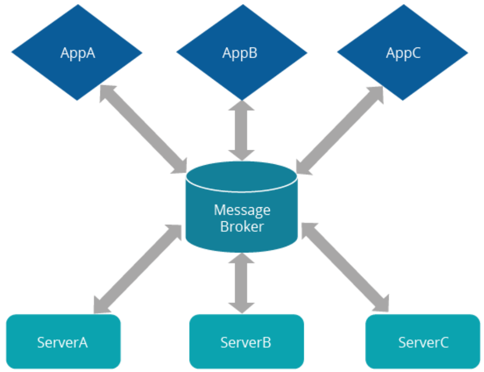
Test bed physical network diagram
Red dot – IoT node, Orange dot - Gateway

Traffic Condition Assessment application – Smarter Travel

Lanza, Jorge, et al. "Large-Scale mobile sensing enabled internet-of-things testbed for smart city services." *International Journal of Distributed Sensor Networks* 11.8 (2015): 785061.

Research Opportunities

- The fusion of cloud computing and Internet of Things
  - The next generation of scalable event-driven architecture and toolchain adaptable to IoT system
  - The network of heterogeneous IoT devices addressing ubiquitous connectivity
  - The geo-distributed data and control platform for wide area big data analytics




Q&A