Optimal Resource Allocation for Analytics and Multimedia Applications in Cloud-to-Things Continuum Platforms

洪華駿

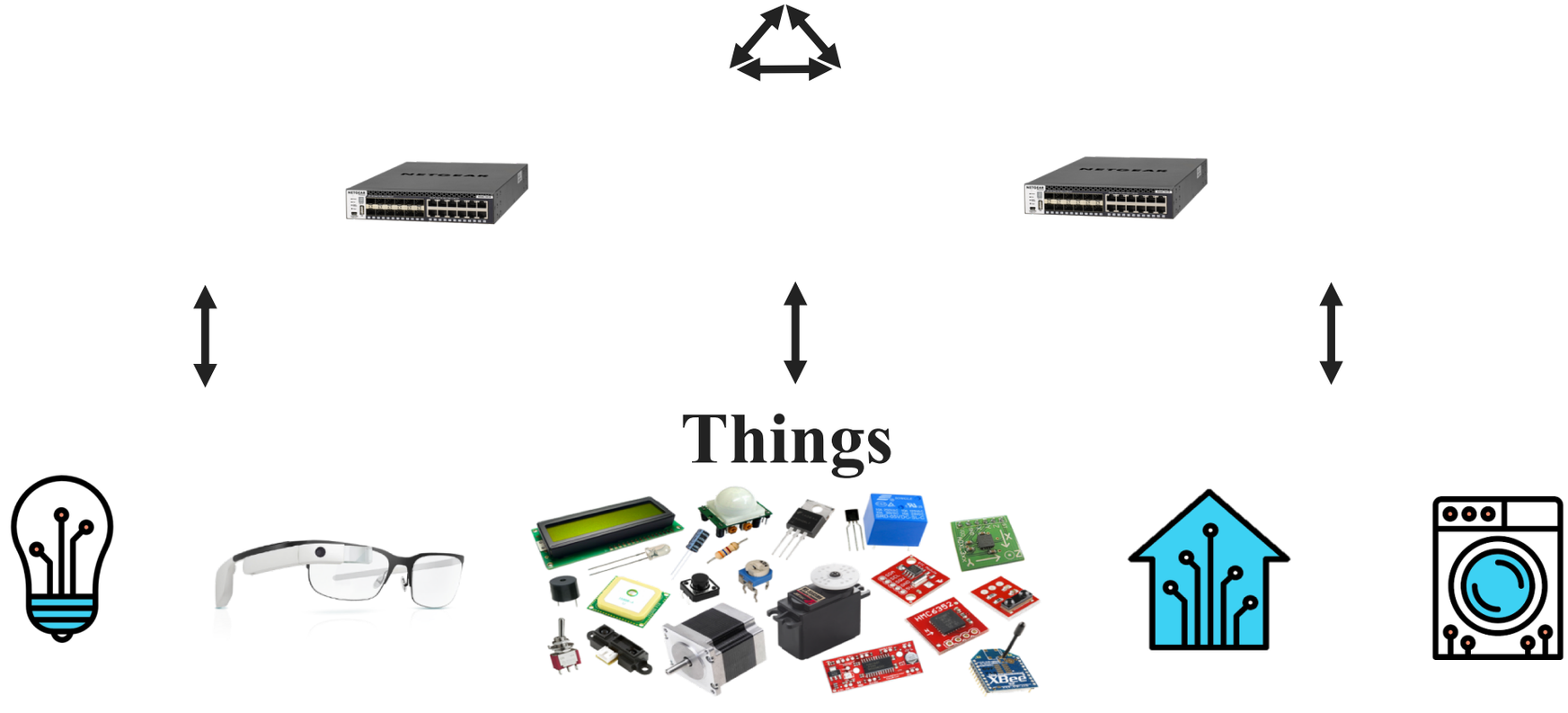
Department of Computer Science, National Tsing Hua University, Taiwan

Agenda

I. Introduction
II. Application Deployment Problem
III. Delay-Sensitive Application Optimization Problem
IV. Delay-Insensitive Application Optimization Problem
V. Conclusion & Future Work

Cloud

Why don’t we leverage the resources in between cloud and things?



Cloud-to-Things Continuum Platform

Provider

Intruder
Fugitive
Detector
Tracking

Xeon
E-2176

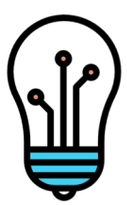
Resource Management

Cat7 Gigi

Fog Devices

PM 2.5

1TB
SSD

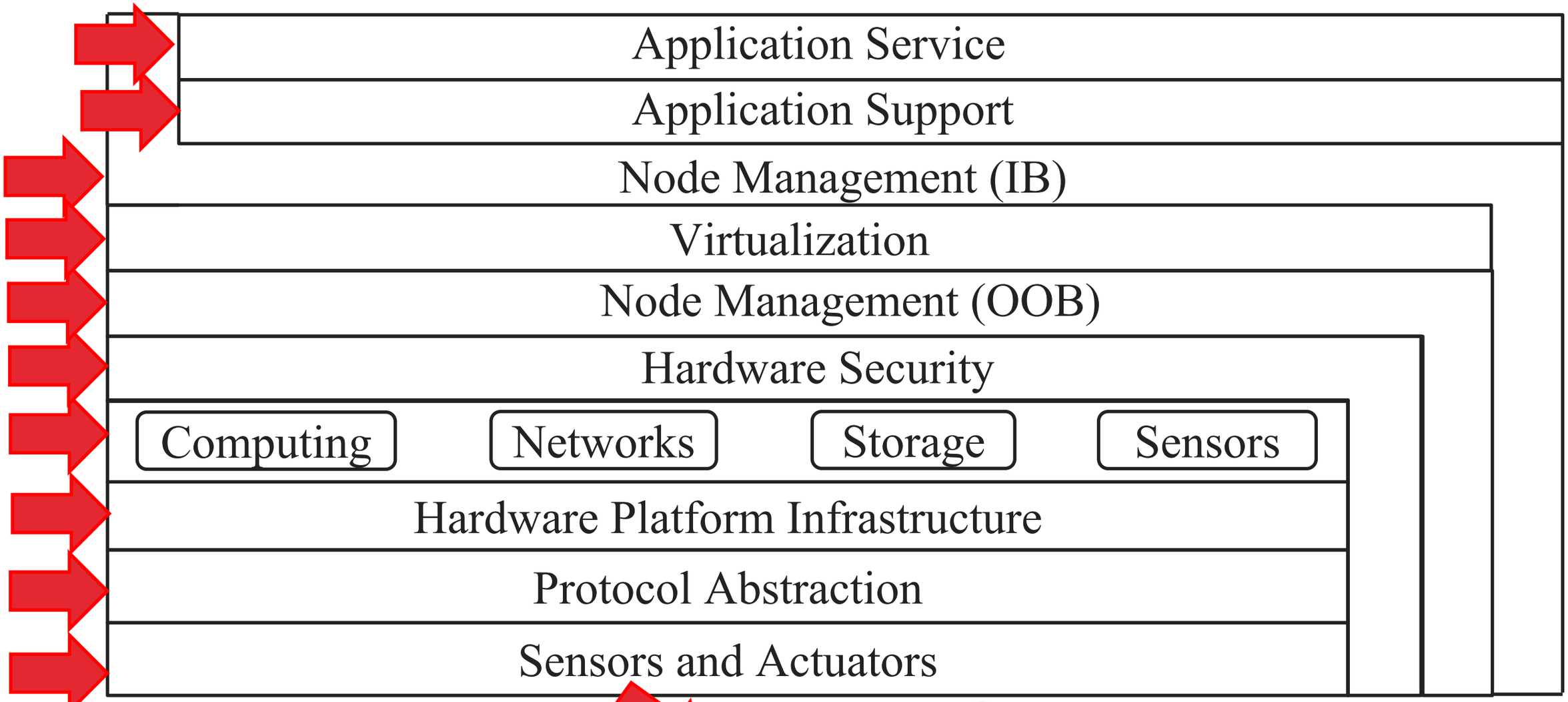


User 2

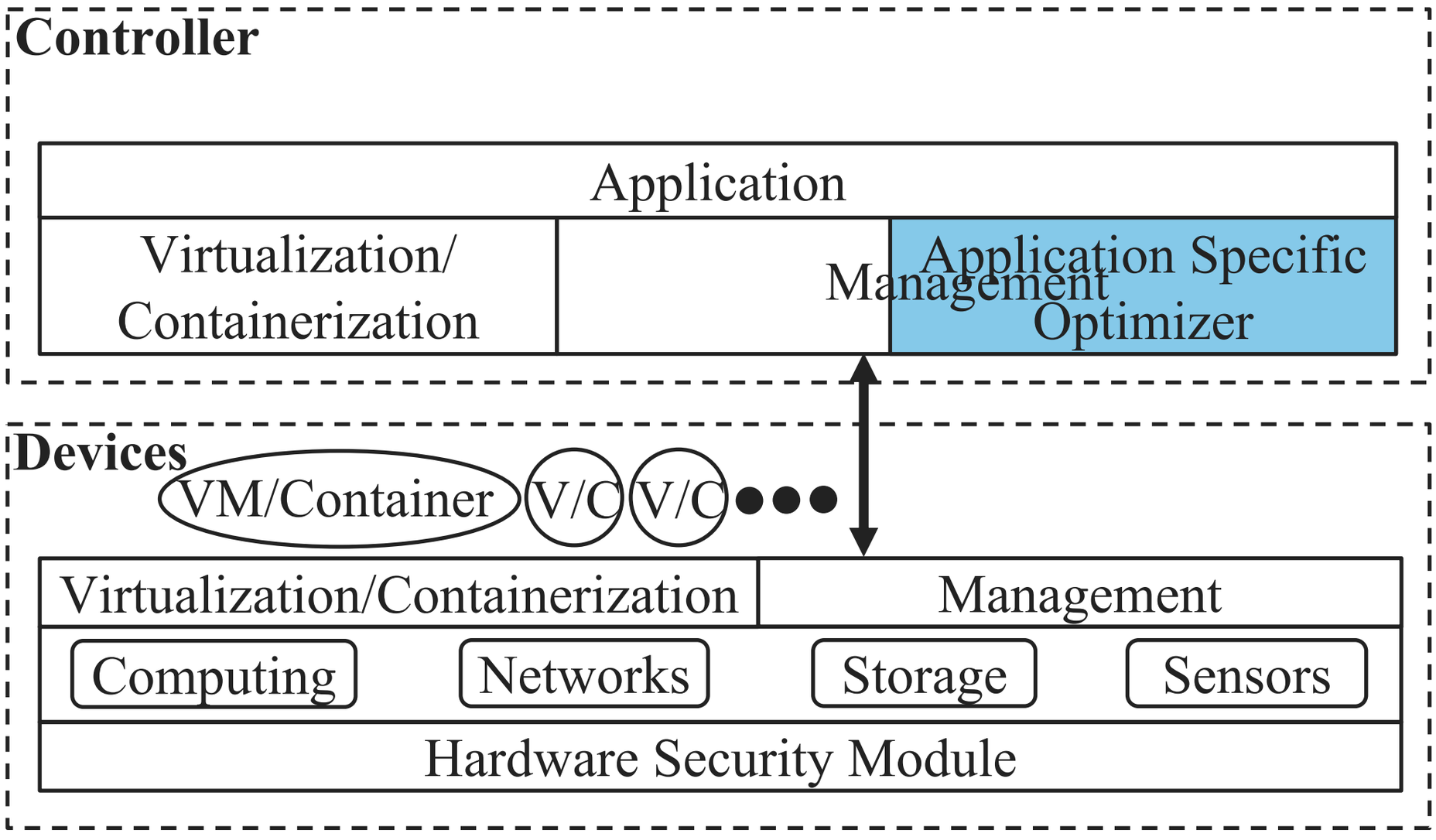
User 1

[1] https://www.openfogconsortium.org/

Cloud-to-Things Continuum Framework [1]



Intelligent Framework



Three Research Problems

Goal: Serving as many users as possible

Provider

Users

Global Optimizer

Application Deployment Problem

1

Application Specific Optimizer

Delay-Sensitive Application Optimization Problem

2

Delay-Insensitive Application Optimization Problem

3

Various Applications

Ongoing Applications

Application: Remote Rendering

Goal: Reduce latency

Application: Content Delivery

Goal: Maximize delivered information

Contributions

Propose an intelligent cloud-to-things continuum framework

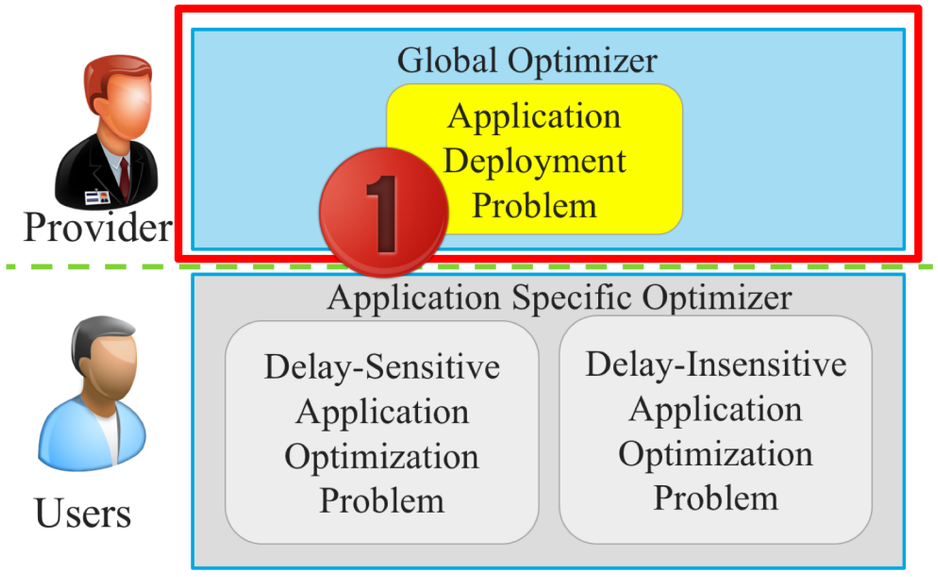
1. Solve application deployment problem
   - Deploy applications while considering decompositions
   - 1/U approximation factor
2. Solve delay-sensitive application optimization problem
   - Adapt bitrates for game streaming applications
   - Optimal solution solved in polynomial time
3. Solve delay-insensitive application optimization problem
   - Provide content delivery plans while considering user behaviors and content representations under challenged networks
   - An efficient algorithm outperforms state-of-the art algorithms

Agenda

I. Introduction
II. Application Deployment Problem 1
III. Delay-Sensitive Application Optimization Problem
IV. Delay-Insensitive Application Optimization Problem
V. Conclusion & Future Work

Agenda: Application Deployment Problem

- Problem Statement
- Proposed Algorithm: Ideal Case
- Proposed Algorithm: General Case
- Evaluation & Summary



Various Deployment Decisions

Provider

Fugitive Tracking (operator 1)

Fugitive Tracking (operator 2)

Fugitive Tracking (operator 3)

Xeon E-2176

Cat7 Gigi

Resource Management

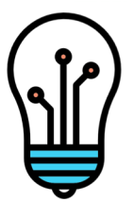
Fog Devices

PM 2.5

1TB SSD

Suffer From High Latency

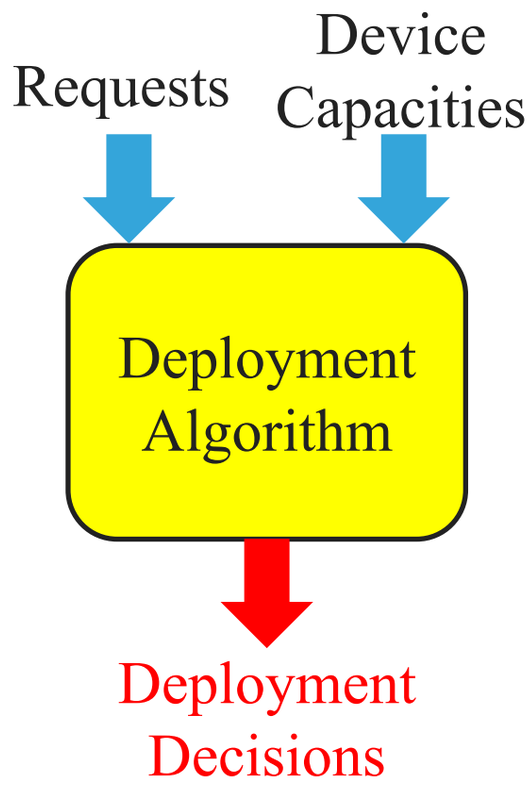
Suffer From Limited Computation Resource



User

Application Deployment Problem

Problem: Given a set of requests $\boldsymbol{Q}$ and a set of fog devices $\boldsymbol{V}$. Each request has three requirements: (i) a splittable application, (ii) a target QoS, and (iii) a specified location. Devices located at various places have $\boldsymbol{U}$ kinds of resources. Our problem is to determine which request $\boldsymbol{q} \in \boldsymbol{Q}$ should be served on which device $\boldsymbol{v} \in \boldsymbol{V}$ without overload the resources. Our goal is to maximize number of served requests.

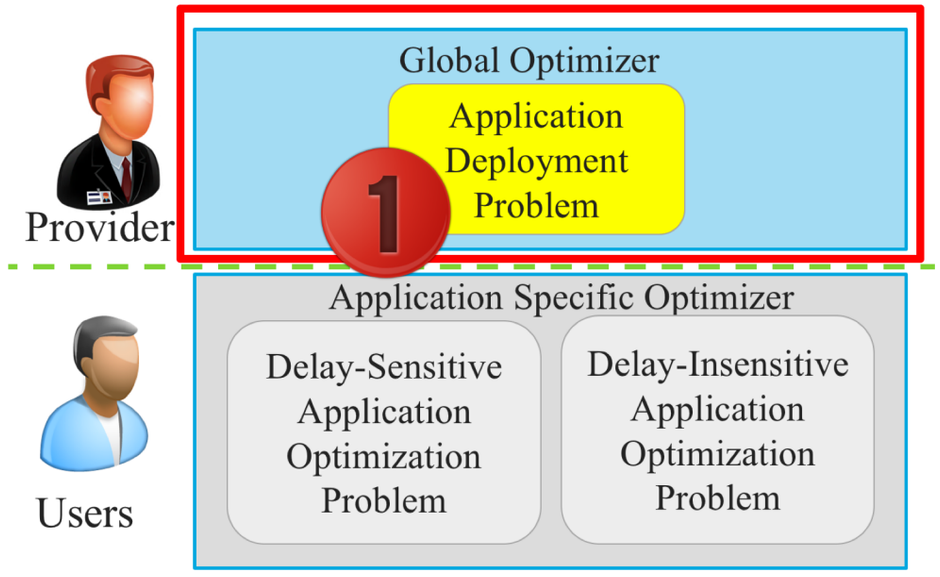


Challenges:

- Diverse users (requests)→ diverse resource requirements
- Heterogeneous fog devices → application decomposition
- Large problem size → polynomial time algorithm

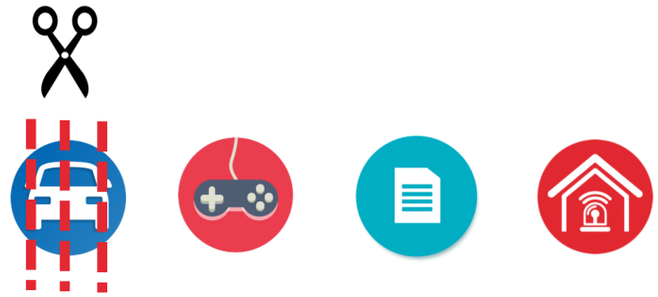
Agenda: Application Deployment Problem

- Problem Statement
- Proposed Algorithm: Ideal Case
- Proposed Algorithm: General Case
- Evaluation & Summary



Application Deployment Problem: Ideal Case

- Splittable applications
  - Arbitrary size of operators
- Transfer link constraints to node constraints
- Grid map
  - Select one of grids
- Reduced device heterogeneity
  - Proportional resource levels



X4

X2

1 core 2 GHz
1 GB RAM
…

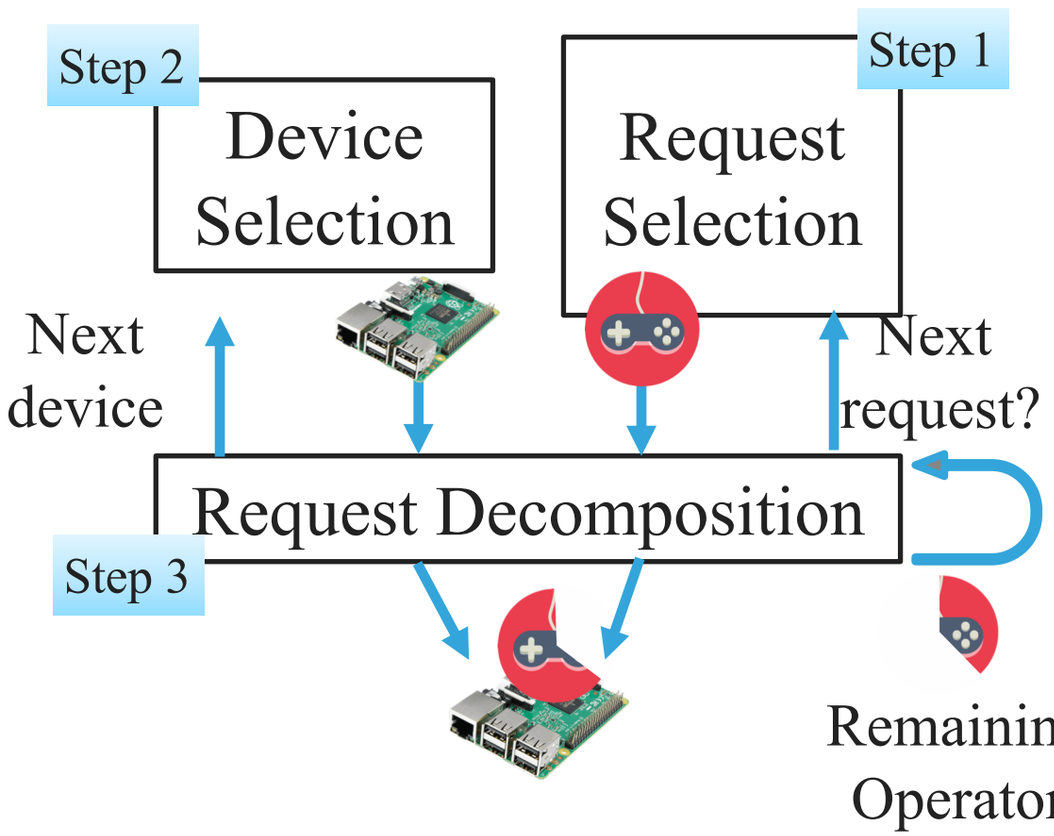
4 cores 2 GHz
4 GB RAM
…

8 cores 2 GHz
8 GB RAM
…

✓ Variant of knapsack problem
✓ NP hard

APproXimation Algorithm (APX)

1. Step 1: Request selection
   - Least total required resource first $\min_{q \in Q} \sum_{u \in U} F(q, u, \cdots)$
   - q: request
   - u: resource type
   - F: resource consumption model
2. Step 2: Device selection
   - Round robin
3. Step 3: Request decomposition
   - As large operator as possible



- O(|Q| log |Q|+|Q||V||U|)
- 1/U Approximation Factor

Q: Request set
V: Device set
U: Resource type set

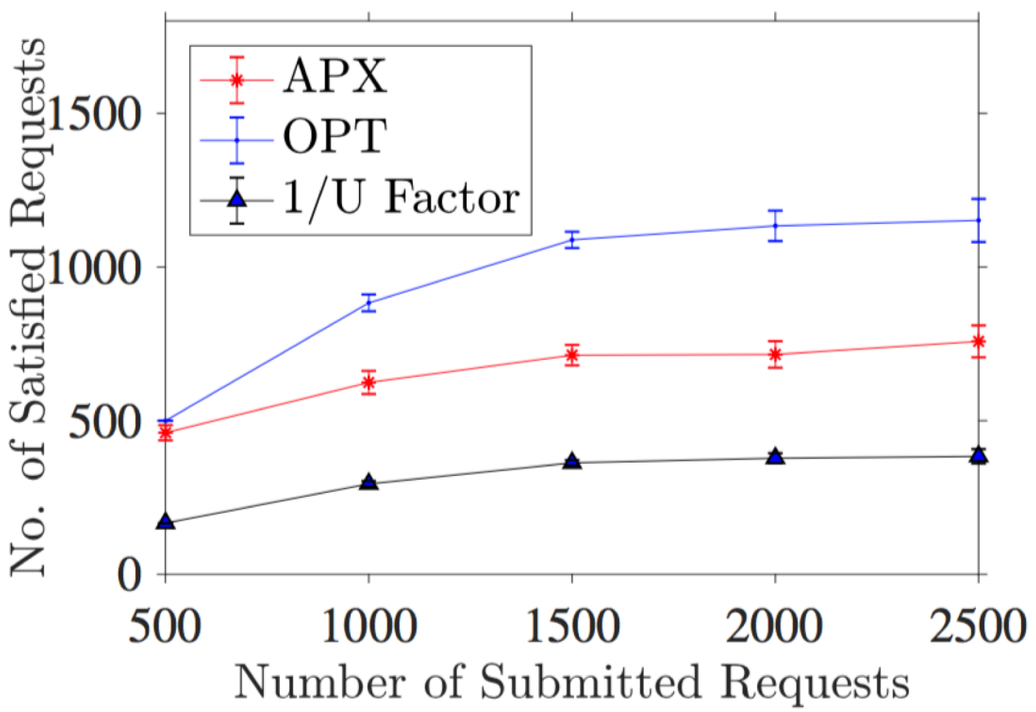
Proof: $\frac{1}{|U|}$ Approximation Factor

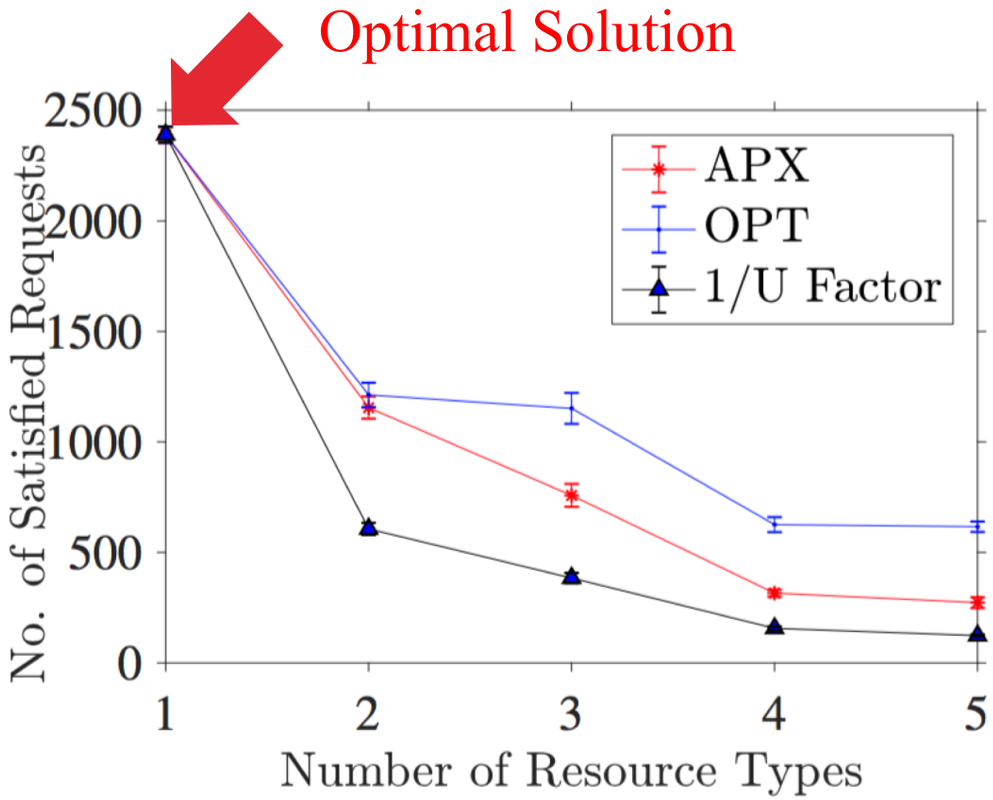
- Step1: Design an upper bound solution (OPT')
  - Intuitions are similar to APX
- Step2: Prove results of OPT' >= optimal solution (OPT)
- Step3: Derive approximation factor (APX/OPT')

The algorithm results in optimal solution while |U| = 1

Validating the Approximation Factor

- OPT: formulate as ILP and optimally solve it using CPLEX
- Results of APX are always above theoretical 1/U bound
- APX == OPT while U=1

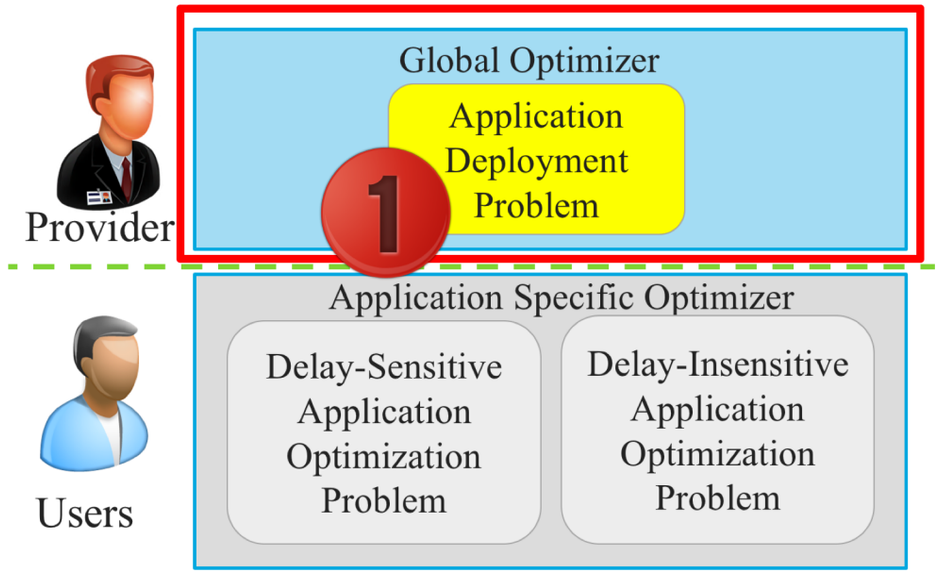




Limitations: Operators, Links, Location, and Devices
→ How to make it more generalized?

Agenda: Application Deployment Problem

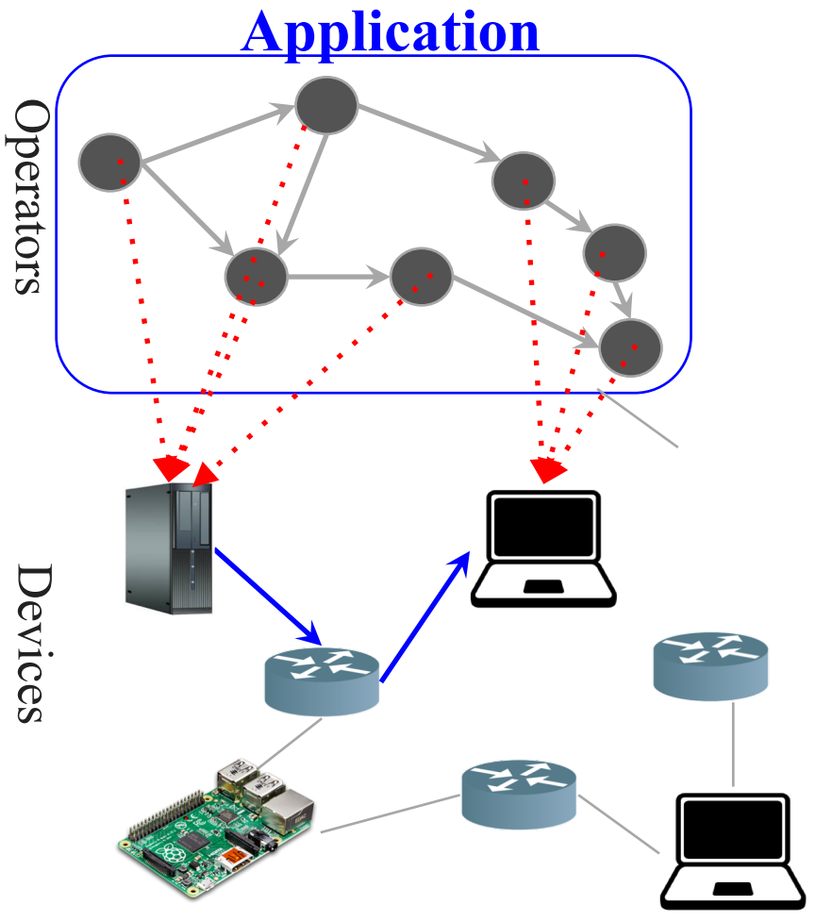
- Problem Statement
- Proposed Algorithm: Ideal Case
- Proposed Algorithm: General Case
- Evaluation & Summary



Application Deployment Problem: General Case

- Goal: Serve as many requests (users) as possible

	Ideal	Generalized
Operator	Any Size	Predefined
Resource Constraint	Node	Node + Link
Location	Grids	Any Location
Device Heterogeneity	Proportional	Any Device



Problem Formulation

$$\max \sum_{q \in \mathbf{Q}} p_q \qquad \text{Decision variable}$$

$$st: z_q = \sum_{i \in \hat{\mathbf{V}}_{\mathscr{A}_\mathbf{q}}} \sum_{k \in \mathbf{V}_{q,i}} x_{q,i,k} / |\hat{\mathbf{V}}_{\mathscr{A}_\mathbf{q}}| \ \forall q \in \mathbf{Q}; \quad \text{(1a)}$$

Objective

Throughput maximization

$$z_q - 1 < p_q \leq z_q \ \forall q \in \mathbf{Q}; \quad \text{(1b)}$$

$$\sum_{k \in \mathbf{V}_{q,i}} x_{q,i,k} \leq 1 \ \forall q \in \mathbf{Q}, i \in \hat{\mathbf{V}}_{\mathscr{A}_\mathbf{q}}; \quad \text{(1c)}$$

Constraints

Node

$$\sum_{q \in \mathbf{Q}} \sum_{i \in \hat{\mathbf{V}}_{\mathscr{A}_q}} \dot{F}(\cdot) x_{q,i,k} \leq R_{k,u} \quad \text{(1d)}$$

$$\forall k \in \mathbf{V}_i, u \in \mathbf{U};$$

$$\sum_{q \in \mathbf{Q}} \sum_{i \in \hat{\mathbf{V}}_{\mathscr{A}_q}} \sum_{i' \in \hat{\mathbf{V}}_{\mathscr{A}_q}} \sum_{k \in \mathbf{V}_{q,i}} \sum_{k' \in \mathbf{V}_{q,i'}} y_{q,(i,i'),(k,k')} \quad \text{(1e)}$$

$$J_{q,i,i'} T_{(k,k'),l} \bar{F}(\cdot) \leq B_l \ \forall l \in \mathbf{E}; \quad \text{(1f)}$$

Link

$$x_{q,i,k} = \sum_{k' \in \mathbf{V}_{q,i'}} y_{q,(i,i'),(k,k')} \quad \text{(1g)}$$

$$\forall q \in \mathbf{Q}, i \in \hat{\mathbf{V}}_{\mathscr{A}_q}, i' \in \hat{\mathbf{V}}_{\mathscr{A}_q}, k \in \mathbf{V}_{q,i};$$

$$x_{q,i',k'} = \sum_{k \in \mathbf{V}_{q,i}} y_{q,(i,i'),(k,k')} \quad \text{(1h)}$$

$$\forall q \in \mathbf{Q}, i \in \hat{\mathbf{V}}_{\mathscr{A}_q}, i' \in \hat{\mathbf{V}}_{\mathscr{A}_q} k' \in \mathbf{V}_{q,i};$$

$$p_q, x_{q,i,k} \in \{0,1\} \ \forall q \in \mathbf{Q}, i \in \hat{\mathbf{V}}_{\mathscr{A}_\mathbf{q}}, k \in \mathbf{V}_{q,i}. \quad \text{(1i)}$$

SSE Algorithm

Node — Request Selection

Reduce node resource consumption

- Scarcest resource first

Link — Source/Destination device selection

Reduce latency

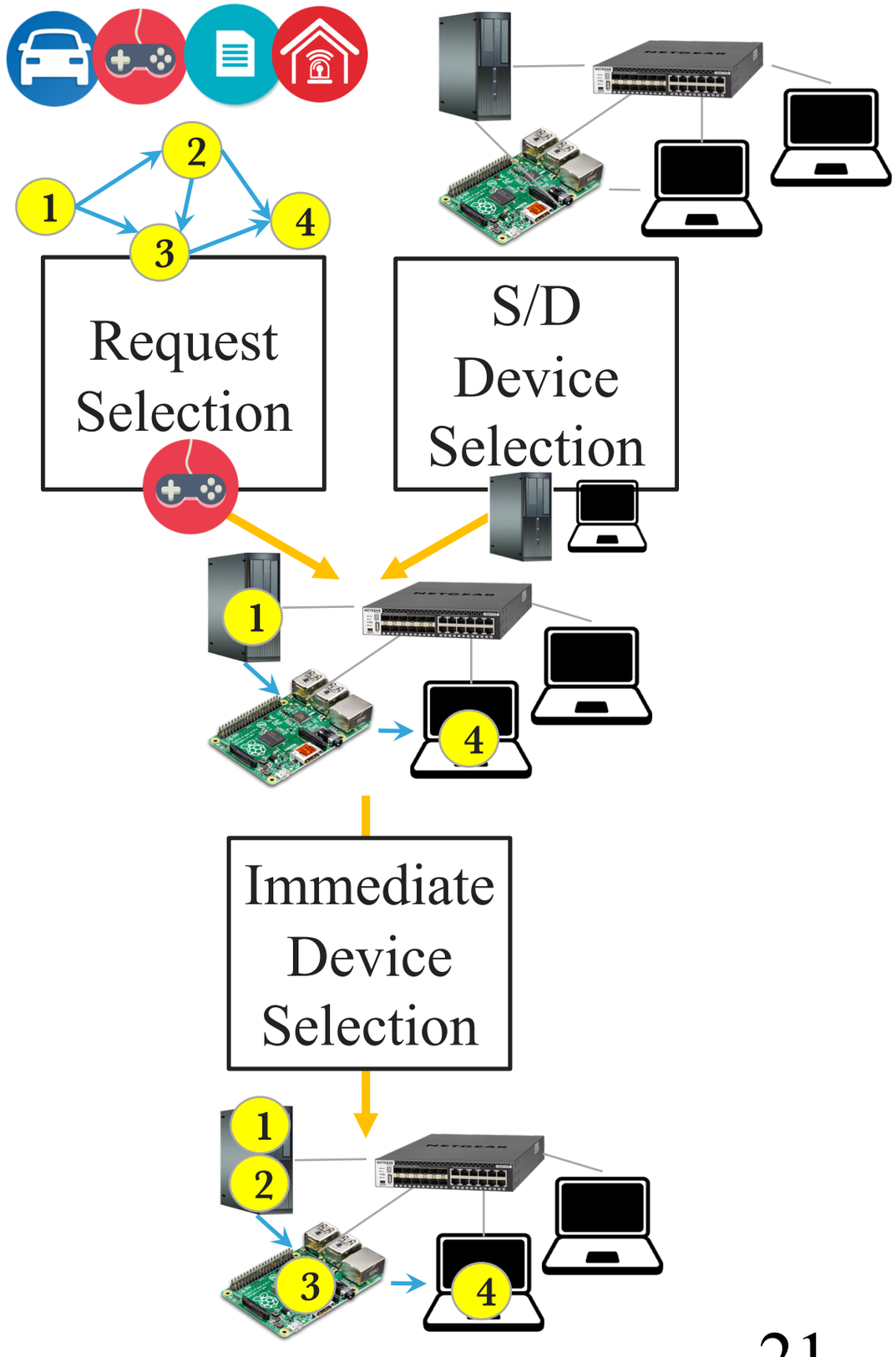
- Shortest path first (hops and latency)

Link — Immediate device selection

- Early feature extraction

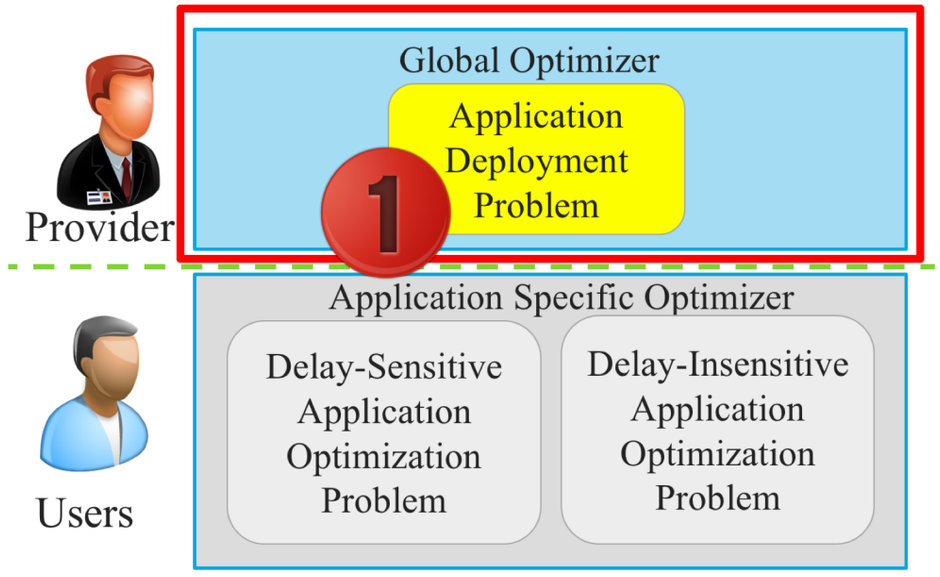
Reduce network resource consumption

Device Selection



Agenda: Application Deployment Problem

- Problem Statement
- Proposed Algorithm: Ideal Case
- Proposed Algorithm: General Case
- Evaluation & Summary



Evaluation Setup

- Requests
  - Poisson arrival / departure rates: 1 min / 10 mins
- Network topology (BRITE [1])
  - Number of fog devices: [10, 25, 50, 75, 100]
  - Location of fog devices and latencies of their edges: based on BRITE
- Resource capacities of fog devices
  - CPU: [100% ~ 800%] → random
  - RAM: [1 GB ~ 16 GB] → random
- Bandwidth of links:
  - [45 kbps (LoRa), 8 Mbps (WiFi), 25 Mbps (4G)] → Random

[1] BRITE, https://www.cs.bu.edu/brite/user_manual/

Baseline Algorithms

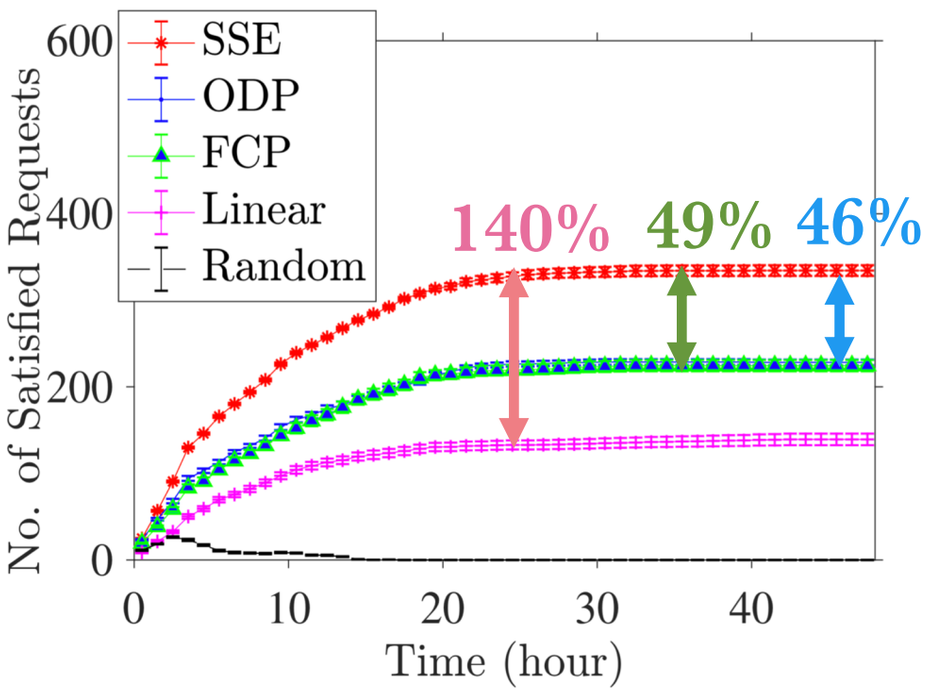
- Designed for fog devices
- Splittable application
- Consider both node and link capacities

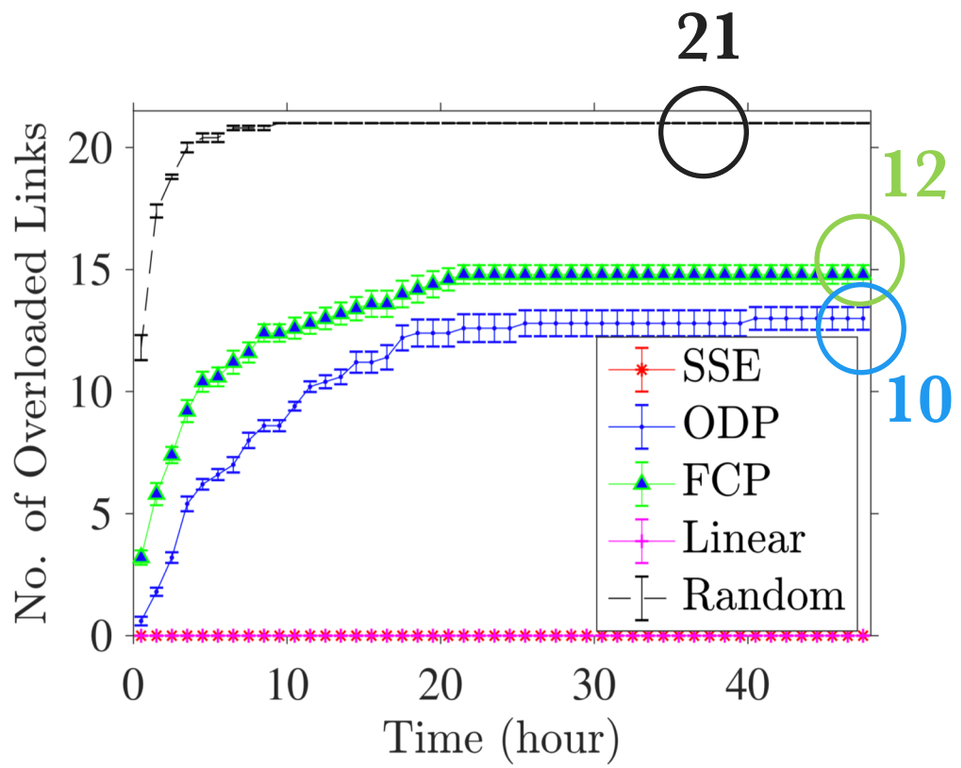
- Optimal Data Stream Processing Placement (ODP) algorithm [1]
  - CPLEX
- Fog and Cloud Placement (FCP) algorithm [2]
  - Heuristic
- Linear algorithm
  - Greedily deploys operators to neighbor fog devices
  - Considers all the constraints
- Random algorithm
  - Mimics a platform without centralized server
  - Does not consider any constraint

[1] V. Cardellini, V. Grassi, F. Presti, and M. Nardelli. Optimal operator placement for distributed stream processing applications. In Proc. of ACM International Conference on Distributed and Event-based Systems (DEBS), Irvine, CA, June 2016.
[2] M. Taneja and A. Davy. 2017. Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In Proc. of IFIP/IEEE Symposium on Integrated Network and Service Management (IM). Lisbon, Portugal.

SSE Serves More Users

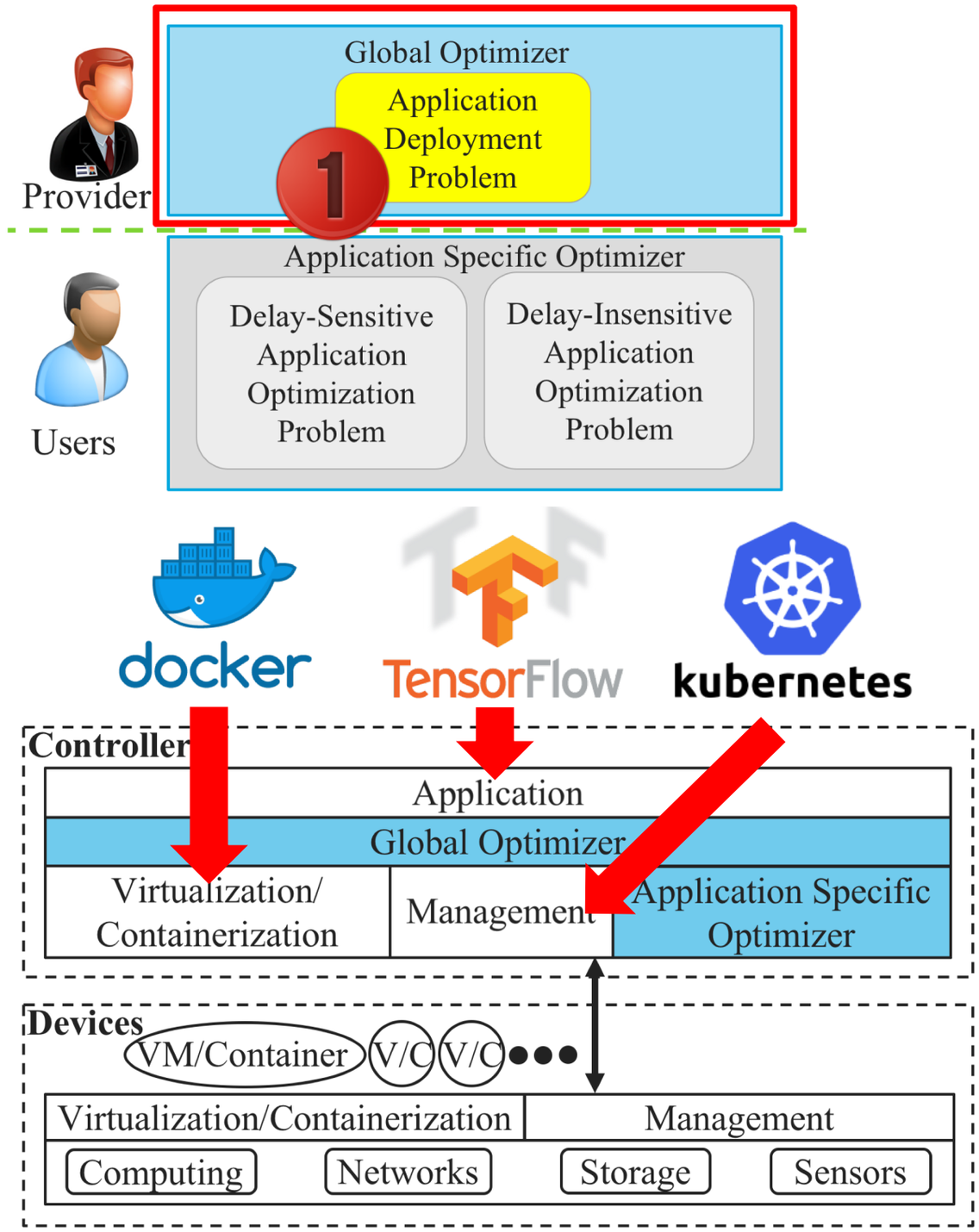
- SSE outperforms others in terms of supported users
  - 140%, 49%, and 46% compared to Linear, FCP, and ODP
  - Random satisfies zero requests after running 14 hours
- Random, FCP, and ODP are too aggressive
  - Overload links by up to 21, 12, and 10 links





Summary: Application Deployment

- Solved ideal and general application deployment problem
  - APX algorithm with $\frac{1}{|U|}$ approximation factor
  - Validated by numerical analysis
  - Satisfy at least 46% more requests compared to state-of-the-art algorithms
- Derived resource consumption models



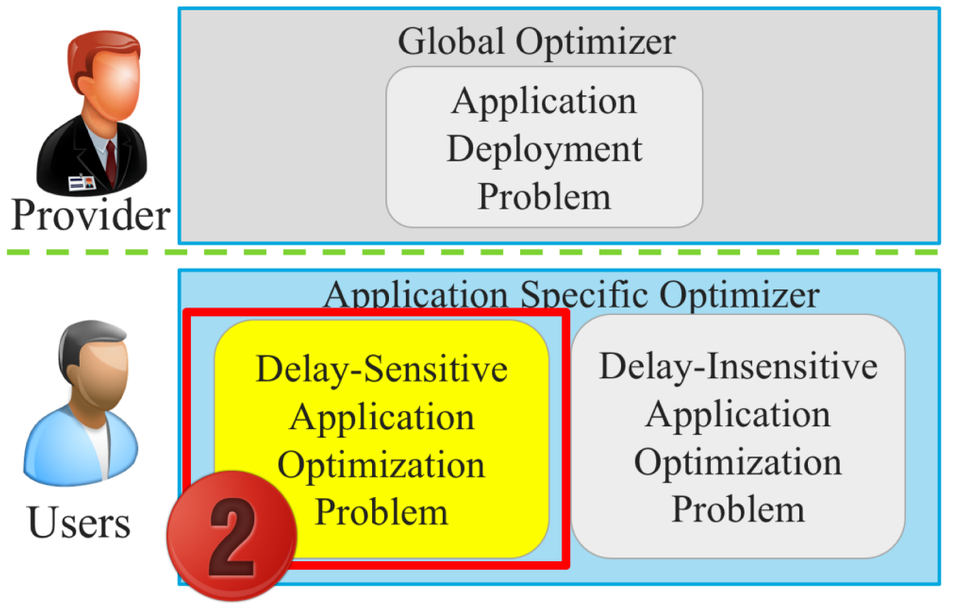
Our Testbed

Agenda

I. Introduction
II. Application Deployment Problem
III. Delay-Sensitive Application Optimization Problem 2
IV. Delay-Insensitive Application Optimization Problem
V. Conclusion & Future Work

Agenda: Delay-Sensitive Application Optimization Problem

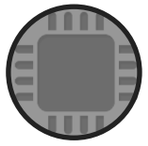
- Problem Statement
- Proposed Algorithm
- Evaluations & Summary



Game Streaming Applications

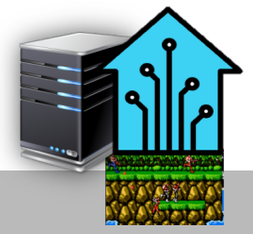
Provider

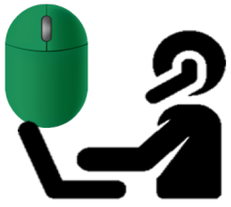
Xeon
E-2176



Cat7 Gigi

Fog Devices





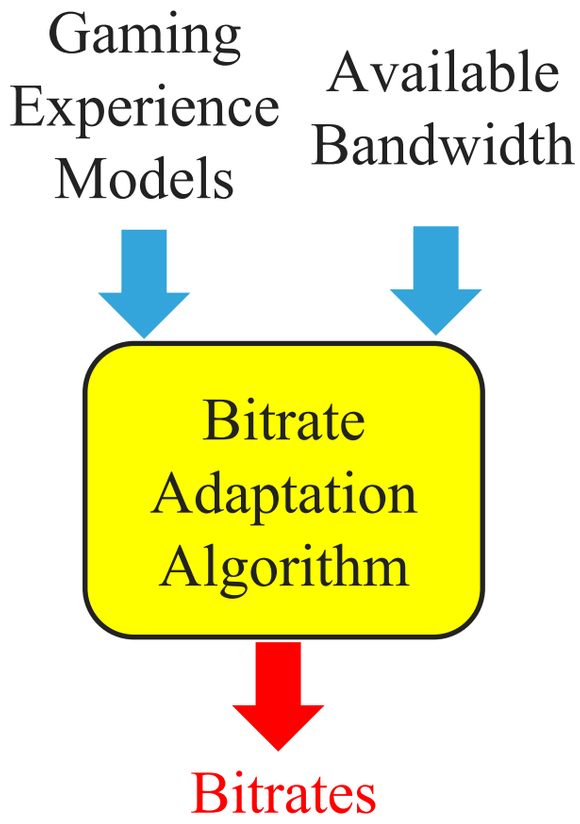
PM 2.5

1TB
SSD

Goal: Maximize gaming experience by adapting to network dynamics

Bitrate Adaptation Problem

Problem: Given a set of gamers $\boldsymbol{G}$ and a shared bottleneck link with available bandwidth $\boldsymbol{B}$. Our problem is to determine allocating bitrates, which do not overload $\boldsymbol{B}$ to $\boldsymbol{G}$. Our goal is to maximize gaming experience. The gaming experience model $m(g,b)$ is affected by game types g and allocated bitrates b.

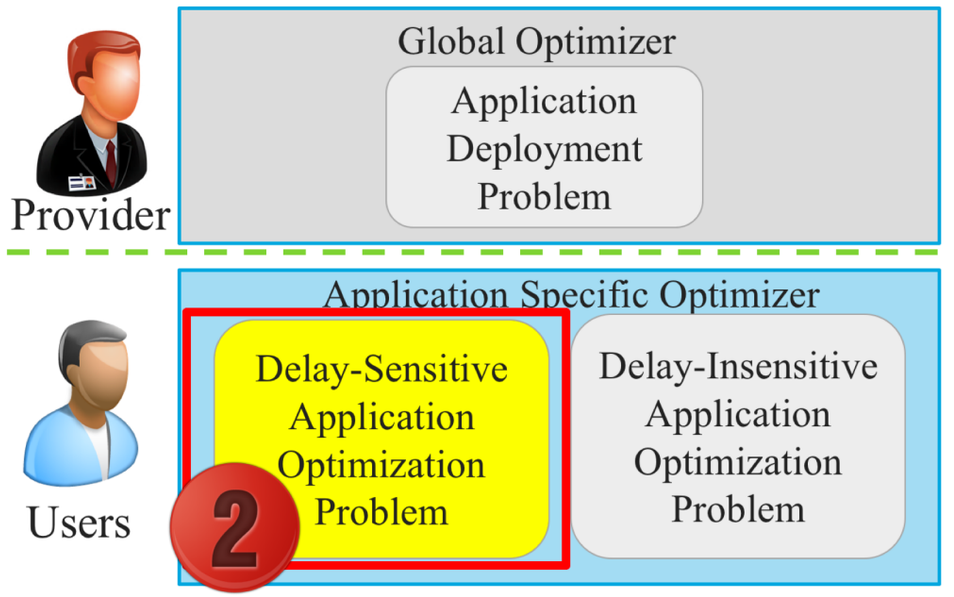


Challenges:

- Various game types and gamers → various gaming experience models
- Delay sensitive→ real-time algorithm

Agenda: Delay-Sensitive Application Optimization Problem

- Problem Statement
- Proposed Algorithm
- Evaluations & Summary



Optimal Bitrate Adaptation Algorithms

- Quality maximization: EFF_{avg}

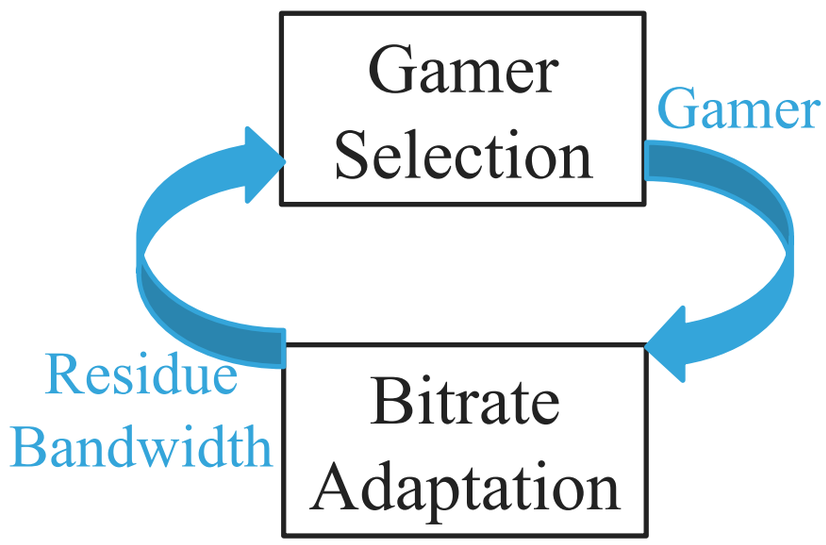
  Gamer selection: highest gaming experience improvement first

  Bitrate adaptation: add one unit ($w = 1kbps$)

- Quality fairness: EFF_{mm}

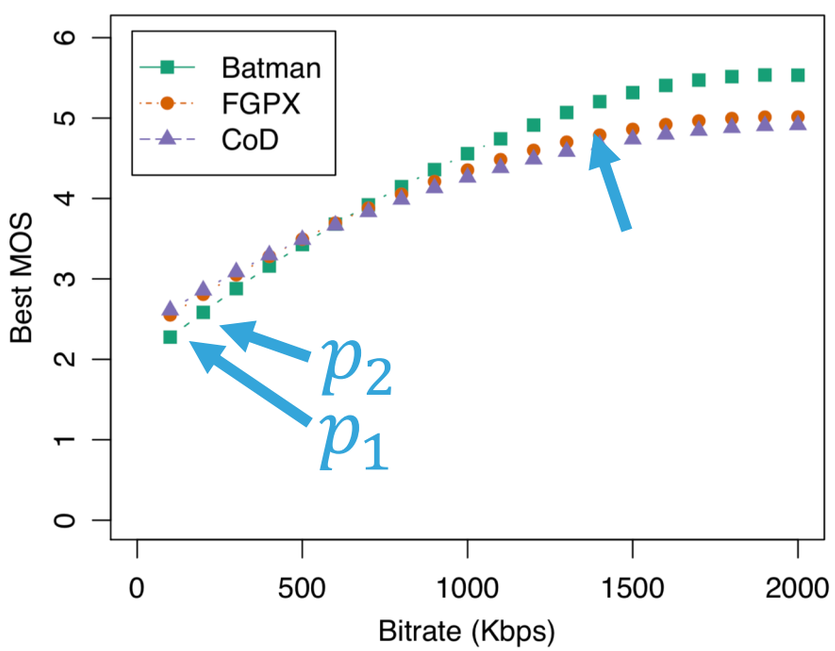
  Gamer selection: worst gaming experience first

  Bitrate adaptation: add one unit ($w = 1kbps$)



Optimality of the EFF_{avg} Algorithm

- Lemma: EFF_{avg} is optimal if gaming experience model $m^*(g,b)$ is monotonic decreasing
- Proof:
  - EFF_{avg} finds steepest slope at every iteration: p_1, p_2, p_3 ...
    → $p_i > p_{i+1}$ → No alternative allocation
    ✓ if it is feasible solution (monotonic decreasing) → Optimal
  - Monotonic decreasing: double deviation of $m^*(g,b)$
    → $m^*(g,b)'' < 0$



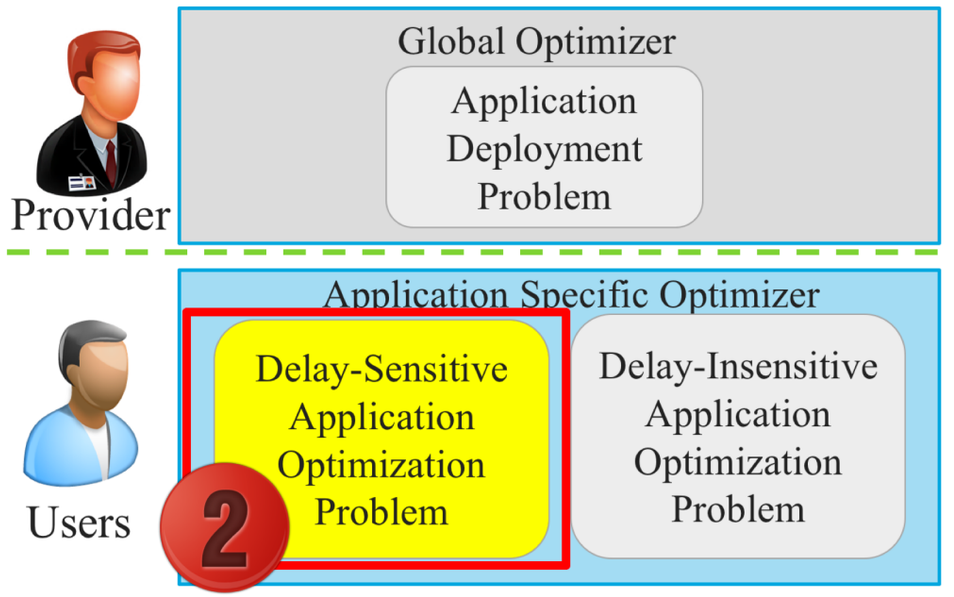
p_n: gaming experience improvement of adding one unit

Optimality of the EFF_{mm} Algorithm

- Lemma: EFF_{mm} is optimal if it allocates w to gamer u who has the worst gaming experience
- Proof:
  - EFF_{mm} allocates w (1kbps) to user u who has the worst gaming experience = l
    → MOS improvement = q
  - Consider alternative user u'
    → Improvement = 0 (the worst gaming experience is still l)
    → Need to allocate another w to u to achieve the same improvement
  - Hence, we cannot find an alternative leading to a better solution
    → EFF_{mm} is optimal

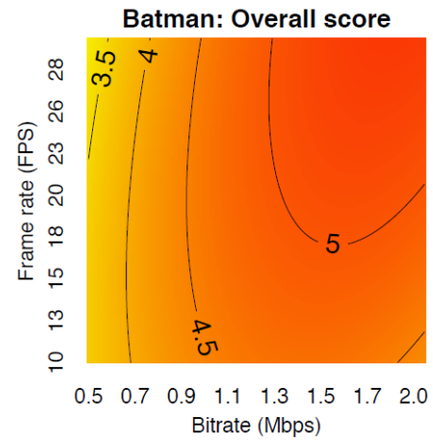
Agenda: Delay-Sensitive Application Optimization Problem

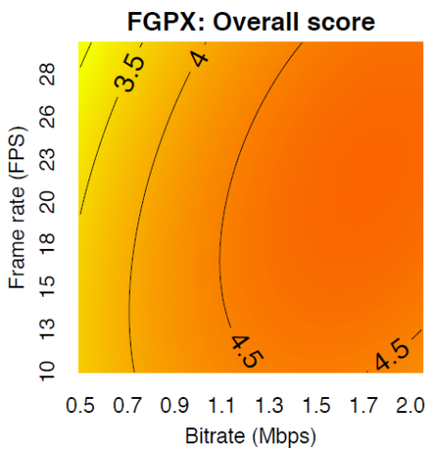
- Problem Statement
- Proposed Algorithm
- Evaluations & Summary

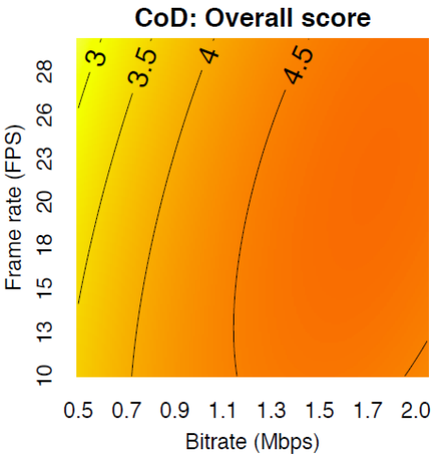


Evaluation Setup

- Gaming experience model: user study
- Available resource = 760 Mbps (PlanetLab)
- Optimal algorithms: CPLEX (an optimization solver)
  - OPT_{avg}
  - OPT_{mm}







Gaming Experience Model

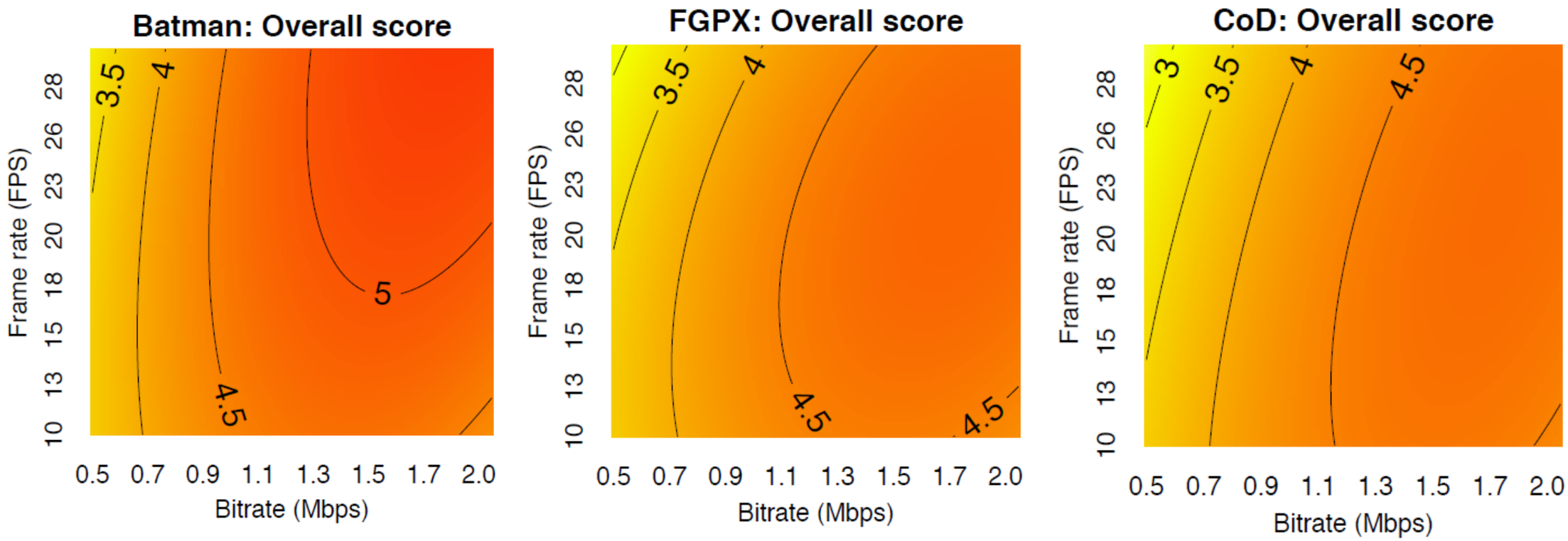
- Quadratic model

$$m(g, f, b) = \alpha_{g,1} f + \alpha_{g,2} b + \alpha_{g,3} f^2 + \alpha_{g,4} b^2 + \alpha_{g,5} fb + \alpha_{g,6}$$

Game Type — Frame Rate — Bitrate

- MOS Scores: 1 ~ 7
- R-square values > 0.98

How to simplify the problem? → focus on bitrate allocation



[Hong et al. TCSVT'15, Credits: collaborators @ Academia Sinica]

Optimal Frame Rate Selection

- Lemma: $m(g, f, b)$ calculates optimal MOS under certain bitrate if we have $f^*(g, b)$, which gives us optimal frame rate
- Quadratic gaming experience model

$$m(g, f, b) = \alpha_{g,1} f + \alpha_{g,2} b + \alpha_{g,3} f^2 + \alpha_{g,4} b^2 + \alpha_{g,5} fb + \alpha_{g,6}$$

- Partial derivation to f

→ Optimal frame rate: $f^*(g, b) = \dfrac{-(\alpha_{g,1} + \alpha_{g,5} b)}{2\alpha_{g,3}}$

- Optimal MOS:

$$\begin{aligned} m^*(g, b) = & \alpha_{g,1} f^*(g, b) + \alpha_{g,2} b + \alpha_{g,3} (f^*(g, b))^2 \\ & + \alpha_{g,4} b^2 + \alpha_{g,5} f^*(g, b) b + \alpha_{g,6}. \end{aligned}$$

$$m(g, f, b) \rightarrow m^*(g, b)$$

Our Algorithms are Optimal

- The resulting MOS scores are exactly the same
- Efficient algorithms run much faster than OPT algorithms

# of Gamers	$\mathbf{OPT}_{avg}$	$\mathbf{EFF}_{avg}$	# of Gamers	$\mathbf{OPT}_{mm}$	$\mathbf{EFF}_{mm}$
	Mean	Mean		Worst	Worst
100	5.16	5.16	1	5.53	5.53
200	5.15	5.15	2	5.01	5.01
400	5.15	5.15	4	4.91	4.91
800	4.49	4.49	8	4.91	4.91

Same

[MOS Scores]

# of Gamers	$\mathbf{OPT}_{avg}$		$\mathbf{EFF}_{avg}$		# of Gamers	$\mathbf{OPT}_{mm}$		$\mathbf{EFF}_{mm}$	
	Mean	Max	Mean	Max		Mean	Max	Mean	Max
100	0.02	0.03	0.090	0.097	1	0.02	0.01	0.009	0.009
200	0.15	0.16	0.102	0.103	2	0.36	0.37	0.015	0.016
400	0.95	0.96	0.153	0.157	4	4.47	4.48	0.022	0.023
800	5.01	5.02	0.237	0.248	8	125.8	127.1	0.029	0.029

21X faster

[Running Time]

4K+X faster

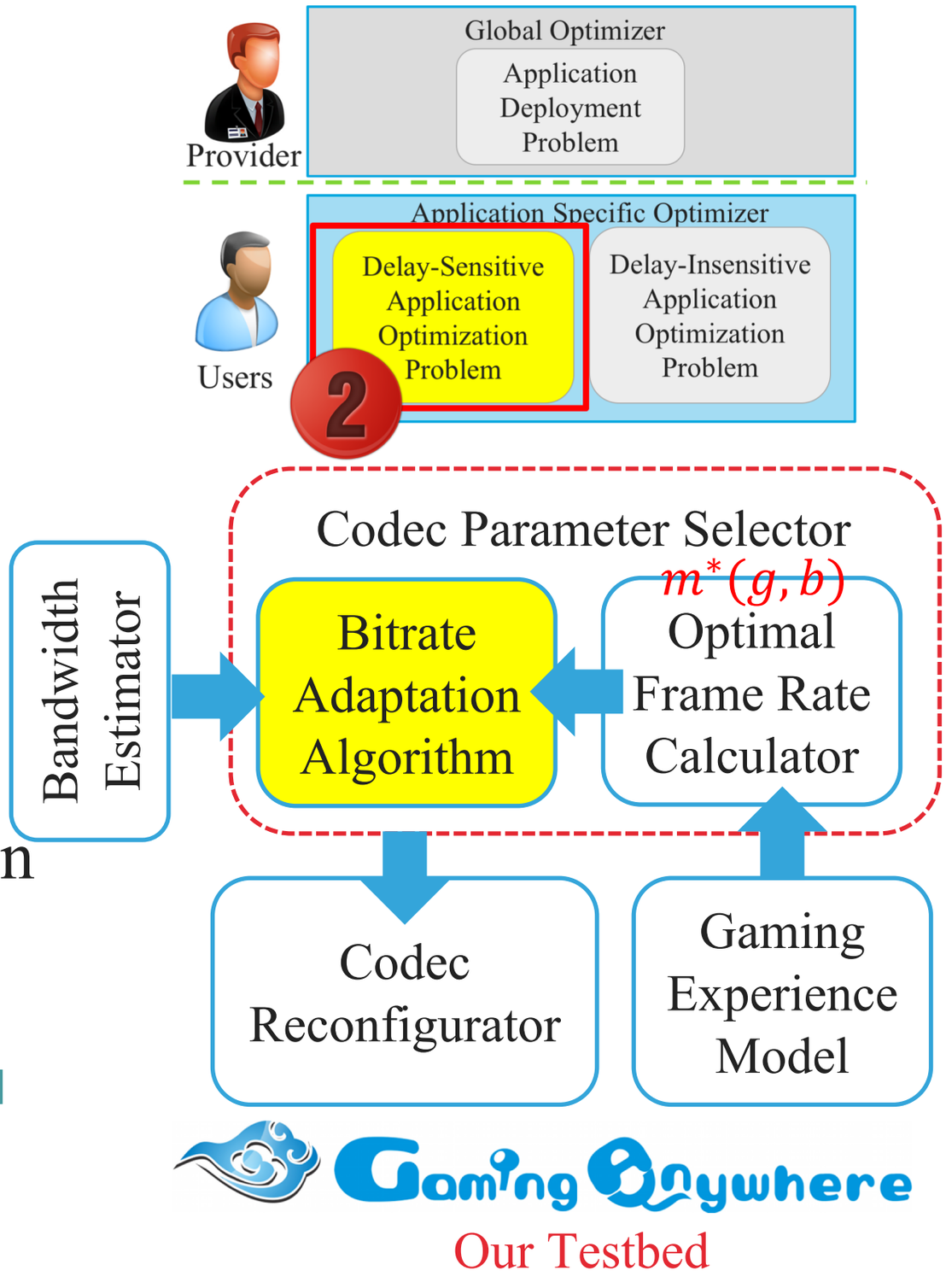
Summary: Delay-Sensitive Applications

- Solved bitrate adaptation problem
  - Two optimal algorithms
  - Outperform baselines by up to 30% and 46%, respectively
- Solved bandwidth estimation
  - > 80% accuracy
- Solved real-time codec reconfiguration

[Hong et al. TCSVT'15, Credits: collaborators @ NCTU]

- Derived gaming experience models

[Hong et al. TCSVT'15, Credits: collaborators @ Academia Sinica]

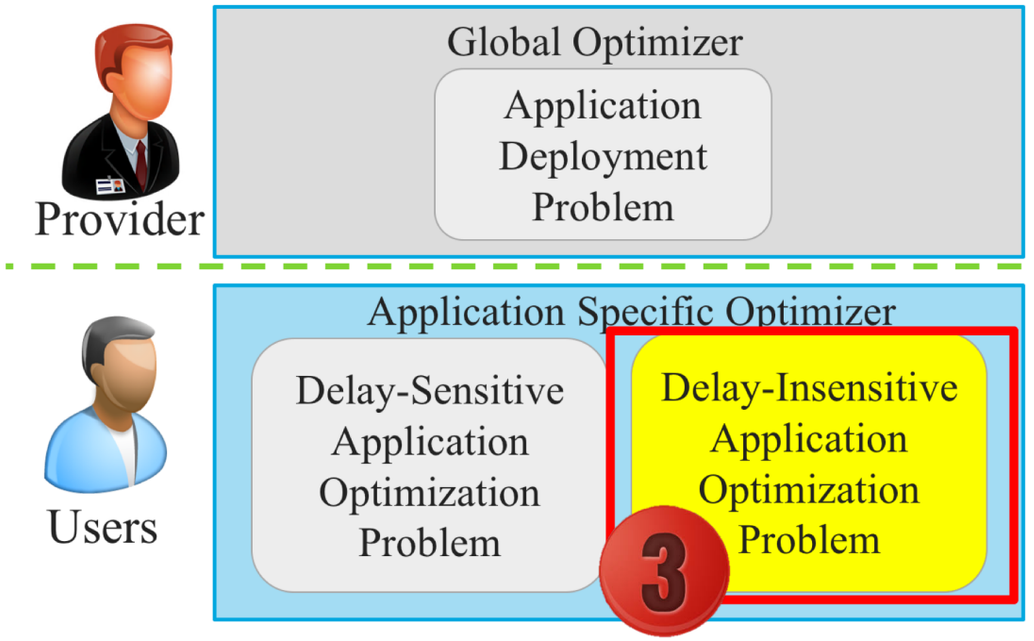


Agenda

I. Introduction
II. Application Deployment Problem
III. Delay-Sensitive Application Optimization Problem
IV. Delay-Insensitive Application Optimization Problem 3
V. Conclusion & Future Work

Agenda: Delay-Insensitive Application Optimization Problem

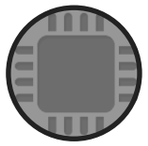
- Problem Statement
- Proposed Algorithm
- Evaluation & Summary



Content Delivery Under Challenged Network

Provider

Xeon
E-2176



Cat7 Gigi

Fog Devices

PM 2.5

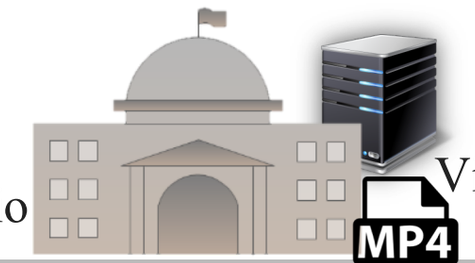
1TB
SSD

No
Network Access

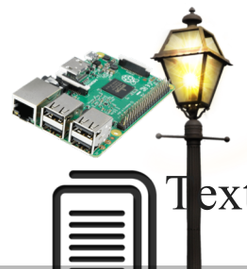
Network Access

Intermittent

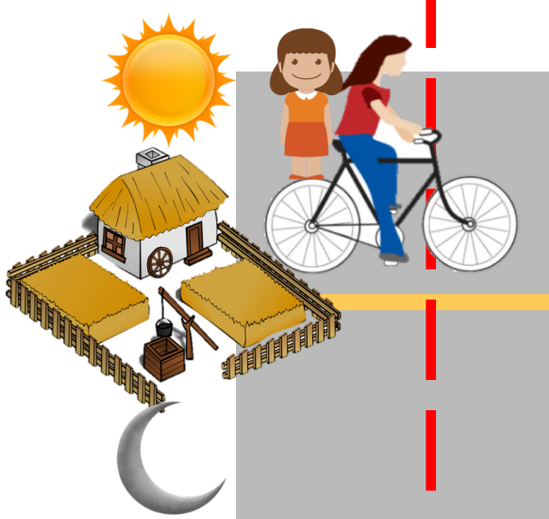
Audio



Video



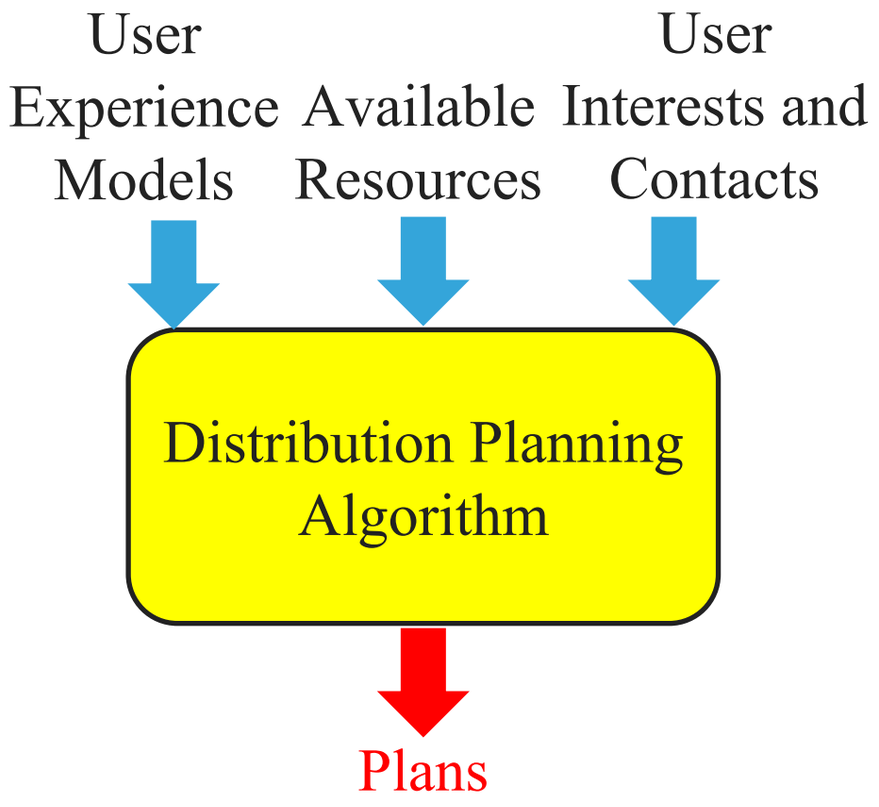
Text



Read Content
Without the Internet

Distribution Planning Problem

Problem: Given a set of contents $\boldsymbol{N}$ and a set of users $\boldsymbol{U}$. Each content has $\boldsymbol{L}$ representations. Each user has $\boldsymbol{C}$ contacts (run into fog devices). Each contact has $\boldsymbol{R}$ resources. Our problem is to determine delivering which representation $\boldsymbol{l} \in \boldsymbol{L}$ of which content $\boldsymbol{n} \in \boldsymbol{N}$ to which user $\boldsymbol{u} \in \boldsymbol{U}$ at which contact $\boldsymbol{c} \in \boldsymbol{C}$ without overloading the resources $\boldsymbol{R}$. Our goal is to maximize user experience.

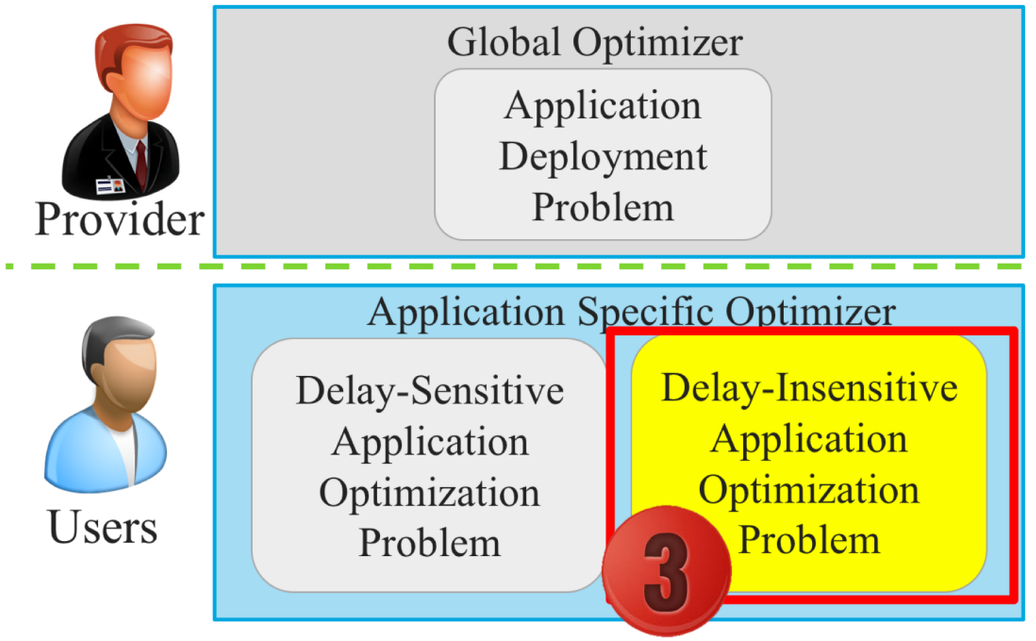


Challenges:

- Definition of user experience → user study
- Various user behaviors → individual distribution plans
- Limited resources and large size of contents → multiple representations

Agenda: Delay-Insensitive Application Optimization Problem

- Problem Statement
- Proposed Algorithm
- Evaluation & Summary



Distribution Planning Problem Formulation

Decision Variable

$$\max \sum_{u=1}^{U} \sum_{n=1}^{N} \sum_{l=1}^{L} \sum_{c=1}^{C_{u'}} x_{u,n,l,c} \rho_{nL+l} \psi_{u,n} \quad (1a)$$

Objective: UE
Maximize User Experience

$$st: \sum_{n=1}^{N} \sum_{l=1}^{L} b_{nL+l} x_{u',n,l,c'} \leq R_{u',c'}; \quad (1b)$$

Resource Budgets R

$$\sum_{c=1}^{C_{u'}} x_{u',n',l',c} \geq \sum_{c=1}^{C_{u'}} x_{u',n',l'',c}; \quad (1c)$$

Layer Dependency

$$\psi_{p_{u',c'},n'} \geq \bar{\psi} x_{u',n',l',c'}; \quad (1d)$$

Viewing Probability Checker

$$\sum_{c=1}^{C_{u'}} x_{u',n',l',c} \leq 1; \quad (1e)$$

Duplication Avoidance

Dynamic Programming

- Termination: (1b) ~ (1e)
- Recursion & Memorization:

$$DP(i, R_1, R_2, \dots, R_C) = \max\{$$
$$DP\,(i-1, R_1 - b[i], R_2, R_3 \dots) + \text{UE},$$
$$DP\,(i-1, R_1, R_2 - b[i], R_3 \dots) + \text{UE},$$
$$DP\,(i-1, R_1, R_2, R_3 - b[i] \dots) + \text{UE},$$
$$\dots$$
$$DP\,(i-1, \boldsymbol{R_u})\}$$

(1a) Objective: UE
Maximize User Experience

(1b) Resource Budgets R

(1c) Layer Dependency

(1d) Viewing Probability Checker

(1e) Duplication Avoidance

- Time & Space complexity: $O(NL\,\Pi R_c)$

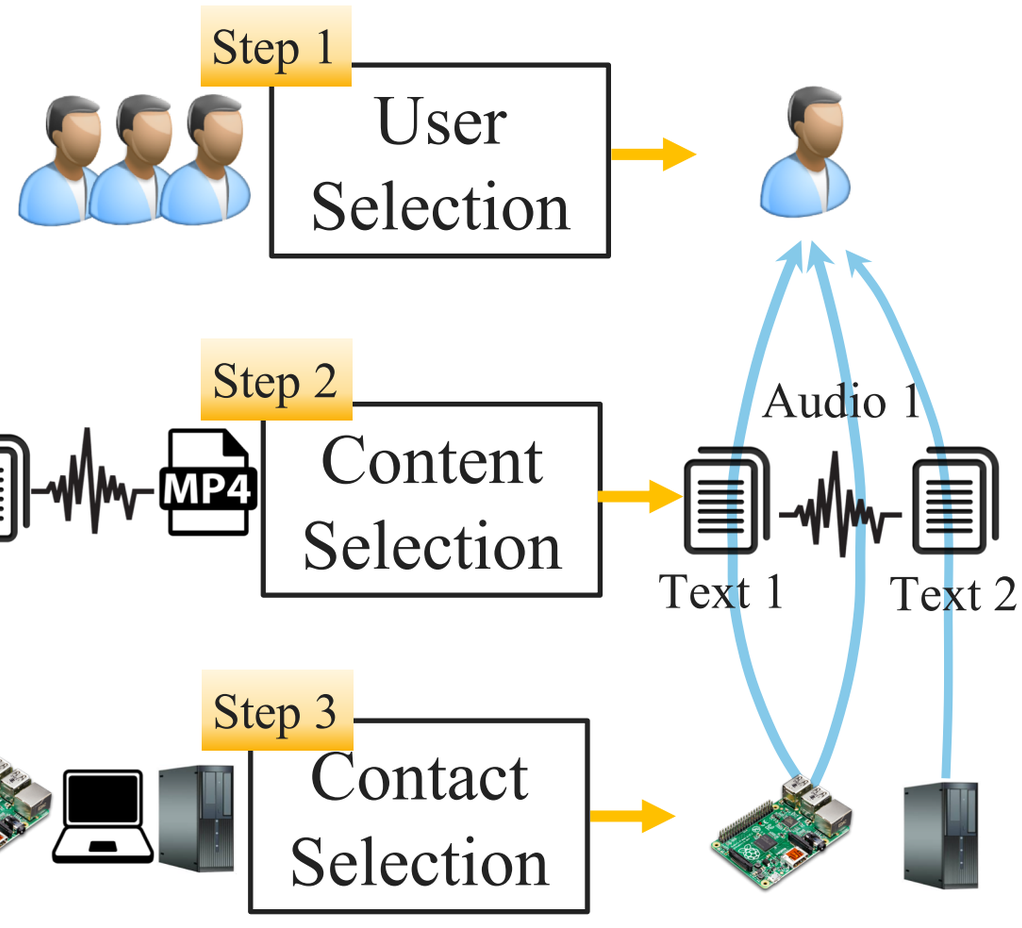
DP can optimally solve the problem under delay-tolerable environment

No. Contents	1	2	3	4	5	6	7
Running Time (sec)	0.08	0.11	0.19	0.61	1.8	111.2	457.2
Memory Usage (MB)	21	32	103	269	1918	10197	> 64GB

Algorithm: Contact-Driven Round Robin (CDRR)

1. User Selection
   - R Round Robin
2. Content Selection
   - R Higher user experience using less resources first
   - UE More contents for popular users
3. Contact Selection
   - R Unpopular fog devices first

$O(UNLlog(UNL) + UNLC)$

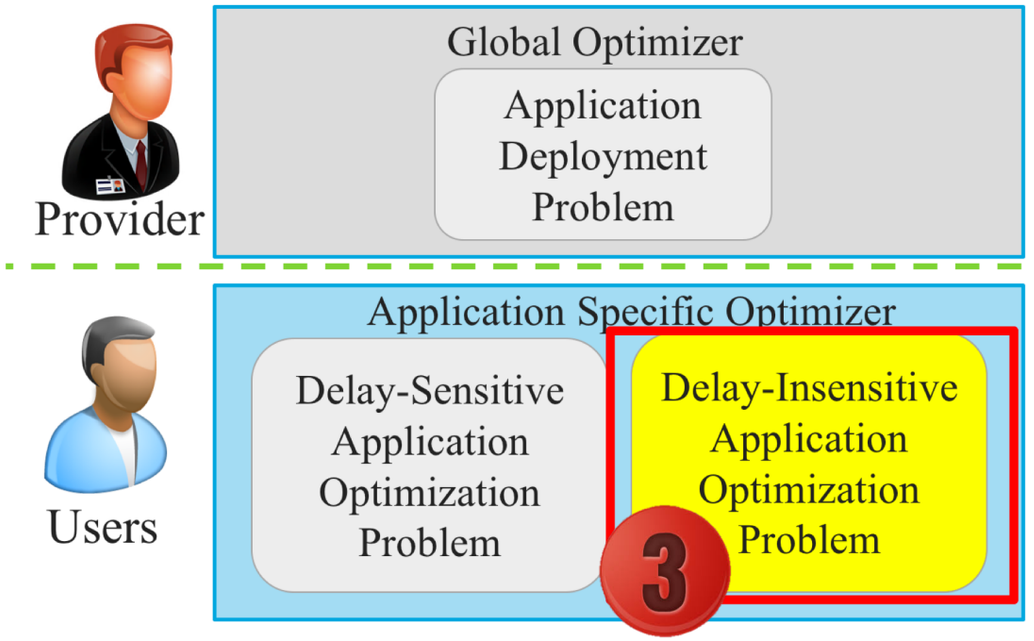


No. Content	Total User Experience		
	DP	CDRR	Perf. Gap
1	0.43	0.42	3%
2	0.36	0.34	6%
3	0.33	0.31	6%
4	0.31	0.29	6%
5	0.27	0.25	7%
6	0.29	0.27	7%
7	0.29	0.27	7%

DP vs CDRR Performance Gap: 7%

Agenda: Delay-Insensitive Application Optimization Problem

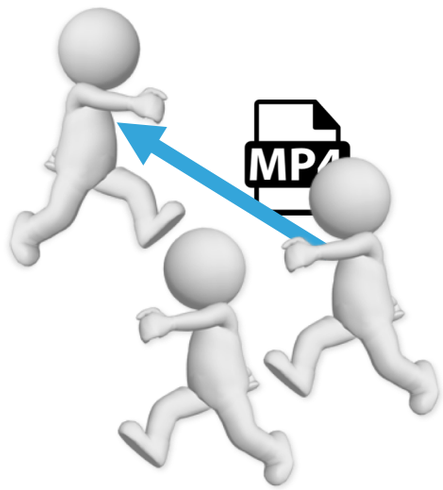
- Problem Statement
- Proposed Algorithm
- Evaluation & Summary



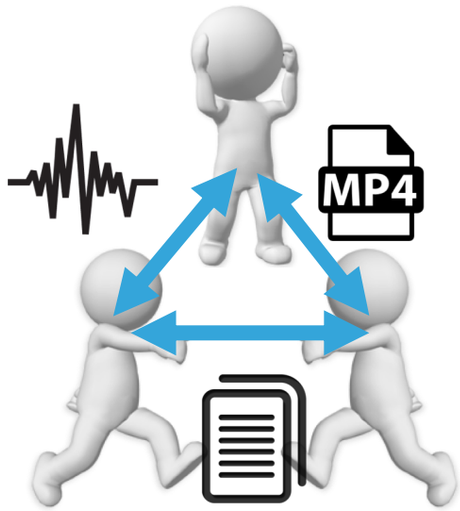
Evaluation Setup

- User experience: user study
- Trace: a real testbed
- Baseline algorithms
  - CSI [1]: Consider trajectories of users User Behaviors
  - Epidemic [2]: Classic content delivery algorithm in challenged networks

Classic Algorithm



CSI: Deliver to users having different trajectories



Epidemic: Broadcast to each other

[1] W. Hsu, D. Dutta, and A. Helmy, "CSI: A paradigm for behavior- oriented profile-cast services in mobile networks," *Ad Hoc Networks*, vol. 10, no. 8, pp. 1586–1602, 2012.
[2] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., 2000.

User Experience of Different Representations

- Definition of user experience: **understanding level**
- 182 participants (recruited)
  - 120 participants (filtered) with 587 samples
- Content: Apple Daily News
- 5-layer representations: text, audio, low-, medium-, and high- resolution videos

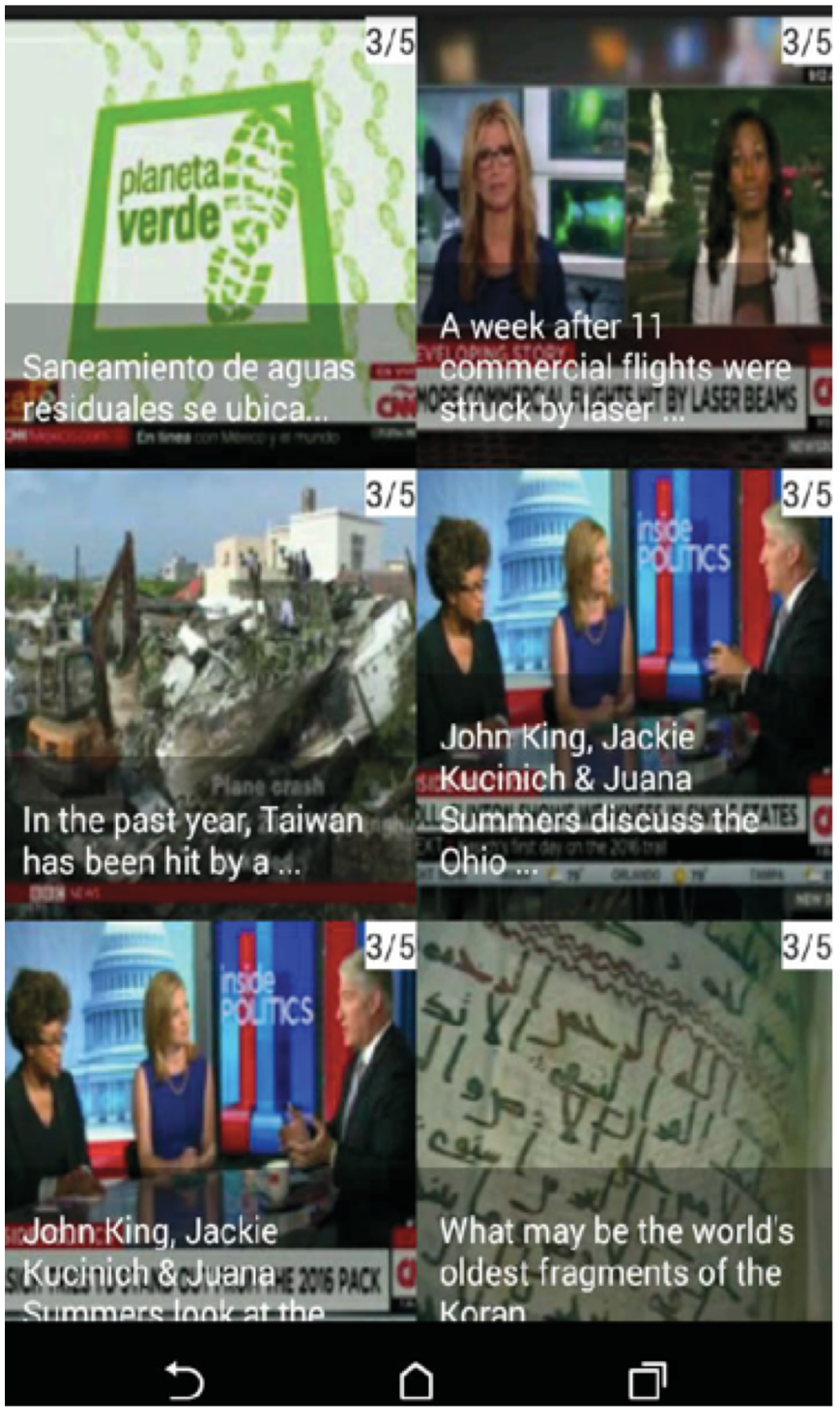
CROWDSOURCED USER EXPERIENCE SCORES

	Article	Audio	240p Video	360p	480p
Average	55%	68%	71%	74%	77%
No. Samples	112	112	134	104	125

Implementation

- Android Application
  - 15 Users
- Fog device: 11 Raspberry PIs

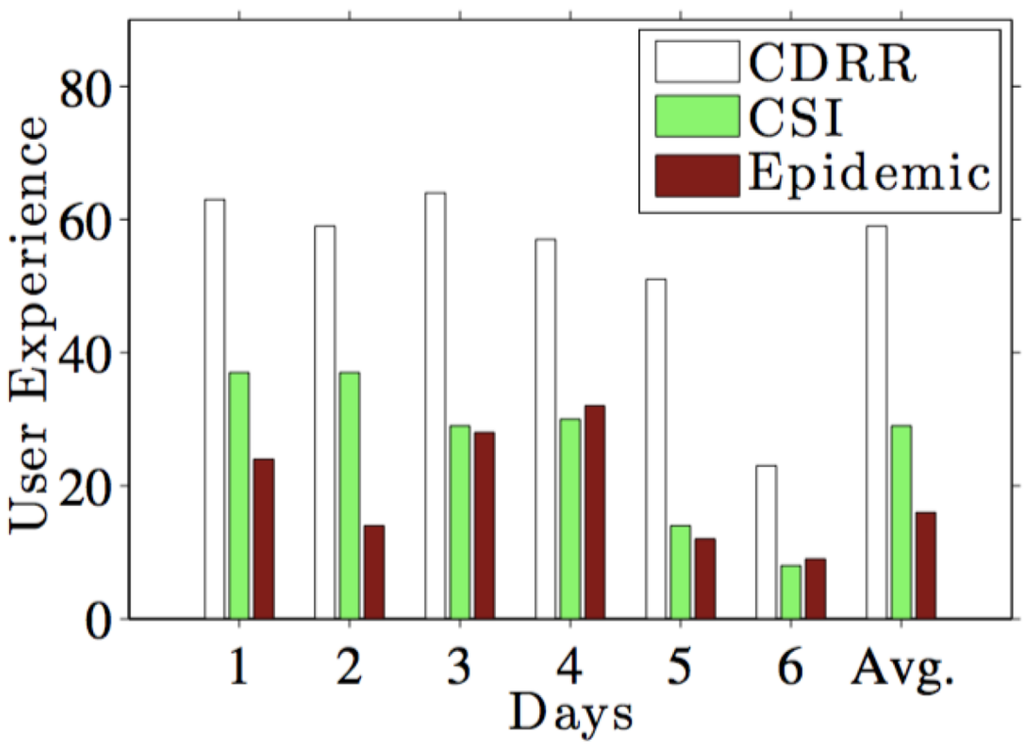
- Location: 3 rural villages + a university
- Content: 46 news from apple daily

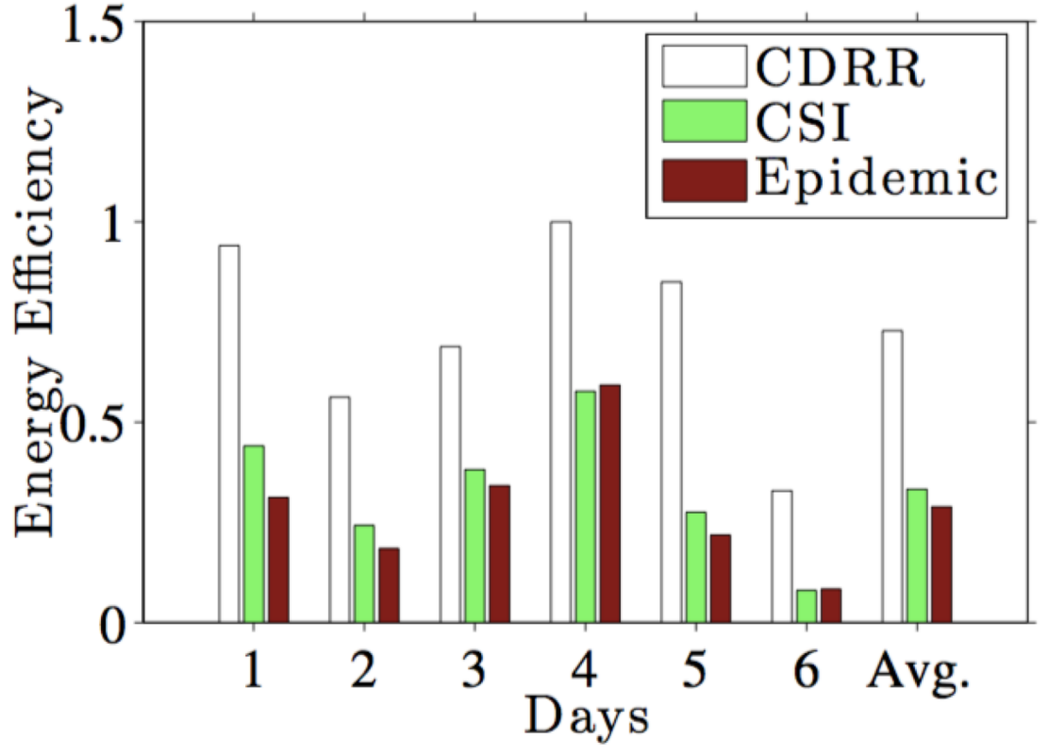


CDRR Results in Higher User Experience Using Less Energy

- CDRR Outperforms others:
  - User experience: 1.1 times (CSI) and 2.7 times (Epidemic)
  - Energy efficiency: 1.2 times (CSI) 1.4 times (Epidemic)

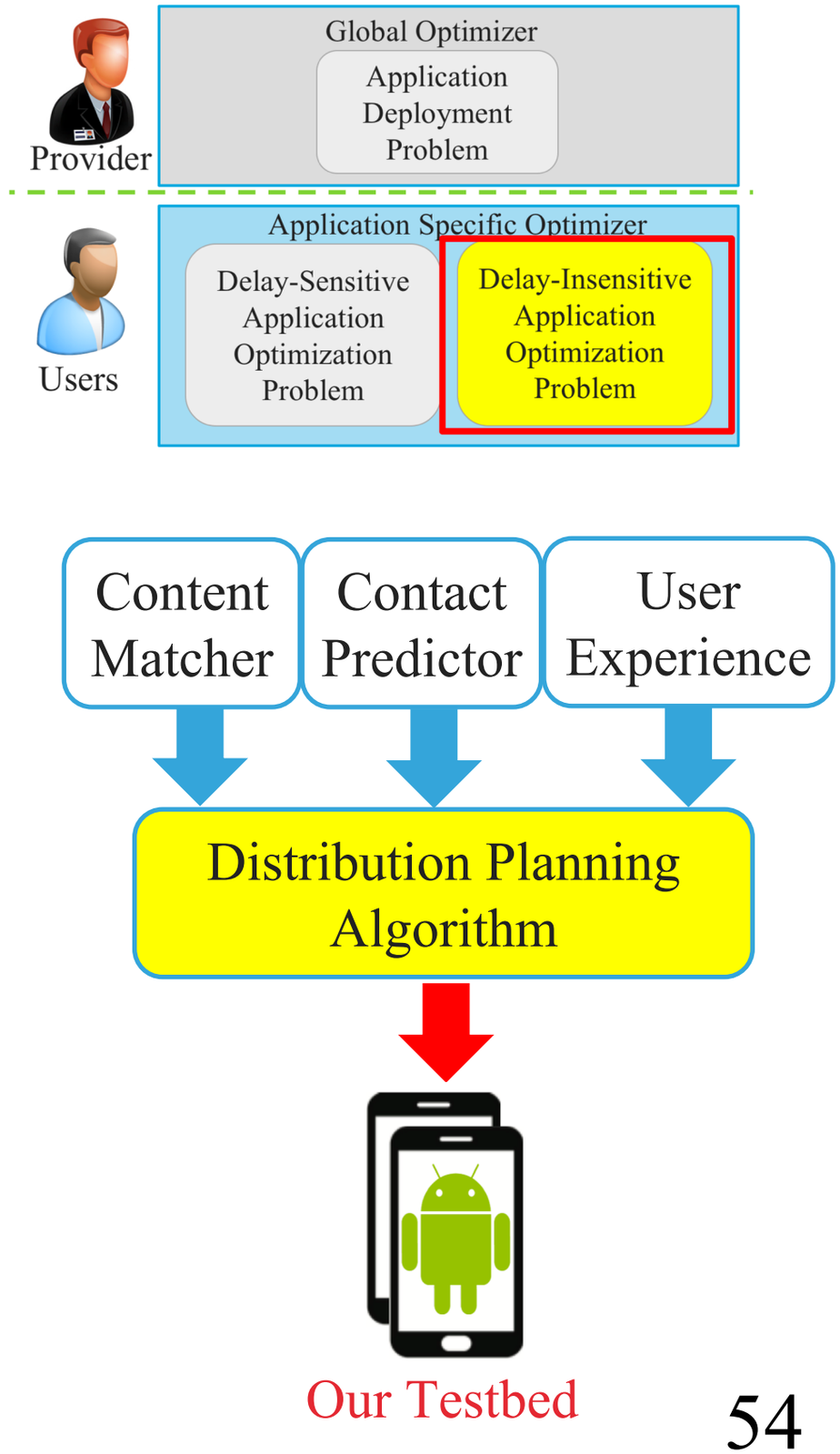
Because of detailed individual plans



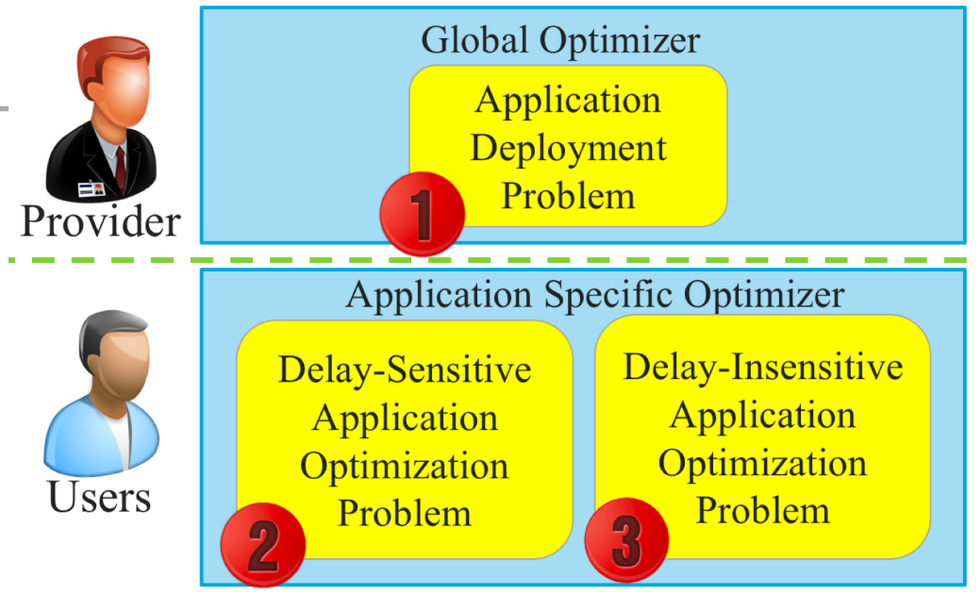


Summary: Delay-Insensitive Application

- Solved distribution planning problem
- Derived user experience models of different representations
- Evaluated our algorithm with real implementation
  - Outperform state-of-the-art and classic algorithms by 110% and 270% in terms of user experience, respectively



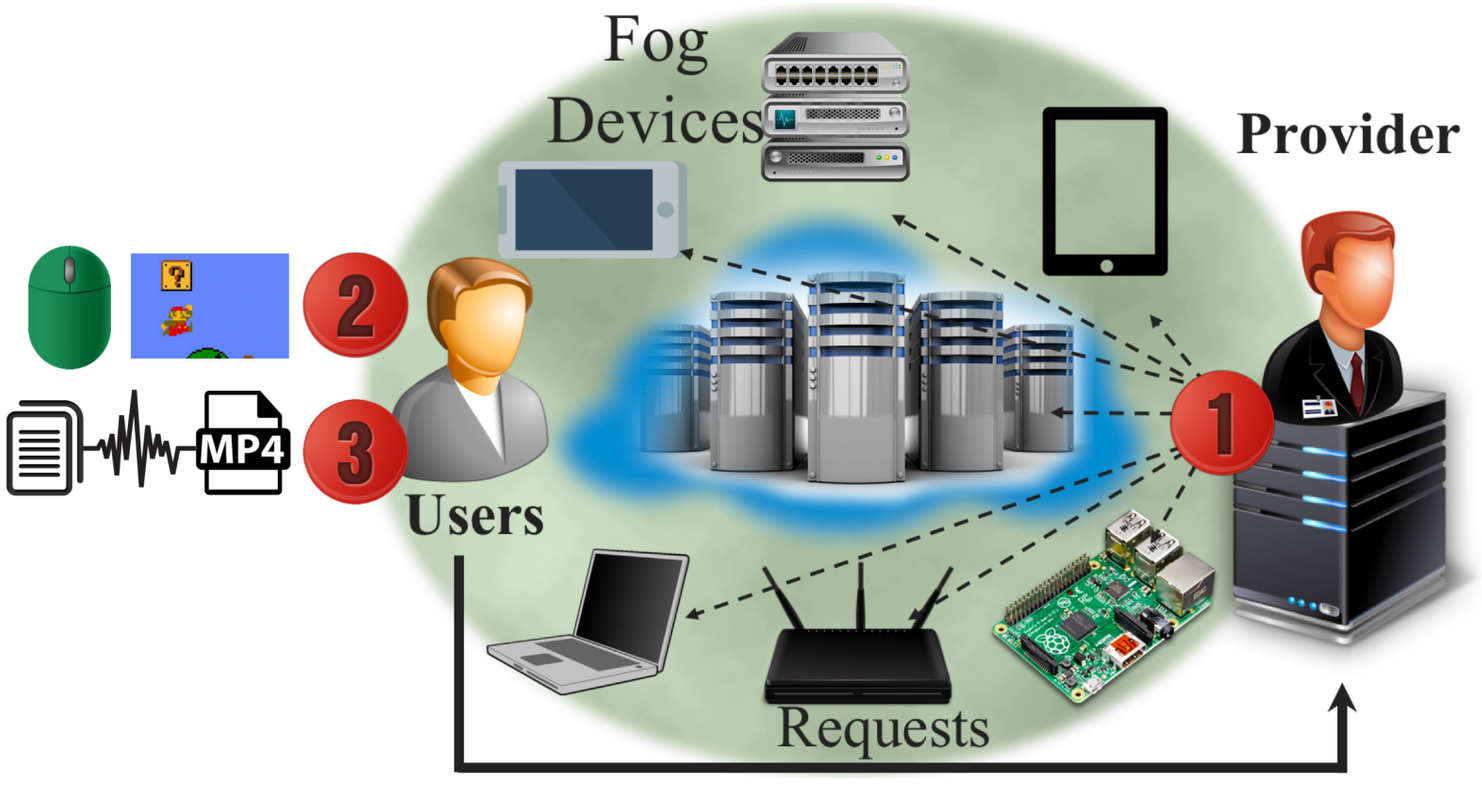
Agenda



I. Introduction
II. Application Deployment Problem
III. Delay-Sensitive Application Optimization Problem
IV. Delay-Insensitive Application Optimization Problem
V. Conclusion & Future Work

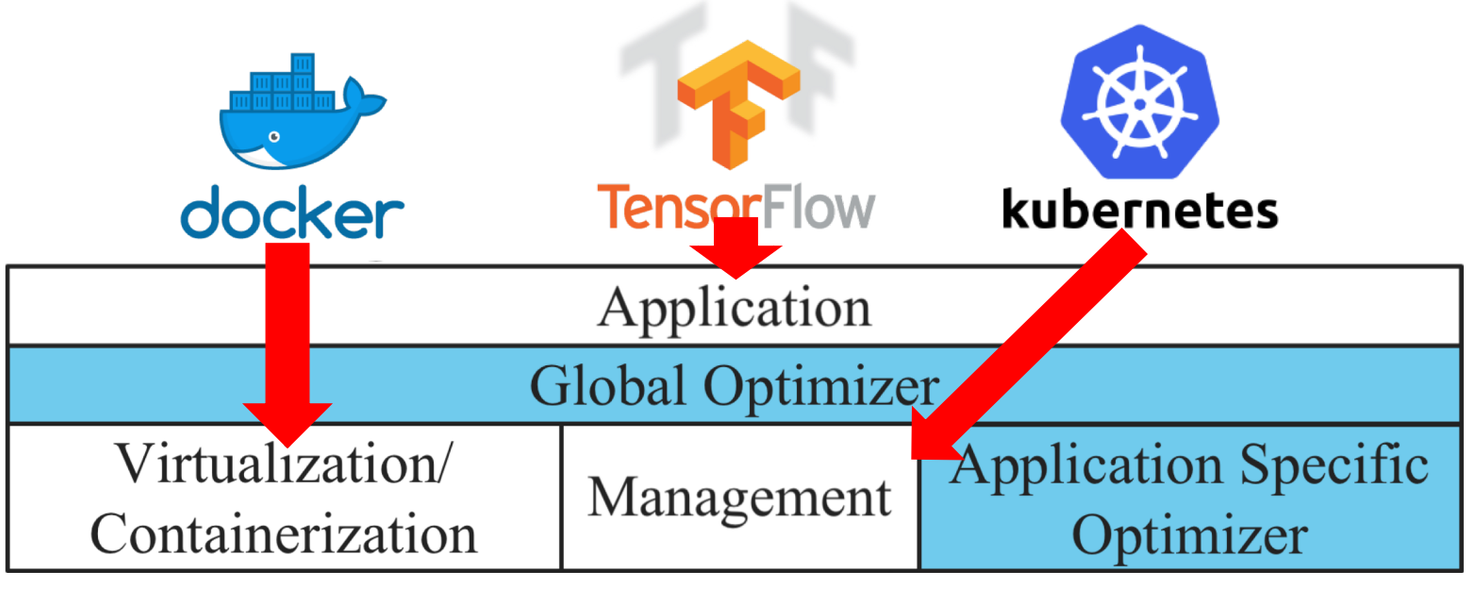
Conclusion

- Help providers to realize the cloud-to-things continuum platform
  1. A performance guaranteed deployment algorithm makes providers' life easier to estimate costs/benefits and formulate prices
- Help users to have optimized user experience
  2. Game streaming applications with optimal gaming experience
  3. Content delivery applications with maximized understanding level



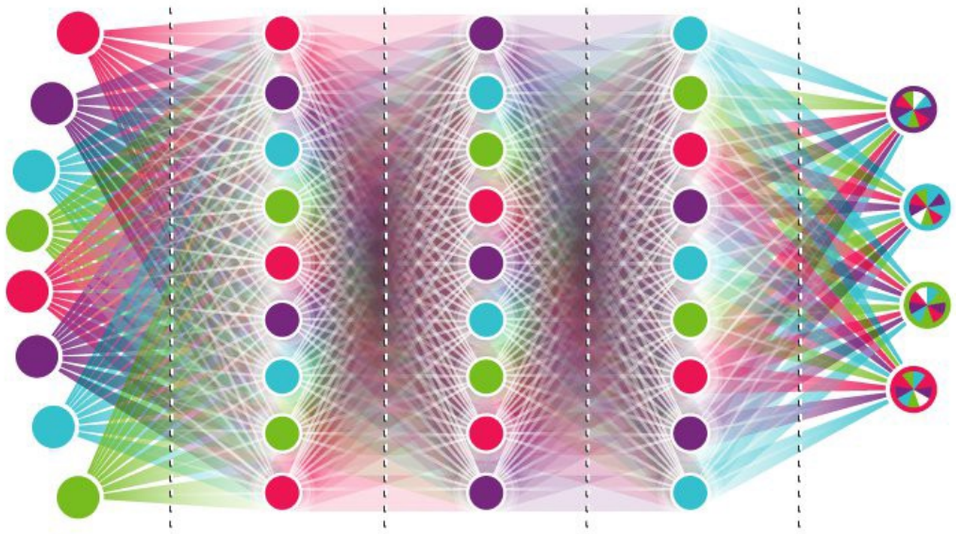
Future Work: Testbed in NTHU

- 8 Smart Street Lamps in NTHU
- Street Lamp
  - Fog devices: Raspberry Pis and IPCs
  - Sensors: camera, air pollution sensors, …
  - Communications: ethernet, WiFi mesh, LoRa, Zigbee, and Bluetooth
- Analytic applications: object recognition, car plate recognition, …

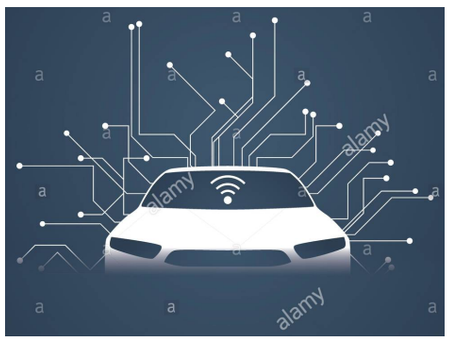


Killer Applications of Our Platform

- Analytic Applications
  - Complicated deep learning models
    → huge amount of resources and data is required



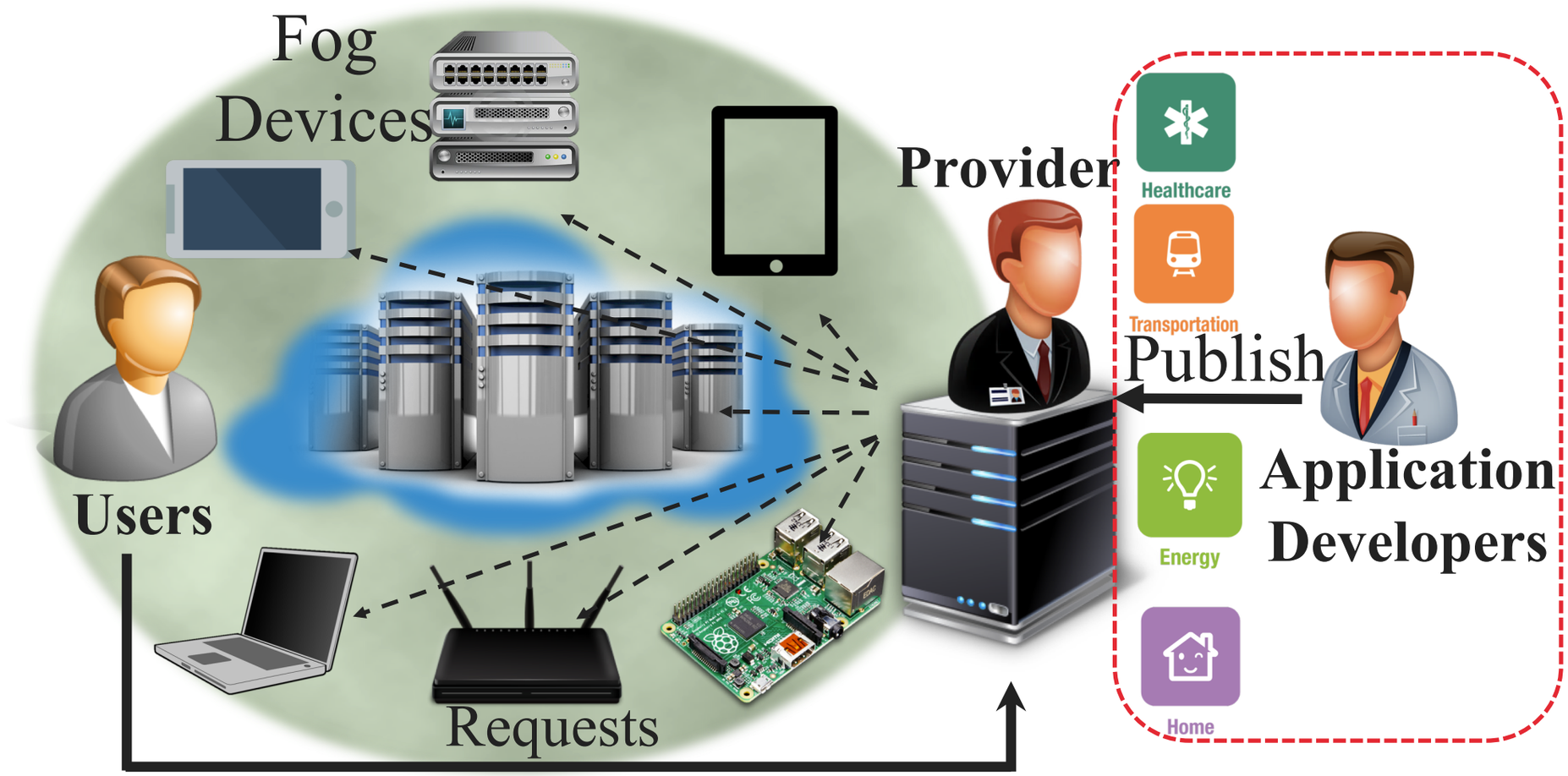
- IoT Applications [1]
  - Low end devices
    → need longer time to run complicated applications



Intelligent IoT application with complicated analytics:
e.g., Human tracking applications on wearable glasses

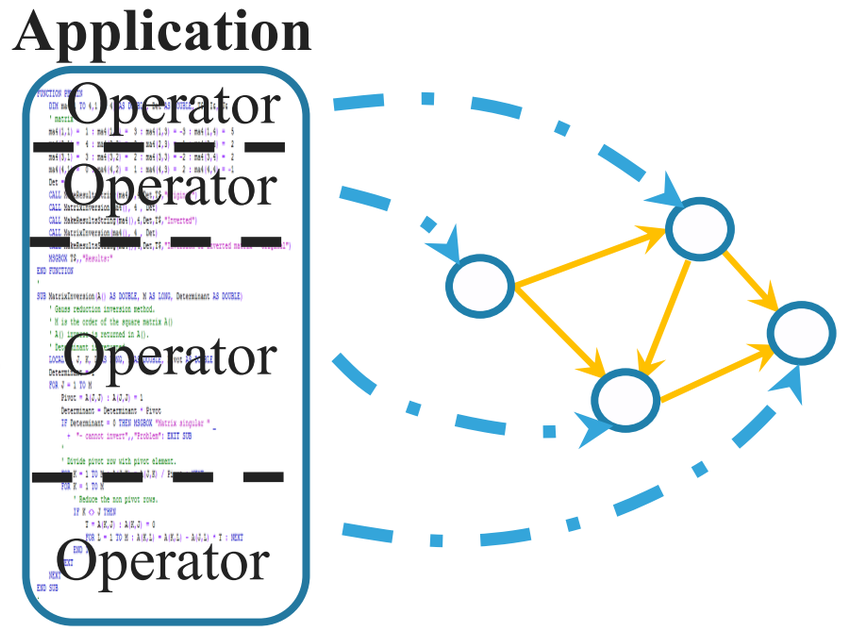
[1] M. Golkarifard, J. Yang, Z. Huang, A. Movaghar, and P. Hui, "Dandelion: A Unified Code Offloading System for Wearable Computing," *IEEE Transactions on Mobile Computing,* 2018.

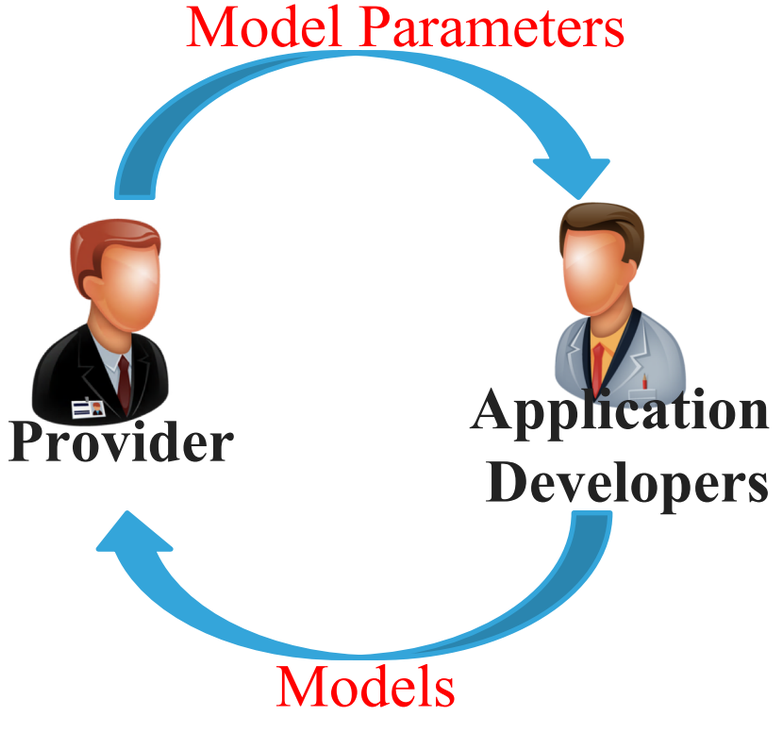
Ecosystem


Fog
Devices
Provider
Healthcare
Transportation
Publish
Application
Developers
Energy
Home
Users
Requests

For Application Developers

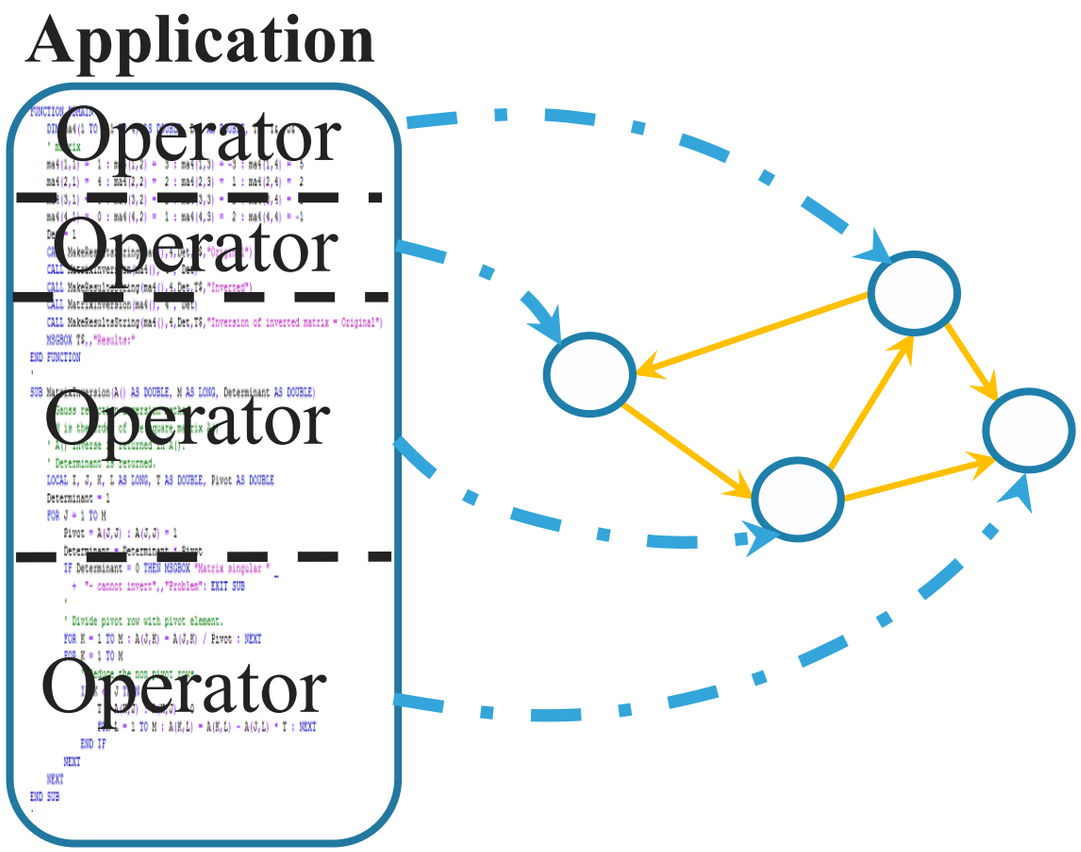
- Challenges
  - High learning curve to implement multi-operator applications → programming model
  - Privacy sensitive data → data management and protection approaches





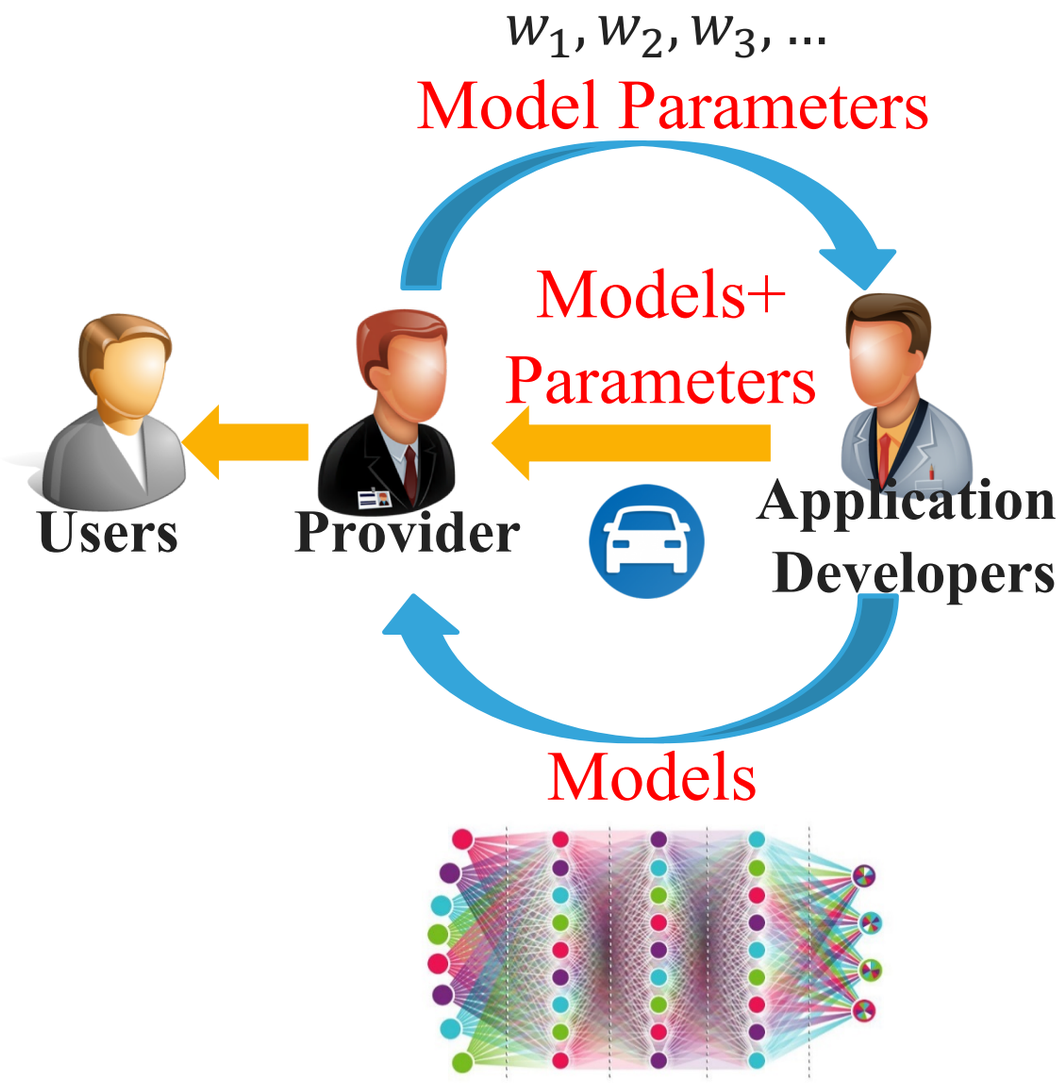
Programing Model

- Decomposition
  - Dynamically split applications into any number of smaller operators
- Communication
  - Automatically handle the communications between operators
- Performance Optimization
  - Stream processing
  - Parallel programming



Data Management and Protection Approaches

- Training as a Service (TaaS)
  - Help developers to train their models
- Data analytic APIs
  - Help developers analyze the data and figure out some characteristics
  - Help developers to figure out root cause of low accuracy



Publications

Journal Articles

2 **H. Hong**, C. Hsu, T. Tsai, C. Huang, K. Chen, and C. Hsu, "Enabling Adaptive Cloud Gaming in an Open-Source Cloud Gaming Platform," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), vol. 25, no. 12, December, 2015.

3 **H. Hong**, T. El-Ganainy, C. Hsu, K. Harras, and M. Hefeeda, "Disseminating Multi-layer Multimedia Content over Challenged Networks," IEEE Transactions on Multimedia (TMM), vol.20, no.2, February, 2017.

Conference Papers

1 **H. Hong**, P. Tsai, A. Cheng, M. Uddin, N. Venkatasubramanian, and C. Hsu, "Supporting Internet-of-Things Analytics in a Fog Computing Platform," in Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, December, 2017. **(Best Paper Award)**

1 **H. Hong**, P. Tsai, and C. Hsu, "Dynamic Module Deployment in a Fog Computing Platform," in Proc. of IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, Japan, October, 2016. **(Best Paper Award)**

Demo Paper

3 **H. Hong**, S. Wang, C. Tan, T. El-Ganainy, K. Harras, C. Hsu and M. Hefeeda, "Challenged Content Delivery Network: Eliminating the Digital Divide," in Proc. of ACM Multimedia Demo Paper (MM Demo), Brisbane, Australia, Oct., 2015.

PhD Symposium

13 **Hua-Jun Hong**, "From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices," in Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, December, 2017.

Other Publications

Book Chapter

- C. Hsu, **H. Hong**, T. El-Ganainy, K. Nahrstedt, and N. Venkatasubramanian, "Multimedia Fog Computing: Minions in the Cloud and Crowd," ACM Frontiers of Multimedia Research, Chapter 10, Association for Computing Machinery and Morgan & Claypool, January 2018.

Journal Articles

- **H. Hong**, C. Fan, Y. Lin, and C. Hsu, "Optimizing Cloud-Based Video Crowdsensing," IEEE Internet of Things Journal (JIoT), vol. 3, no. 3, January, 2016.
- **H. Hong**, D. Chen, C. Huang, K. Chen, and C. Hsu, "Placing Virtual Machines to Optimize Cloud Gaming Experience," IEEE Transactions on Cloud Computing (TCC), vol. 3, no. 1, 2014.

Conference and Workshop Papers

- M. Rahman, A. Rahman, A. Afrin, **H. Hong**, P. Tsai, M. Uddin, N. Venkatasubramanian, and C. Hsu, "Adaptive Sensing Using Internet-of-Things with Constrained Communications," in Proc. of ACM Adaptive and Reflective Middleware (ARM), Las Vegas, NV, USA, December, 2017.
- P. Tsai, **H. Hong**, A. Cheng, and C. Hsu, "Distributed Analytics in Fog Computing Platforms Using TensorFlow and Kubernetes," in Proc. of IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, Korea, September, 2017.
- Y. Chen, **H. Hong**, S. Yao, A. Khunvaranont, and C. Hsu, "Gamifying Mobile Applications for Smartphone Augmented Infrastructure Sensing," in Proc. of IEEE Annual Workshop on Network and Systems Support for Games (NetGames), Taipei, Taiwan, June, 2017.
- **H. Hong**, Y. Lin, J. Chuang, and C. Hsu, "Animation Rendering on Multimedia Fog Computing Platforms," in Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg, December, 2016.
- T. Fan-Chiang, **H. Hong**, C. Hsu, "Segment-of-Interest Driven Live Game Streaming: Saving Bandwidth without Degrading Experience," in Proc. of IEEE Annual Workshop on Network and Systems Support for Games (NetGames), Zagreb, Croatia, December, 2015.
- **H. Hong**, C. Lee, K. Chen, C. Huang, and C. Hsu, "GPU Consolidation for Cloud Games: Are We There Yet?," in Proc. of IEEE Annual Workshop on Network and Systems Support for Games (NetGames), Nagoya, Japan, December, 2014.

Poster and Demo Papers

- Y. Hsieh, **H. Hong**, P. Tsai, Y. Wong, Q. Zhu, M. Uddin, N. Venkatasubramanian, and C. Hsu, "Managed Edge Computing on Internet-of-Things Devices for Smart City Applications," in Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS), Taipei, Taiwan, April, 2018.
- Q. Zhu, M. Uddin, N. Venkatasubramanian, C. Hsu, and **H. Hong**, "Enhancing Reliability of Community Internet-of-Things Deployments with Mobility," in Proc. of IEEE INFOCOM, Honolulu, HI, USA, April, 2018.
- **H. Hong**, D. Chen, C. Huang, K. Chen, and C. Hsu, "QoS-Aware Virtual Machine Placement for Cloud Games," in Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames), Denver, CO, December 2013.

Q&A

hua.j.hong@gmail.com

Introduction

Fog similar concepts comparisons

Timeline (from WSN to clout-to-things continuum)

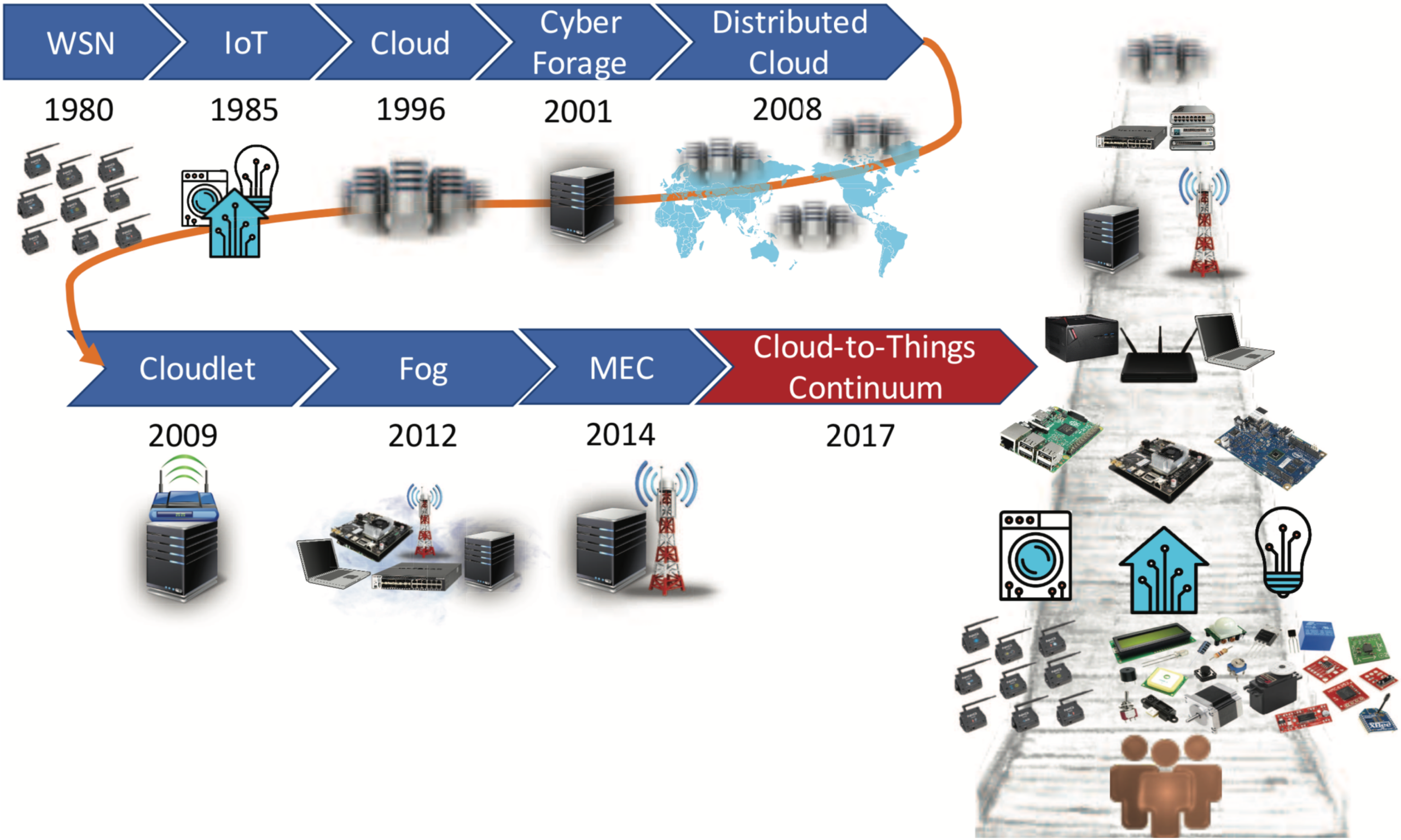
Benefits of Cloud-to-Things Continuum

Cloud, Distributed Cloud, Cyber Foraging, CloudLet, and Fog

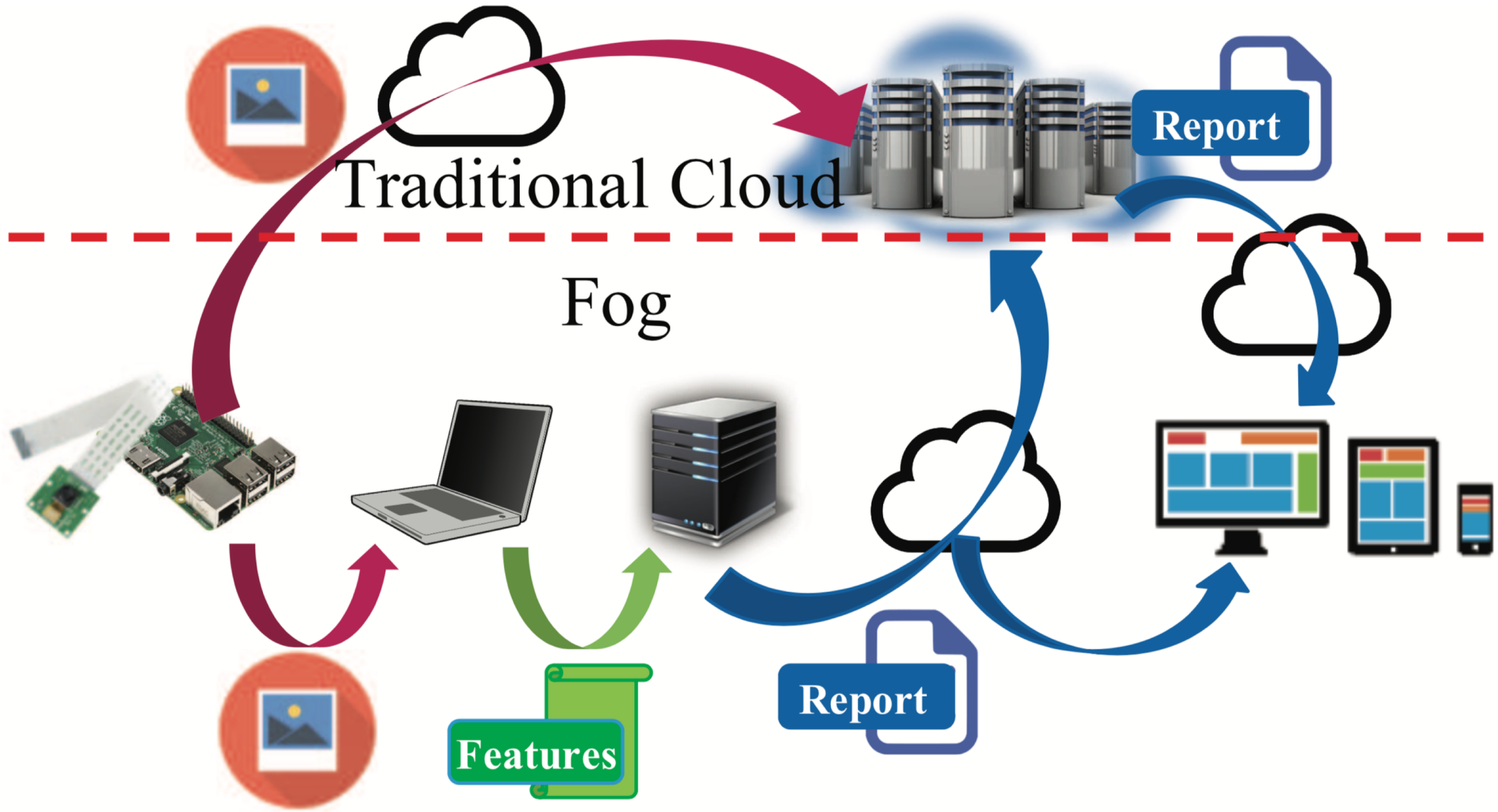
	Low Latency	Location Awareness and Mobility Support	Virtualization Support	High Heterogeneity
Cloud	✕	✕	✓	✕
Distributed Cloud [1]	Δ	Δ	✓	✕
Cyber Foraging [2]	✓	Δ	✕	Δ
CloudLet and MEC [3,4]	✓	✓	✓	Δ
Fog [5]	✓	✓	✓	✓

[1] P. Endo, A. de Almeida Palhares, N. Pereira, G. Goncalves, D. Sadok, J. Kelner, B. Melander, and J. Mangs, "Resource Allocation for Distributed Cloud: Concepts and Research Challenges," *IEEE Network*, 25(4), 42-46, 2011.
[2] M. Satyanarayanan. Pervasive computing: Vision and challenges. IEEE Personal communications, 8(4):10–17, 2001.
[3] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing. *IEEE pervasive Computing,* 8(4), 14-23, 2009.
[4] Mobile-edge computing. https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.
[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things" *in Proc. of ACM SIGCOMM workshop on Mobile Cloud Computing (MCC)*, Helsinki, Finland, 2012.

Timeline


WSN
IoT
Cloud
Cyber Forage
Distributed Cloud
1980
1985
1996
2001
2008
Cloudlet
Fog
MEC
Cloud-to-Things Continuum
2009
2012
2014
2017

Benefits of Cloud-to-Things Continuum



Application Deployment Problem

Knapsack problem variations

Approximation factor proof

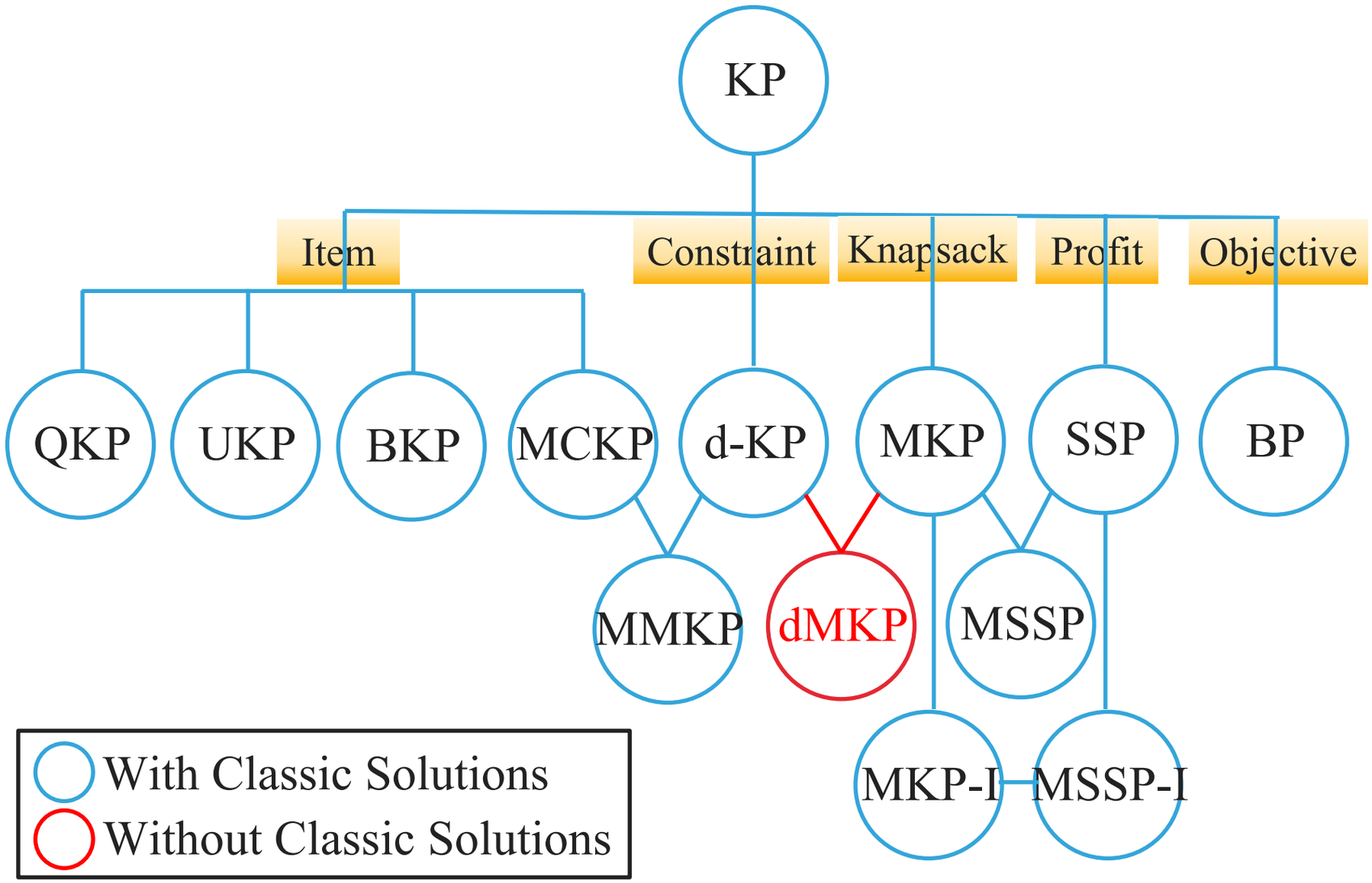
Resource consumption model

Knapsack Problem Variations

- Subset Sum Problem (SSP): weight = profit
- Bounded Knapsack Problem (BKP): duplicated items with a boundary
- Unbounded Knapsack Problem (UKP): duplicated items without a boundary
- Multi dimensional Knapsack Problem (d-KP): multiple constraints
- Multiple Knapsack Problem (MKP): multiple knapsacks
- Multiple Knapsack Problem with Identical Capacities (MKP-I): knapsacks have the same capacities
- Multiple-Choice Knapsack Problem (MCKP): items have variations (diff weight & profits), need to pick one variation of each item
- Quadratic Knapsack Problem (QKP): two items result in a weight & profit
- Bin Packing (BP): use least number of knapsack to pack all items
- MKP+SSP -> MSSP
- MKP-I + SSP -> MSSP-I
- MCKP+d-KP -> MMKP

[1] H. Kellerer, U. Pferschy, and D. Pisinger, “Knapsack Problems,” Springer, 2004
[2] Silvano Martello, “Knapsack Problems: Algorithms and Computer Implementations,” Wiley & Sons, 1990
[3] C. Wilbaut, S. Hanafi, and S. Salhi, “A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems. IMA Journal of Management Mathematics, 19(3), 227-244, 2008.

Knapsack Variation Graph



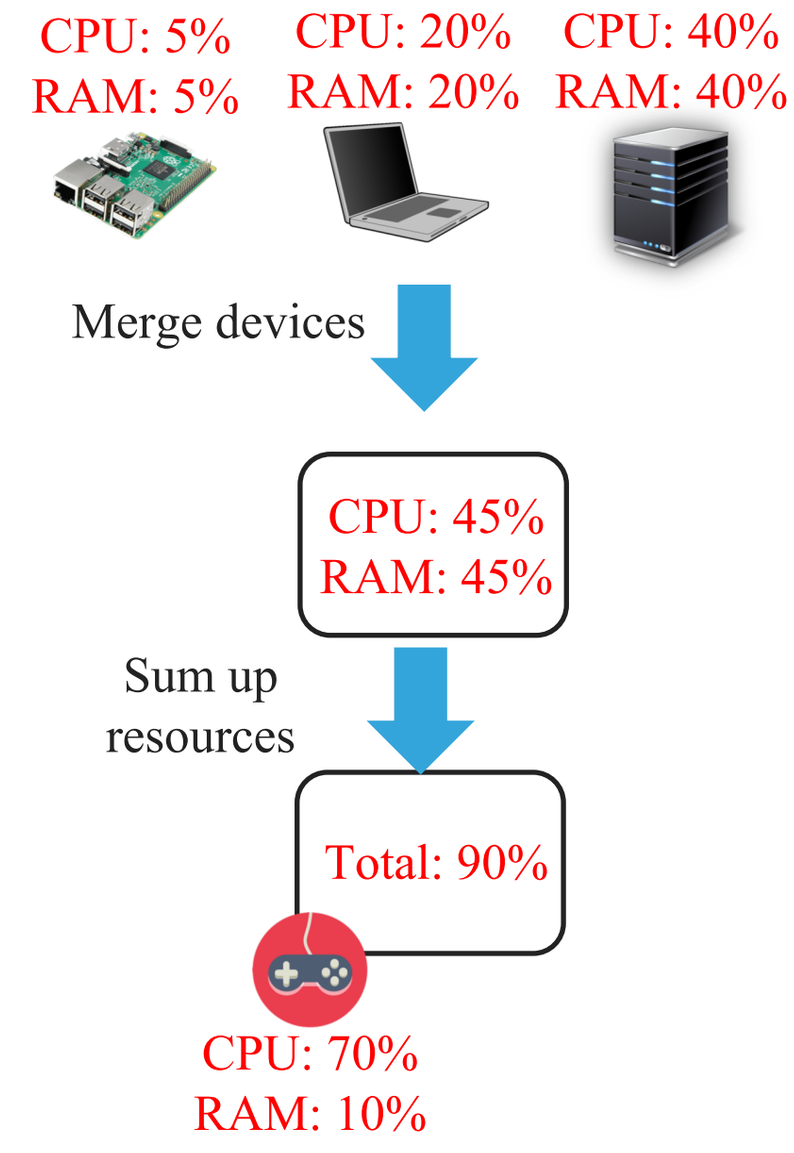
[1] H. Kellerer, U. Pferschy, and D. Pisinger, “Knapsack Problems,” Springer, 2004
[2] Silvano Martello, “Knapsack Problems: Algorithms and Computer Implementations,” Wiley & Sons, 1990
[3] C. Wilbaut, S. Hanafi, and S. Salhi, “A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems. IMA Journal of Management Mathematics, 19(3), 227-244, 2008.

Approximation Factor Proof

- Step1: Design an upper bound solution (OPT')
  - Intuitions are similar to APX
- Step2: Prove results of OPT' >= optimal solution (OPT)
- Step3: Derive approximation factor (APX/OPT')

Step1: Design an Upper Bound Solution (OPT')

- Similar intuitions with APX
- Differences
  - Merge all the fog devices
  - Sum up different type of resources
- Sort requests by total required resources



Step2: Prove that OPT' is Upper Bound

- OPT: $\dot{H}$ satisfied requests $(\dot{r}_1, \dot{r}_2, \cdots \dot{r}_{\dot{H}})$
- OPT': H' requests $(r'_1, r'_2, \cdots r'_{H'})$, which may/may not be satisfied
  - Results are sorted by total required resources

$$\sum_{h=1}^{\dot{H}} \sum_{u \in \mathbf{U}} F(\dot{r}_h, u, \cdots) \leq \sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u}$$

Device Capacities

Contradiction

$$\sum_{h=1} \sum_{u \in \mathbf{U}} F(r'_h, u, \cdots) > \sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u}$$

$\dot{H} \leq H'$

OPT' is an upper bound solution

$$\sum_{u \in \mathbf{U}} F(r'_h, u, \cdots) \leq \sum_{u \in \mathbf{U}} F(\dot{r}_h, u, \cdots) \quad \forall 1 \leq h \leq \dot{H}$$

Assum $\dot{H} > H'$

Step3: Derive Approximation Factor

- APX: H satisfied requests $(r_1, r_2, \cdots r_H)$

$$H \frac{\sum_{h=1}^{H} \sum_{u \in \mathbf{U}} F(r_h, u, \cdots)}{H} \geq \sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u}/|\mathbf{U}| \quad (1)$$

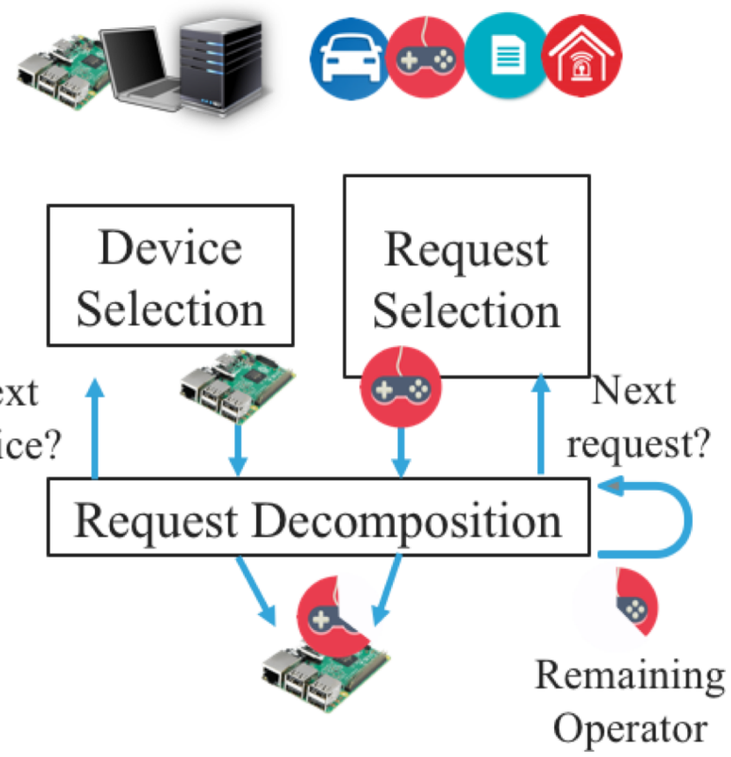
A

$$H' \frac{\sum_{h=1}^{H'} \sum_{u \in \mathbf{U}} F(r'_h, u, \cdots)}{H'} \leq \sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u} \quad (2)$$

B

B ≥ A

$$\sum_{u \in \mathbf{U}} F(r_h, u, \cdots) \leq \sum_{u \in \mathbf{U}} F(r_{h+1}, u, \cdots) \ \forall 1 \leq h \leq H \quad (3)$$



(1) and (2) ➡ $\frac{H}{H'} \geq \frac{1}{|U|}\frac{B}{A}$ ➡ $\frac{H}{H'} \geq \frac{1}{|U|}$ ➡ $\frac{H}{\dot{H}} \geq \frac{1}{|U|}$

B ≥ A H' ≥ $\dot{H}$

Remove the Proportional Heterogeneity Assumption

- APX: H satisfied requests $(r_1, r_2, \cdots r_H)$

$$LH \frac{\sum_{h=1}^{H} \sum_{u \in \mathbf{U}} F(r_h, u, \cdots)}{H} \geq \sum_{k \in \mathbf{V}} \min R_{k,u \in \mathbf{U}} L \qquad (1)$$

A

$$\mathrm{L} = \frac{\sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u}}{\sum_{k \in \mathbf{V}} \min R_{k,u \in \mathbf{U}}}$$

B ≥ A

$$H' \frac{\sum_{h=1}^{H'} \sum_{u \in \mathbf{U}} F(r'_h, u, \cdots)}{H'} \leq \sum_{k \in \mathbf{V}} \sum_{u \in \mathbf{U}} R_{k,u} \qquad (2)$$

B

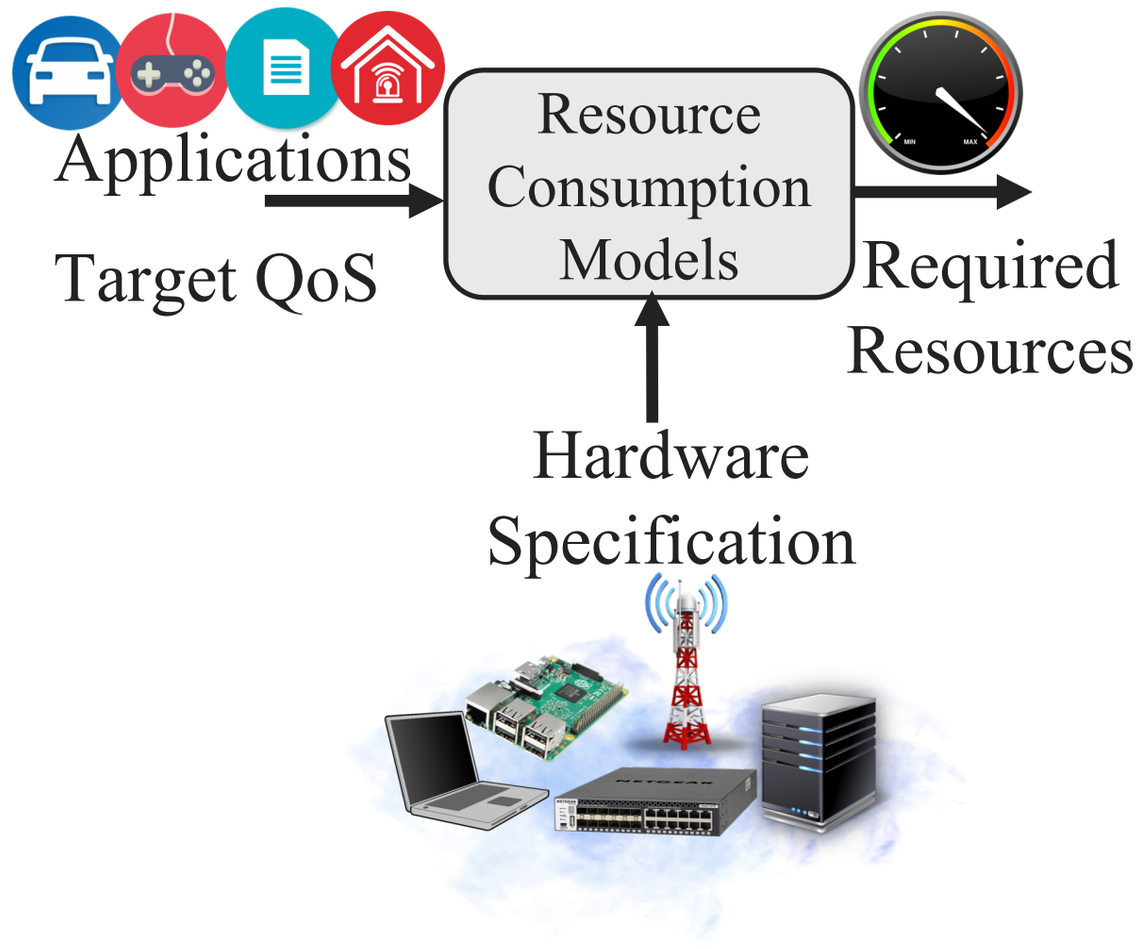
$$\sum_{u \in \mathbf{U}} F(r_h, u, \cdots) \leq \sum_{u \in \mathbf{U}} F(r_{h+1}, u, \cdots) \quad \forall 1 \leq h \leq H \qquad (3)$$

(1) and (2) ➡ $\frac{H}{H'} \geq \frac{1}{L}\frac{B}{A}$ ➡ $\frac{H}{H'} \geq \frac{1}{L}$ ➡ $\frac{H}{H'} \geq \frac{1}{L}$

B ≥ A　　H’ ≥ $\dot{H}$

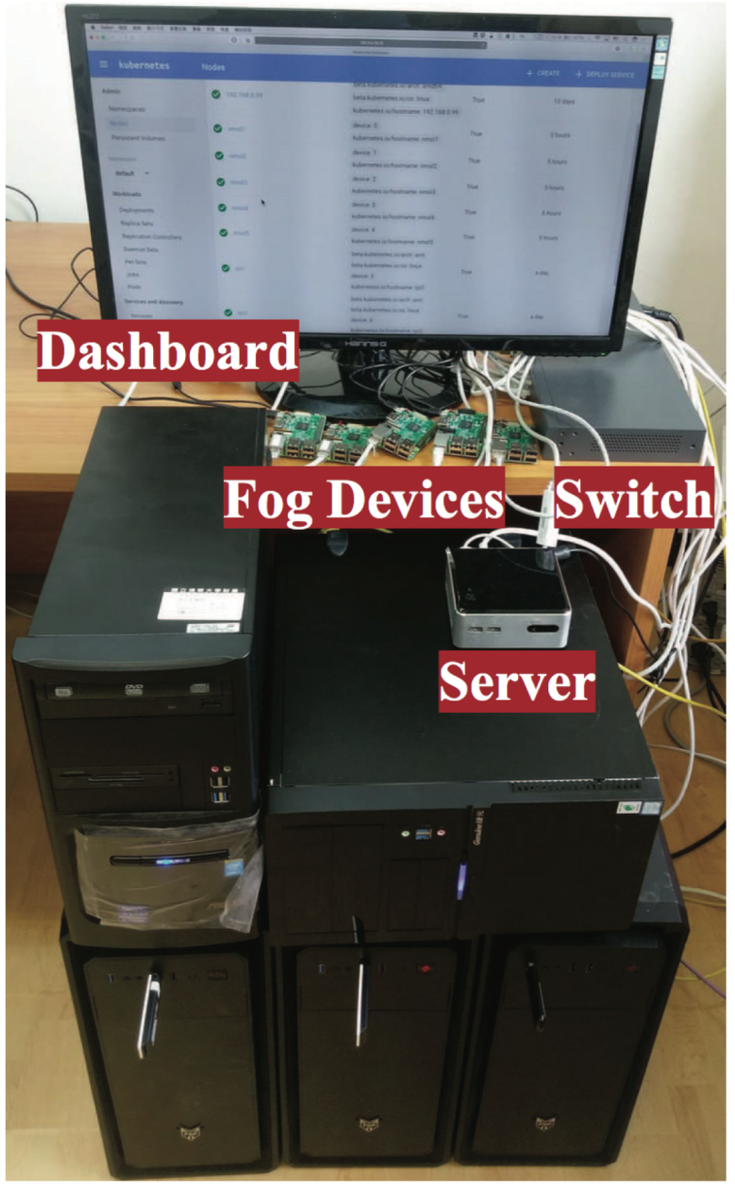
Resource Consumption Models

- Input:
  - Applications
  - Target QoS
  - Hardware specifications
- Output: required resources
  - CPU load
  - RAM usage
  - Network throughput



Testbed

- Fog devices: 5 Raspberry PI3s + 5 PCs (i7/8GB RAM)
  - Docker + Kubernetes
- Server: a tiny PC (i5/8GB RAM)
  - Kubernetes
- Switch
  - Private networks
- Analytic applications (implemented with TensorFlow)
  - Air quality monitor (QoS: 0.25 – 4 Hz)
  - Sound classifier (QoS: 6/60 – 10/60 Hz)
  - Object recognizer (QoS: 5/60 – 9/60 Hz)

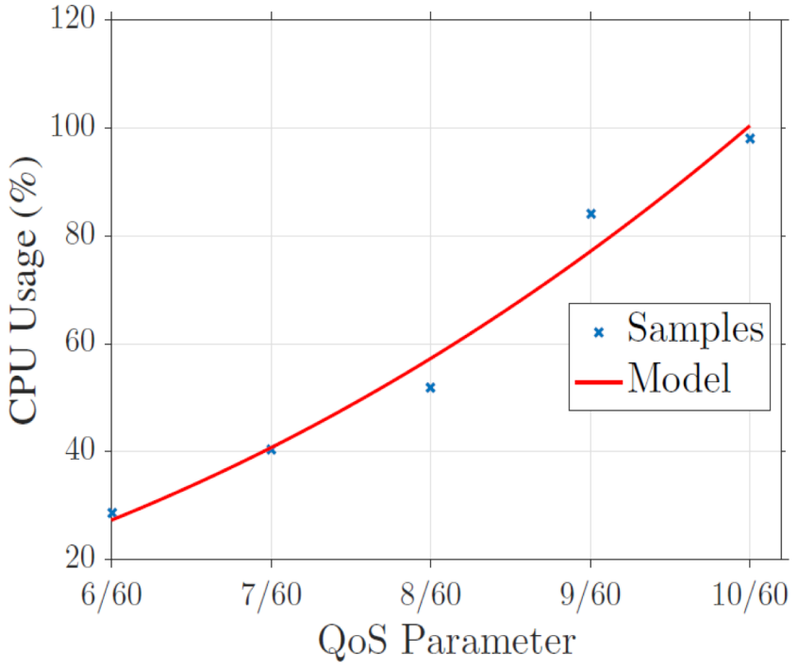


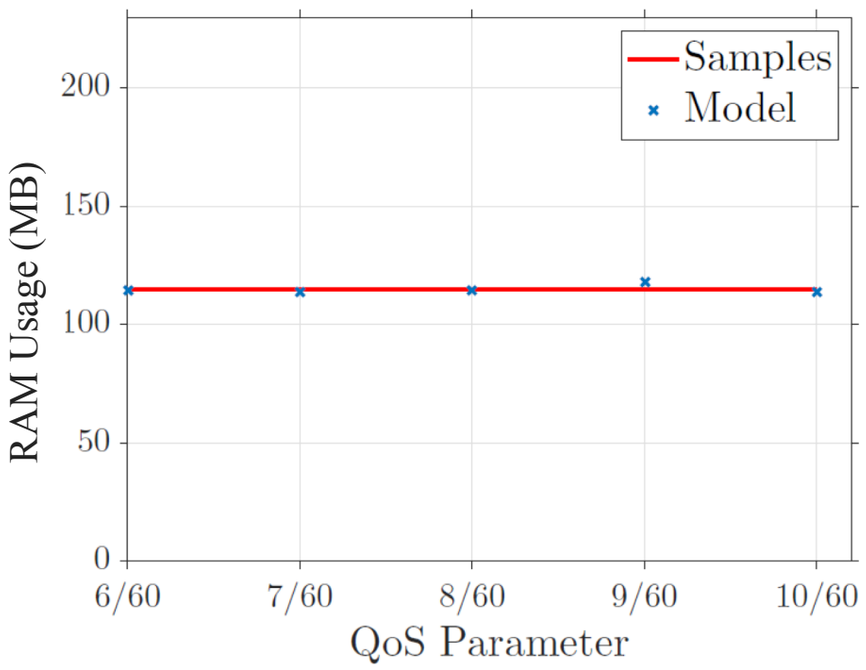
Derived Models on Raspberry PIs

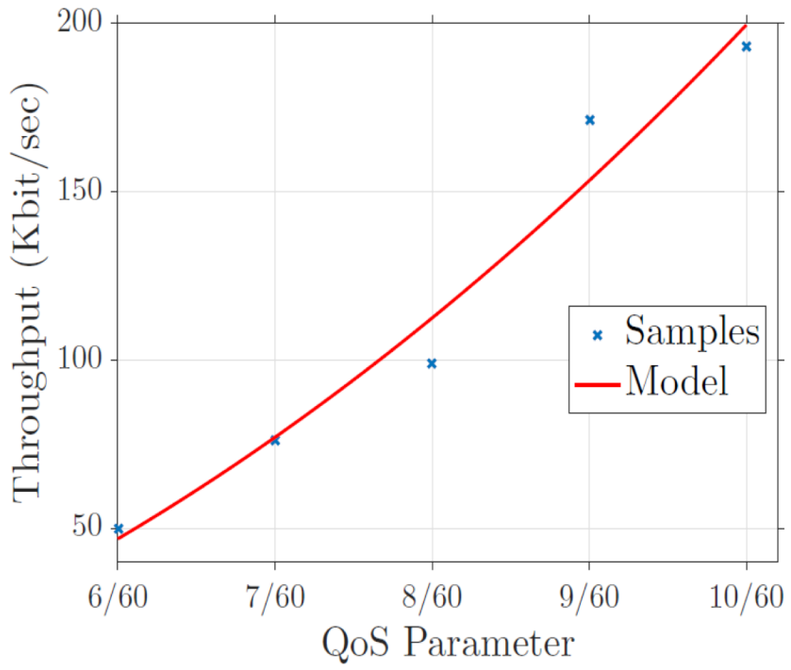
- Sample figures from the object recognizer
  - Constant model for RAM
  - Power models for CPU and network throughput

Applications	CPU Load	Network Throughput
Air Quality Monitor	0.99 (0.000023)	0.99 (0.000001)
Sound Classifier	0.85 (0.023263)	0.84 (0.069494)
Object Recognizer	0.99 (0.012540)	0.97 (0.004982)

How about new devices?

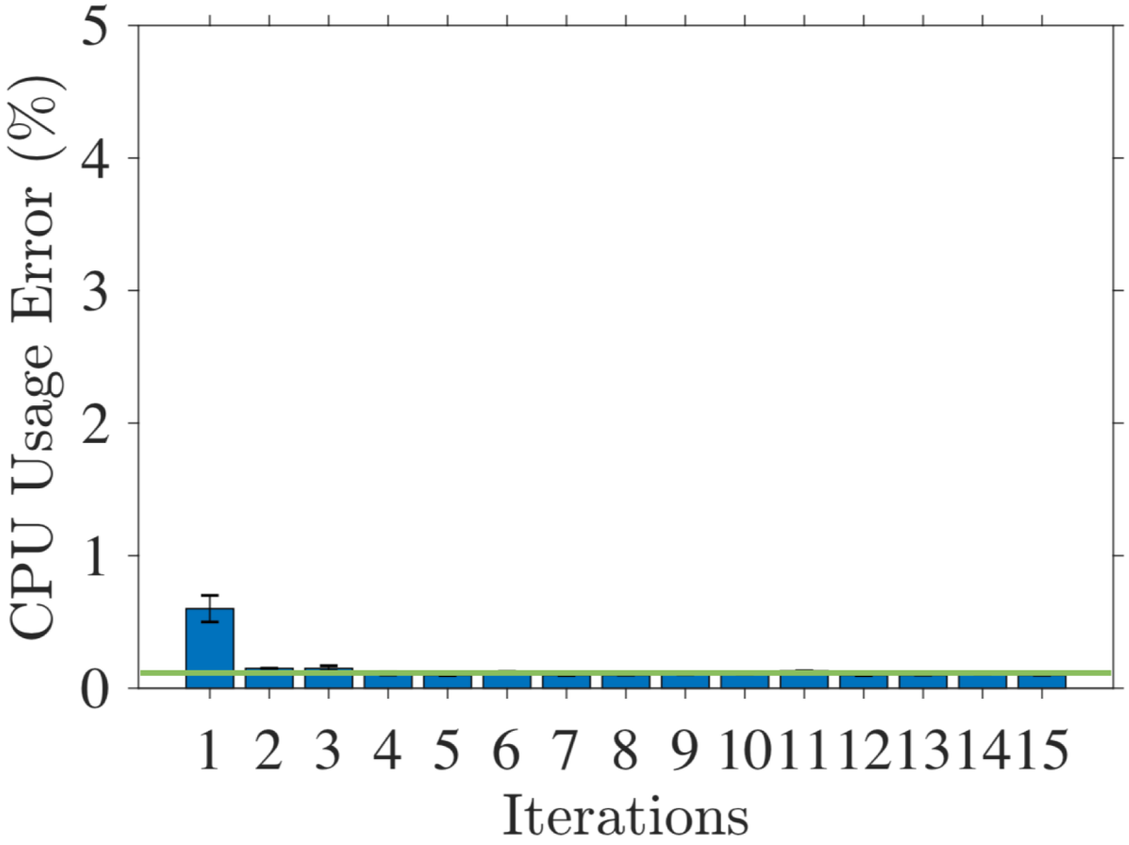


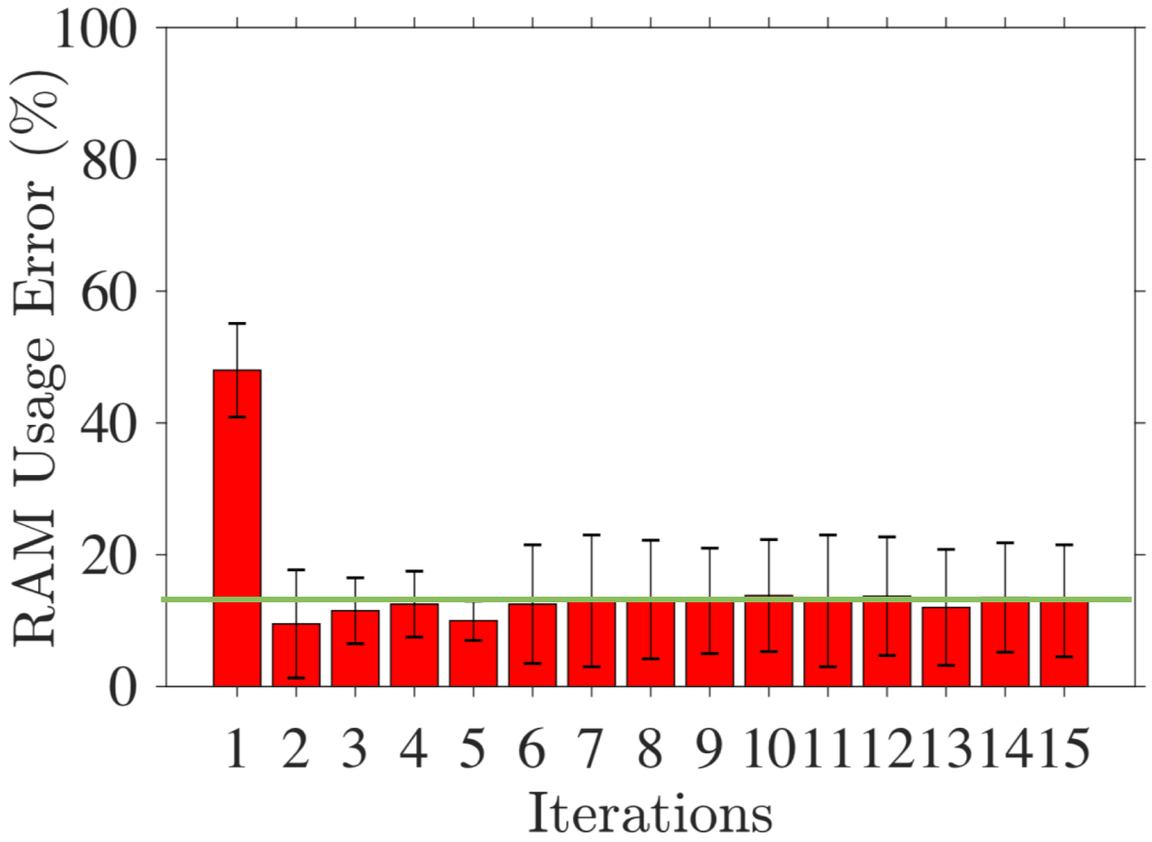




Bootstrapping Online Regression

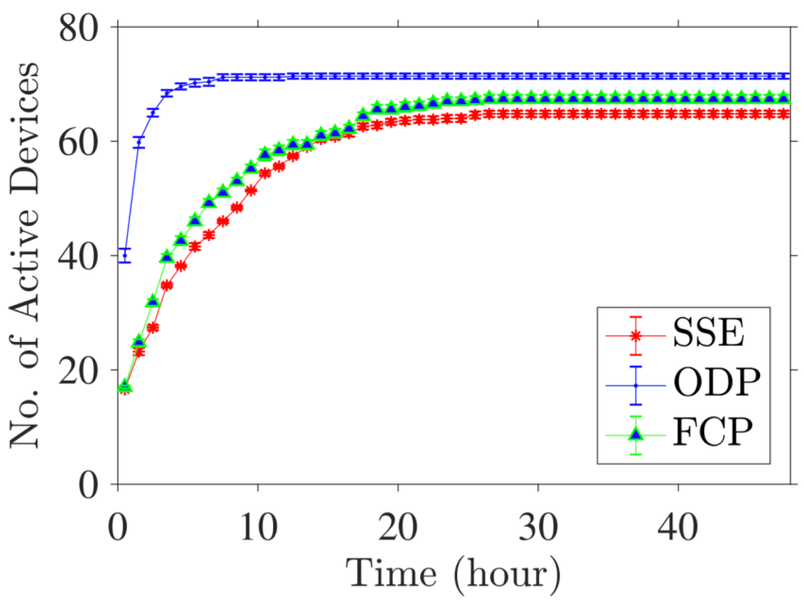
- If PC is a new type of fog device adding to our platform
  - Bootstrap from Raspberry PIs' system models
  - 15 Iterations
  - Error: CPU = 2% and RAM = 13% on average

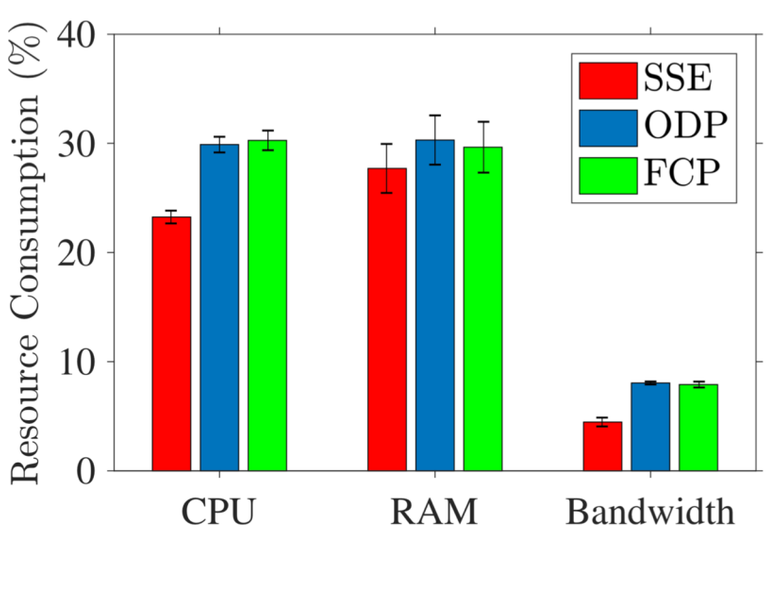


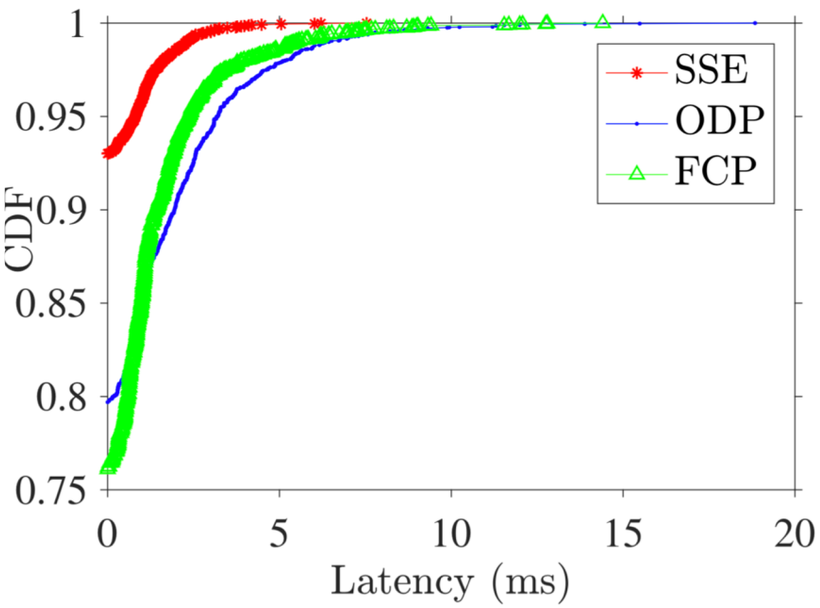


SSE is Resource Efficient and Real-Time Analytics Friendly

- Turns off 6% and 12% fog devices compared to ODP and FCP, respectively
- Use fewer CPU/RAM/bandwidth to satisfy more requests
- Lower latency compared to others







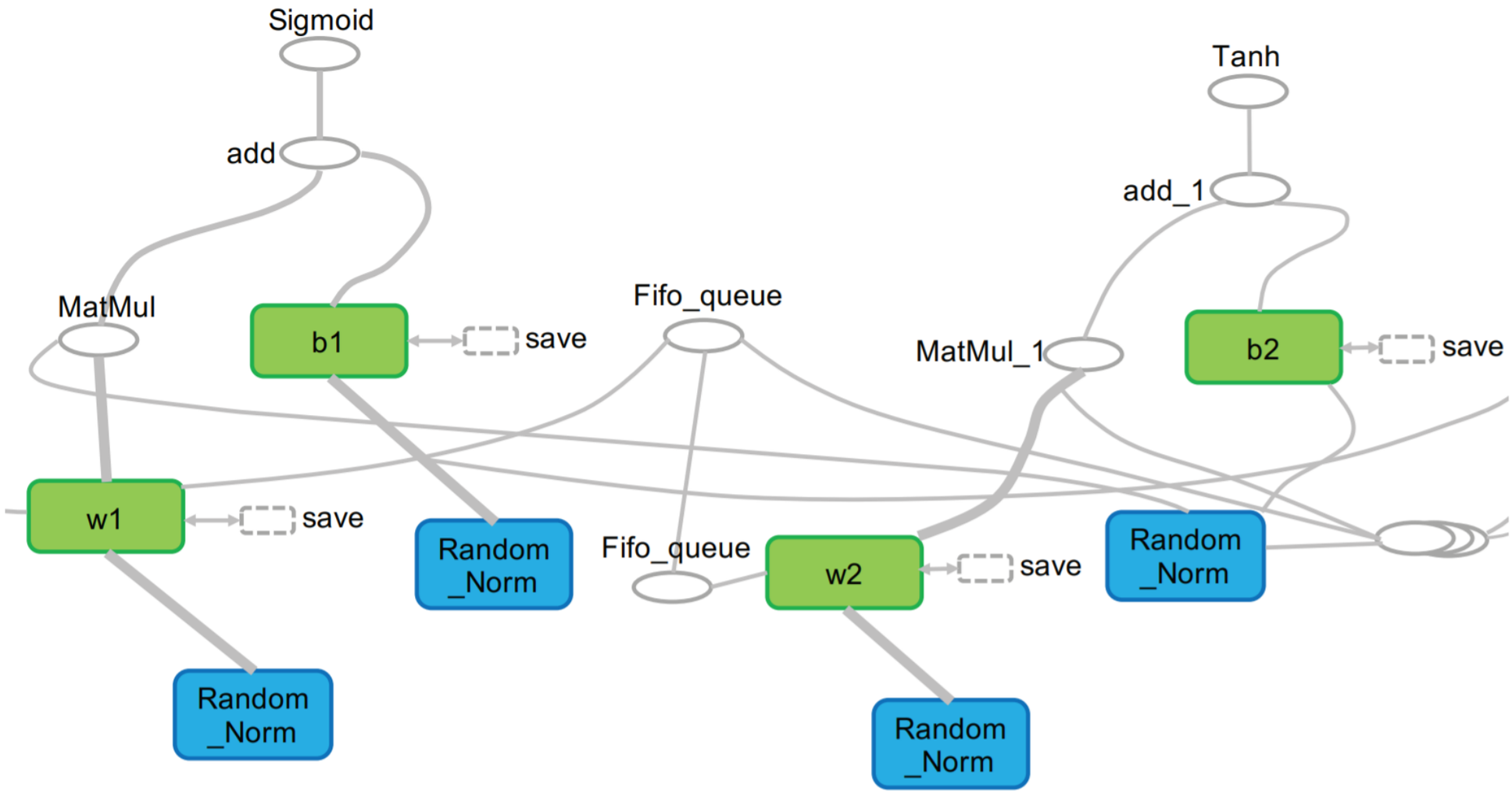
Running Time

No. Devices	10	25	50	75	100	150
SSE	1.07	2.41	5.36	8.88	9.78	13.52
ODP	2.37	22.66	101.93	382.83	>500	>500
FCP	0.07	0.10	0.21	0.27	0.31	0.46

Requests / U	500		1000		1500		2000		2500	
	APX	OPT	APX	OPT	APX	OPT	APX	OPT	APX	OPT
1	0.25	1.76	0.71	3.35	1.32	5.25	1.92	7.27	3.46	12.67
2	0.42	2.14	1.06	4.87	2.49	9.79	3.69	16.43	5.41	>500
3	1.01	2.64	2.73	7.86	4.44	203.65	6.81	>500	8.74	>500
4	1.94	3.27	4.61	>500	7.57	>500	10.16	>500	12.52	>500
5	2.63	5.33	5.76	>500	8.88	>500	11.88	>500	15.21	>500

Any Size of Operators

- A sound classifier application (4-layer NN model)
  - 348 operators



Delay-Sensitive Application Problem

Optimality Proof

Problem Formulation

Bandwidth Estimator

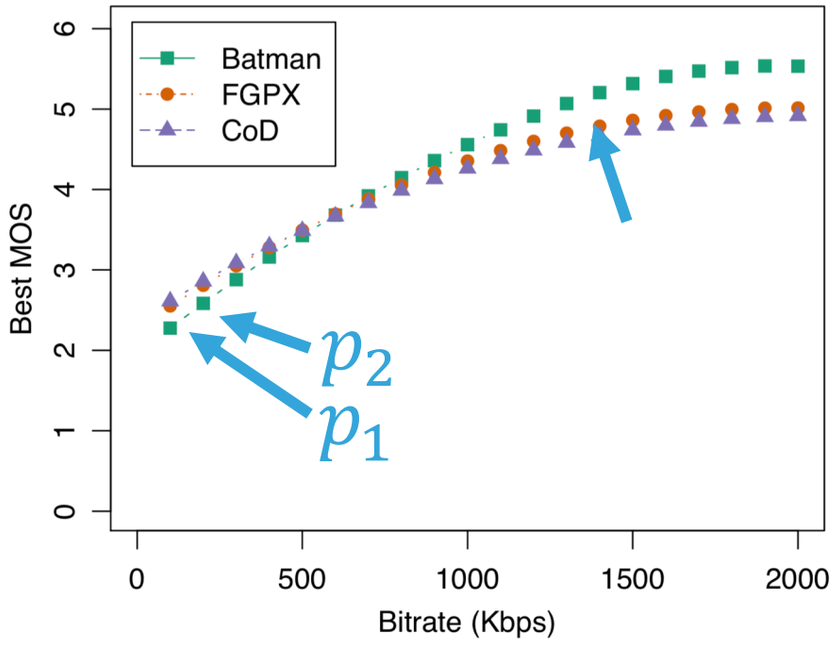
Gaming Experience Model

Performance Comparing to Baselines

Running Time of our Algorithms

Optimality of the EFF_{avg} Algorithm

- Lemma: EFF_{avg} is optimal if $m^*(g,b)$ is monotonic decreasing
- Proof:
  - EFF_{avg} finds steepest slope at every iteration: p_1, p_2, p_3 ...
    → $p_i > p_{i+1}$ → No alternative allocation
    ✓ if it is feasible solution (monotonic decreasing) → Optimal
  - Monotonic decreasing: double deviation of $m^*(g,b)$ (negative)
    → $m^*(g,b)'' = 2\alpha_{g,4} - \alpha_{g,5}^2 / 2\alpha_{g,3} < 0$



Our models satisfy this equation

p_n: quality improvement of adding one unit

Optimality of the EFF_{mm} Algorithm

- Lemma: EFF_{mm} is optimal if it allocates w to gamer u who has the worst MOS
- Proof:
  - EFF_{mm} allocates w (1kbps) to user u who has the worst MOS (l)
    → MOS improvement = q
  - Consider alternative u'
    → MOS improvement = 0 (the worst MOS is still l)
    → Need to allocate another w to u to achieve the same MOS improvement
  - Hence, we cannot find an alternative leading to a better solution
    → EFF_{mm} is optimal

Bitrate Adaptation Problem Formulation

Decision Variables

$$maximize \sum_{u=1}^{U} m_{p_u,x_u,y_u}$$

Objective 1:
Gaming Experience Maximization

$$s.t.: \sum_{u=1}^{U} y_u \leq R;$$

Bandwidth Constraint

$$\check{k}_{u'} \leq y_{u'} \leq \hat{k}_{u'}, \ \forall \ 1 \leq u' \leq U;$$

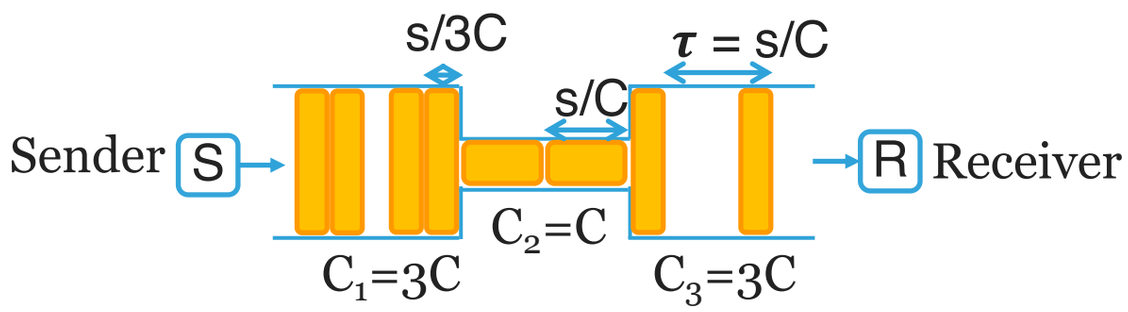
$$1 \leq x_u \leq F, 1 \leq y_u \leq B, \ \forall \ 1 \leq u \leq U.$$

$$maximize \min_{u=1}^{U} m_{p_u,x_u,y_u}$$

Objective 2:
Quality Fairness

Bandwidth Estimator

- WBest [1]
  - Estimate bandwidth capacity by sending probing packets



capacity $C = \frac{s}{\tau}$

packet dispersion time

- Our bandwidth estimator
  - Leverage existing video packets as probing packets
    → Avoid additional network overhead

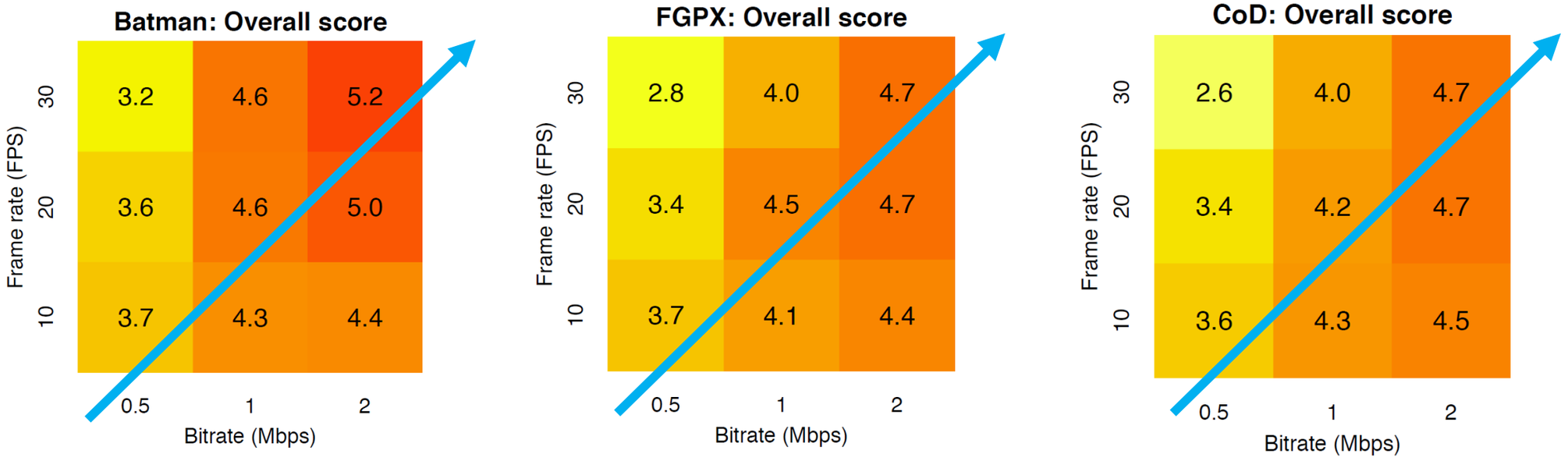
[1] M. Li, M. Claypool, and R. Kinicki, "WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks," *IEEE Conference on Local Computer Networks, 2008.*

The Bandwidth Estimator is Accurate

- Send W packets and select the median value as estimated bandwidth capacity
- W = {100, 200, 300, 400, 500}
  - W = 100 → accuracy < 50%
  - W = 200 ~ 500 → 80% < accuracy < 85%

Gaming Experience User Study

- Gaming experience → Mean Opinion Scores (MoS): 1~7
- 101 Subjects
- 3 type of games = {Batman, FGPX, CoD}
- Bitrate = {0.5, 1, 2} Mbps
- Frame rate = {10, 20 , 30} fps
- Game sessions = 2 ~ 4 minutes



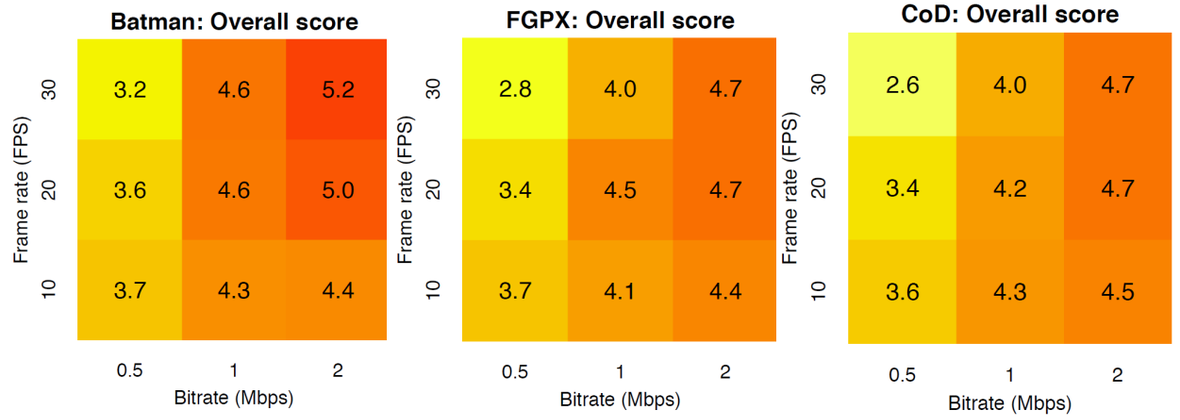
[Hong et al. TCSVT'15, Credits: collaborators @ Academia Sinica]

Codec Reconfiguration [Hong et al. TCSVT'15, Credits: collaborators @ NCTU]

- Migration from *ffmpeg* to *live555*
  - ffmpeg does not have an interface for dynamically reconfiguring video codecs
  - live555 offers a more comprehensive RTCP implementation, which allows us to measure and collect the network flow statistics

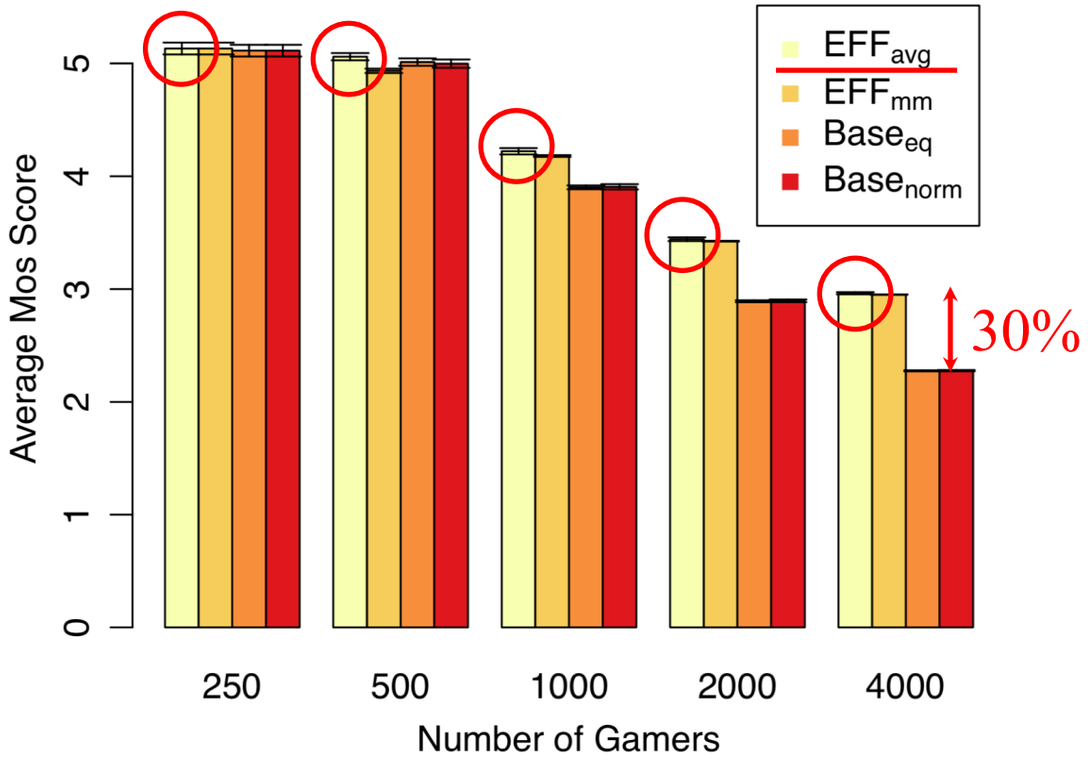
Our Algorithms Outperform Baselines: Setup

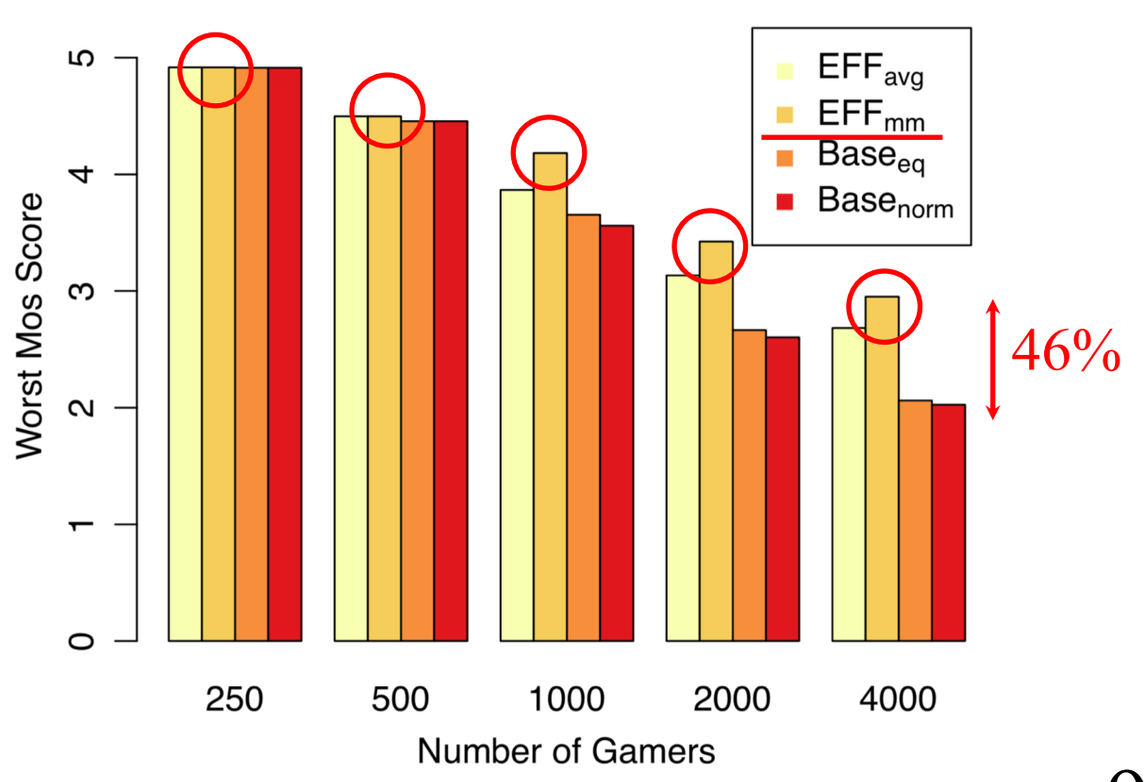
- Optimal algorithms: CPLEX
  - OPT_{avg}
  - OPT_{mm}
- Baseline algorithms
  - $Base_{eq}$: equally allocate
  - $Base_{norm}$: proportionally (to average MOS scores) allocate
- Bottleneck link capacity R = 760 Mbps (PlanetLab)



Our Algorithms Outperform Baselines

- EFF_{avg} results in the best average MOS scores
  - Outperforms baselines by up to 30%
- EFF_{mm} results in the best worst MOS scores
  - Outperforms baselines by up to 46%





Our Algorithms run in Realtime

- 4000 gamers ~ 1 second

Table 5.5: Running Time in Seconds

# of Gamers	$\mathbf{EFF}_{avg}$		$\mathbf{EFF}_{mm}$	
	Mean	Max	Mean	Max
500	0.181	0.183	0.179	0.184
1000	0.296	0.299	0.287	0.290
2000	0.523	0.531	0.520	0.533
4000	1.000	1.104	1.060	1.066
8000	1.677	1.681	1.654	1.661

Delay-Insensitive Application Problem

Dynamic programming problem

CDRR: Practical concerns

Block diagram

Energy consumption measurement

Dynamic Programming

$$d(i,j) = \max\{d(i-1,j), u_i + d(i-1, j-w_i))\}$$

Weight restriction j (j=1,2, … , W)

Items picked (i=1, 2, 3, 4, …)	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	1	1	1	1	1	1
	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}
2	1	1	1	1	1	1	7	8	8	8	8
	{1}	{1}	{1}	{1}	{1}	{1}	{2}	{2,1}	{2,1}	{2,1}	{2,1}
3	1	1	1	4	5	5		8	8	8	11
	{1}	{1}	{1}	{3}	{3,1}	{3,1}	{2}	{2,1}	{2,1}	{2,1}	{3,2}
4	1	2	3	4	5	6	7	8	9	10	11
	{1}	{4}	{4,1}	{3}	{3,1}	{4,3}	{4,3,1}	{2,1}	{4,2}	{4,2,1}	{3,2}

Target

Complexity: O(IW)

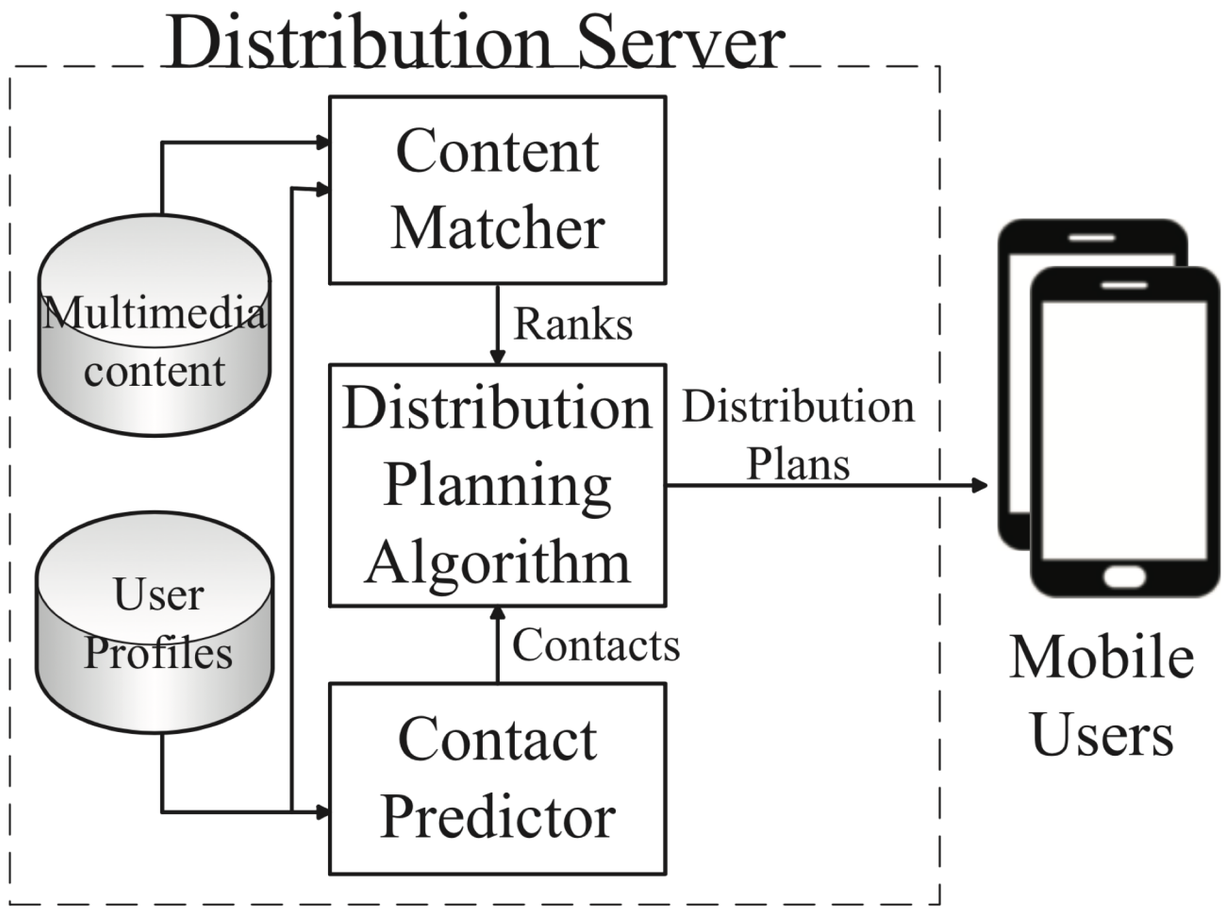
Practical Concerns

- Planed contents are downloaded or plans are not received
  - Determining the downloading order
  - Next outstanding representation of each content

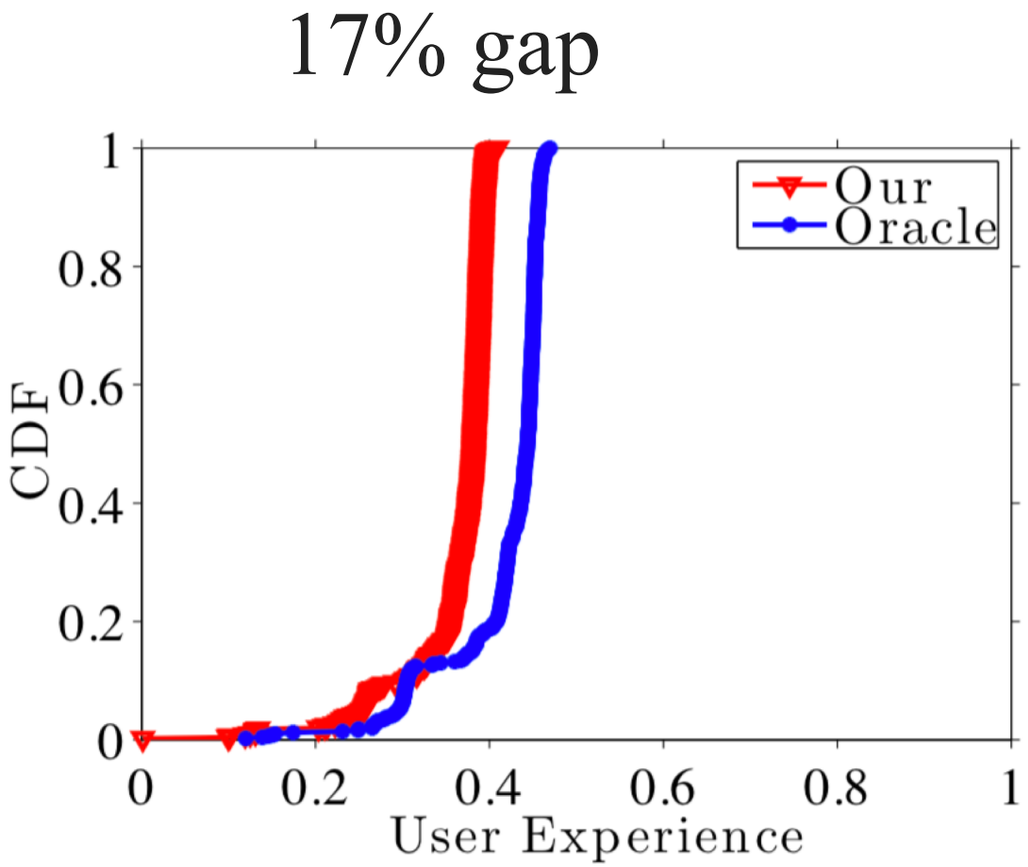
    (sorted by viewing $\frac{\text{probability}}{\text{content size}}$)
- Video size is too large
  - Segmenting video layers

Block Diagram

- Content Matcher
  - Topia Term Extractor
  - Google Bayesian
- Contact Predictor
  - Frequency-based approach



Prediction Accuracy



DP vs CDRR

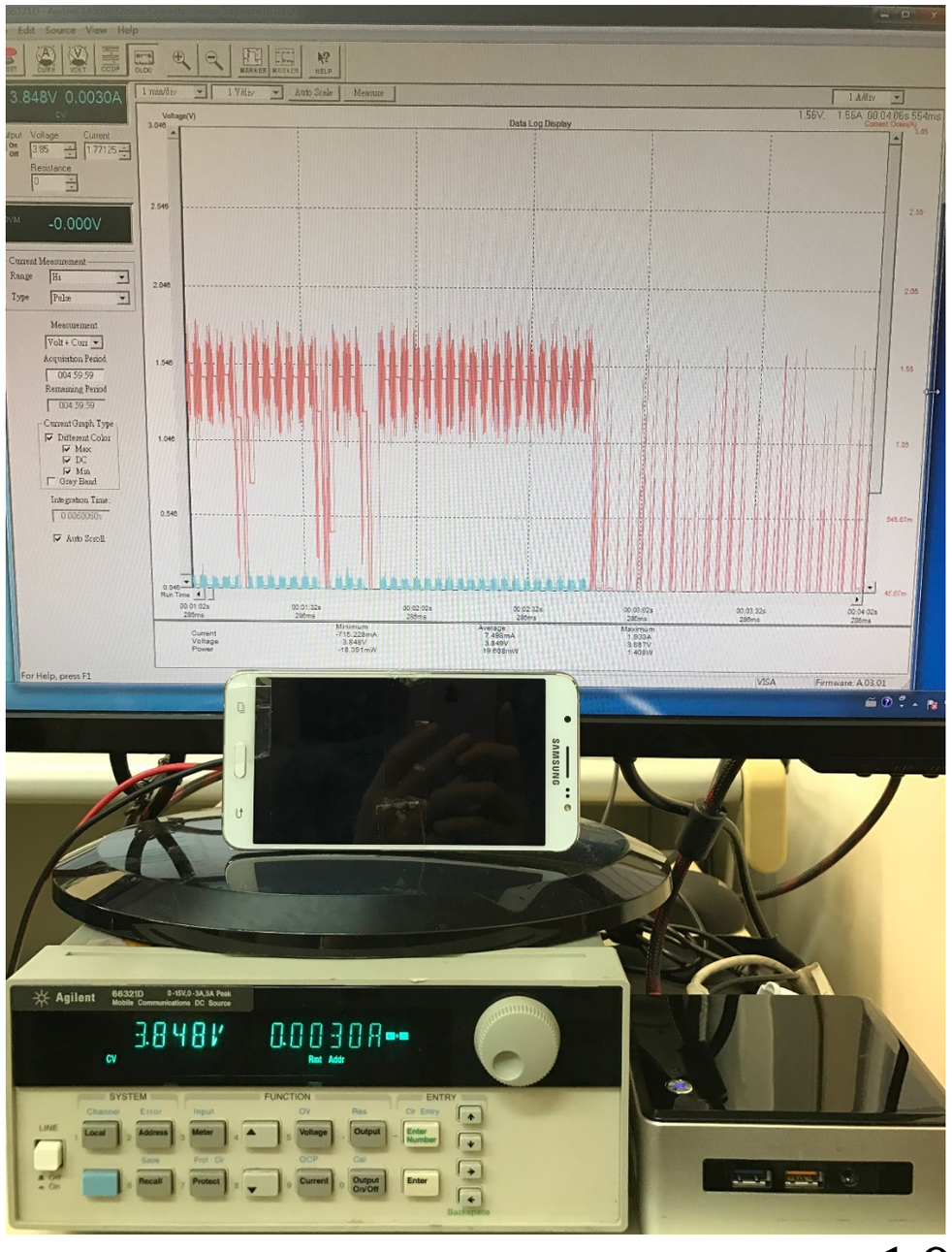
No. Content	Running Time (sec)			
	DP		CDRR	
	Mean	Max	Mean	Max
1	0.08	0.08	0.05	0.05
2	0.11	0.11	0.06	0.07
3	0.19	0.19	0.63	0.66
4	0.6	0.61	0.07	0.07
5	1.8	1.8	0.07	0.08
6	110.2	111.2	0.08	0.08
7	269.4	457.2	0.08	0.09

No. Content	Total User Experience		
	DP	CDRR	Perf. Gap
1	0.43	0.42	3%
2	0.36	0.34	6%
3	0.33	0.31	6%
4	0.31	0.29	6%
5	0.27	0.25	7%
6	0.29	0.27	7%
7	0.29	0.27	7%

How to Know the Energy Consumptions?

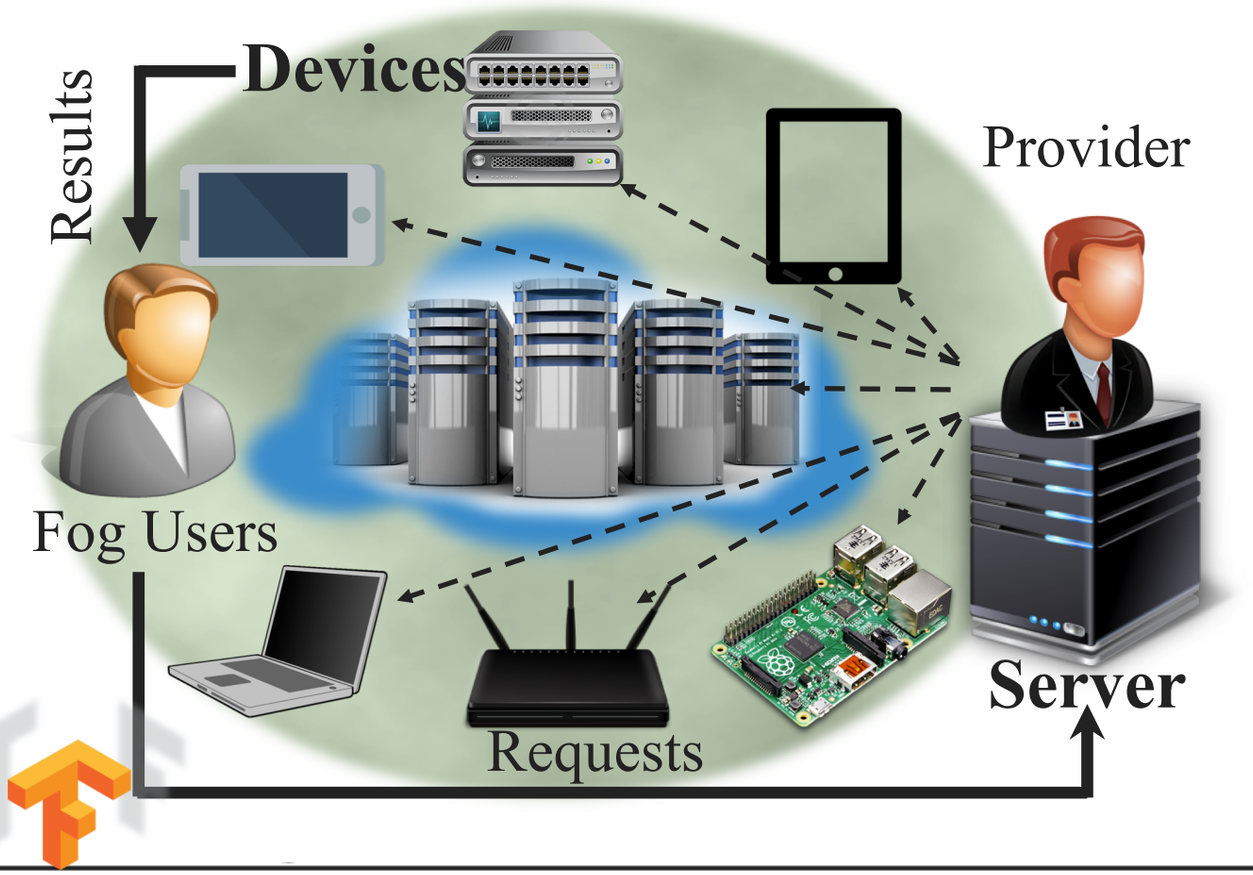
- Power meter: Agilent 66321D
- Read trace:
  - Downloaded news
  - Downloaded plan
  - Uploaded user profiles

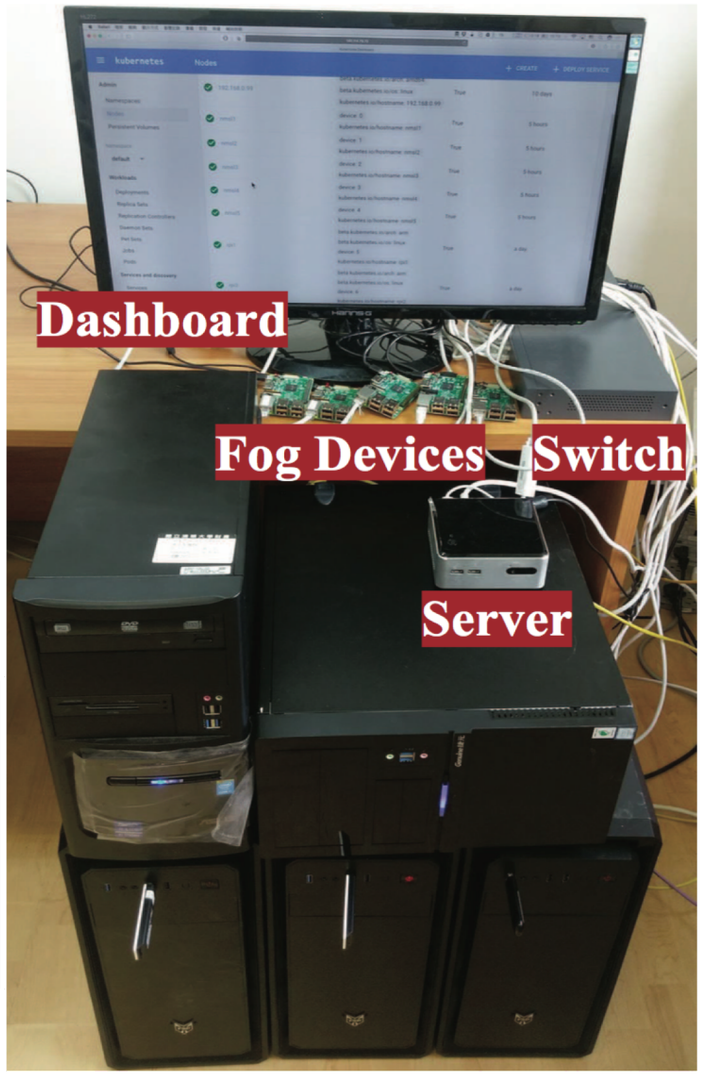
- 1-day distribution plan only consumes up to 153 J, which is 0.3%



Fog Testbed and Performance Measurement

Testbed





TensorFlow	Application	
Global Optimizer		
Virtualization/ Containerization	Management	Application Specific Optimizer

Virtualization & Resource Reservation

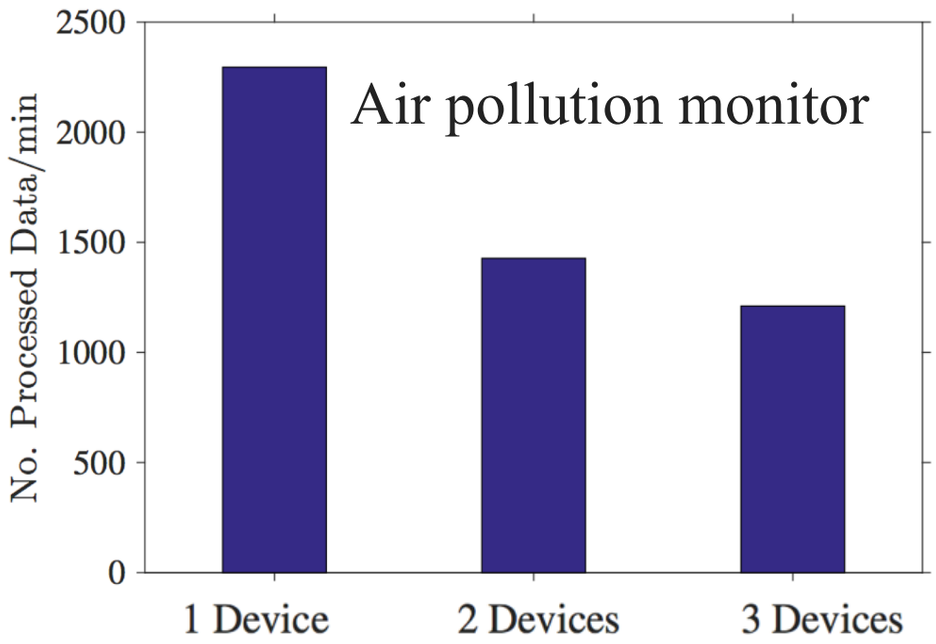
docker

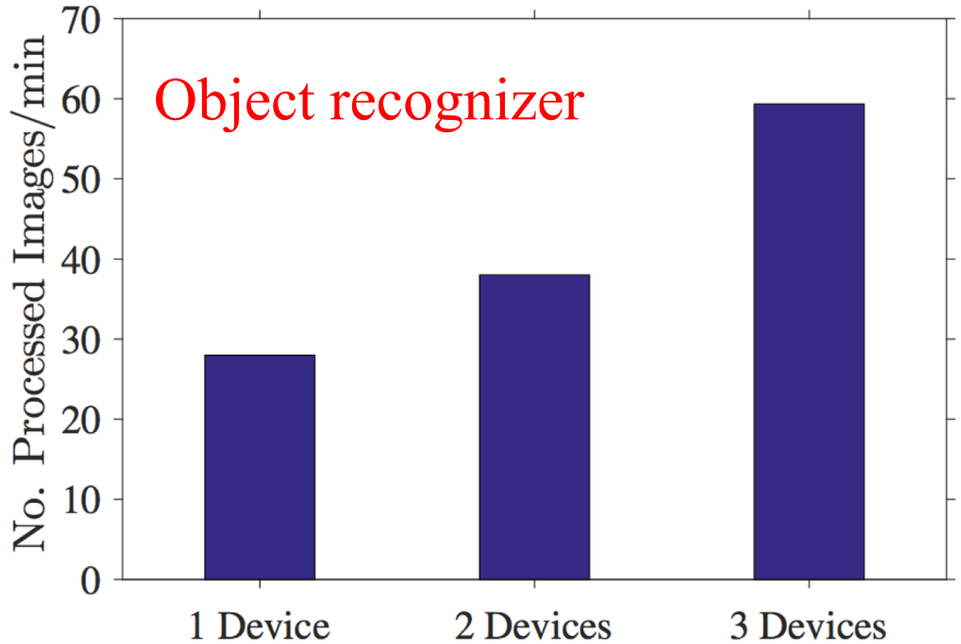
kubernetes

Management & Deployment

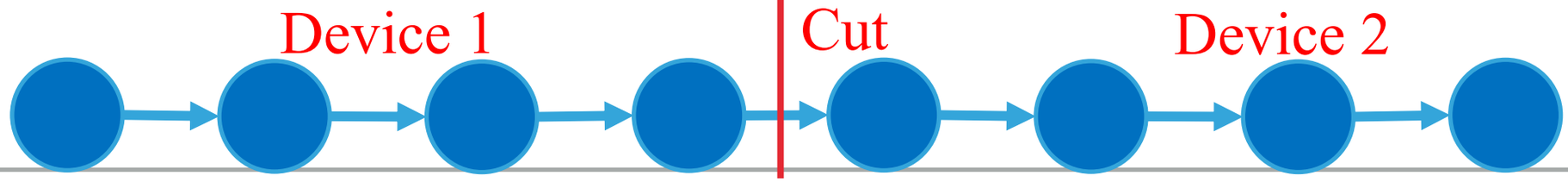
Distributed Analytics Result in Better Performance for Complex Analytics

- Air pollution monitor: too simple to benefit from distributed analytics
- Object recognizer:
  - One more device gives 35.5% and 54.1% improvements



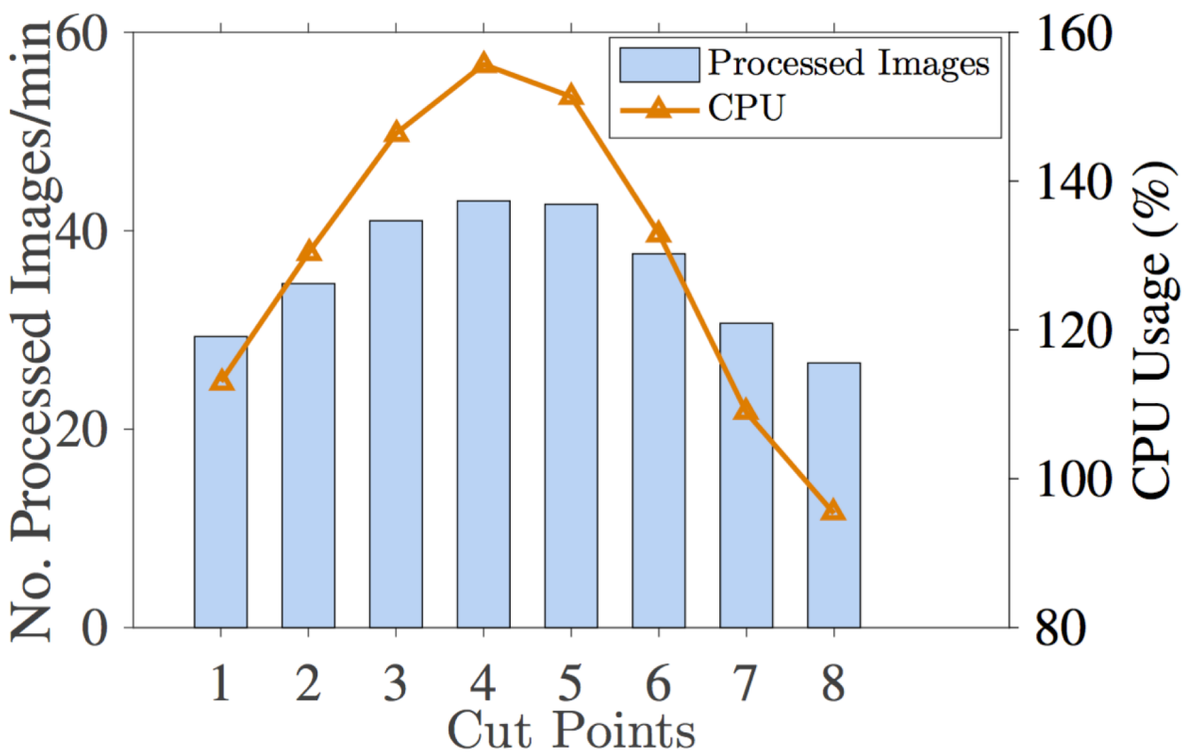


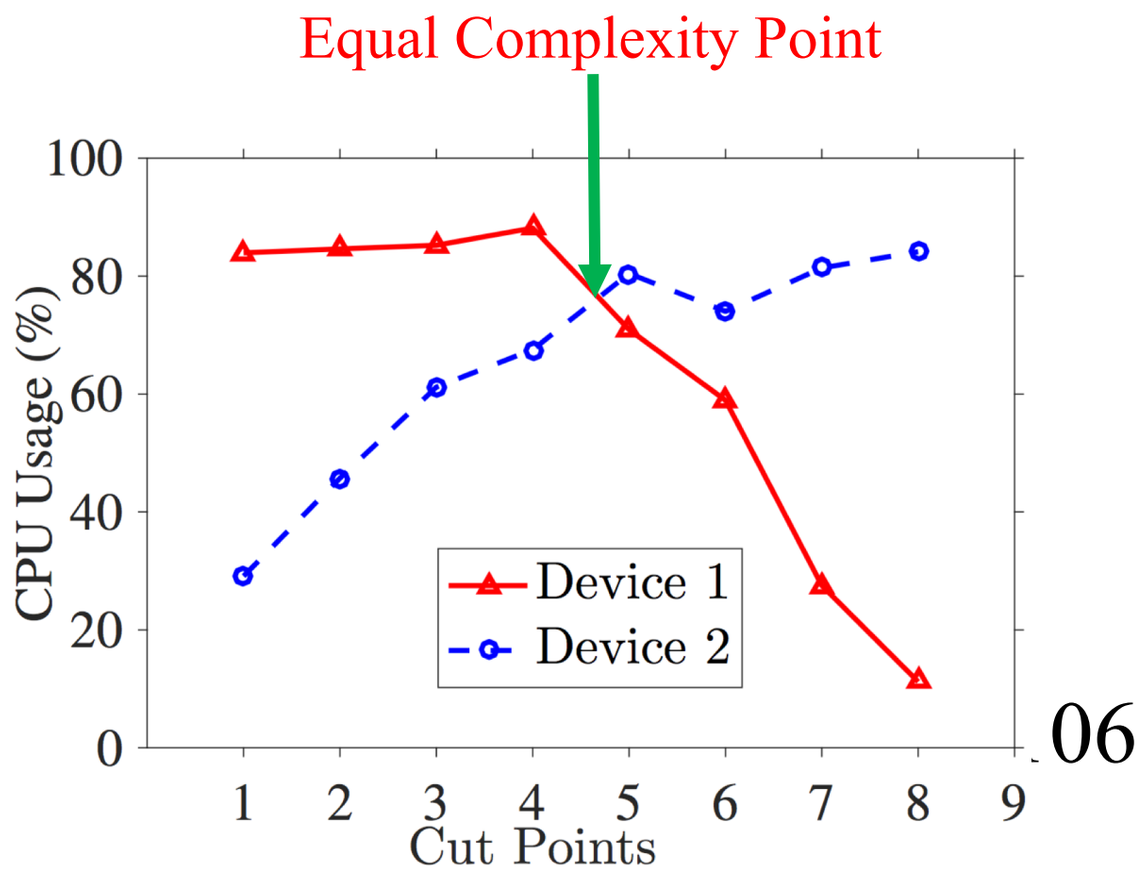
We only consider the object recognizer in the following results

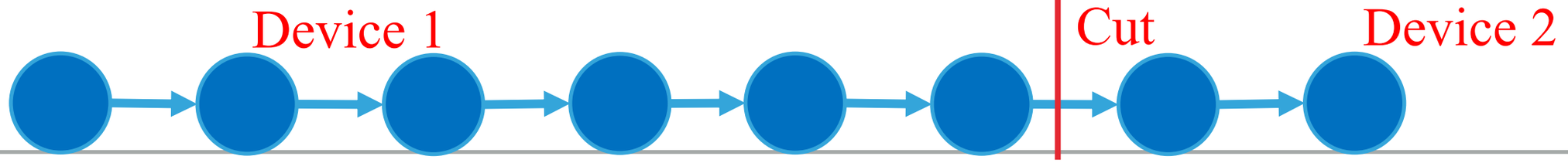


Cut on Equal Complexity Point Results in Better Performance

- Setup: run object recognizer on two fog devices (different cut points)
- Cut point 4&5 result in the most no. processed images
- Two fog devices consume the same CPU resources on cut point 4&5

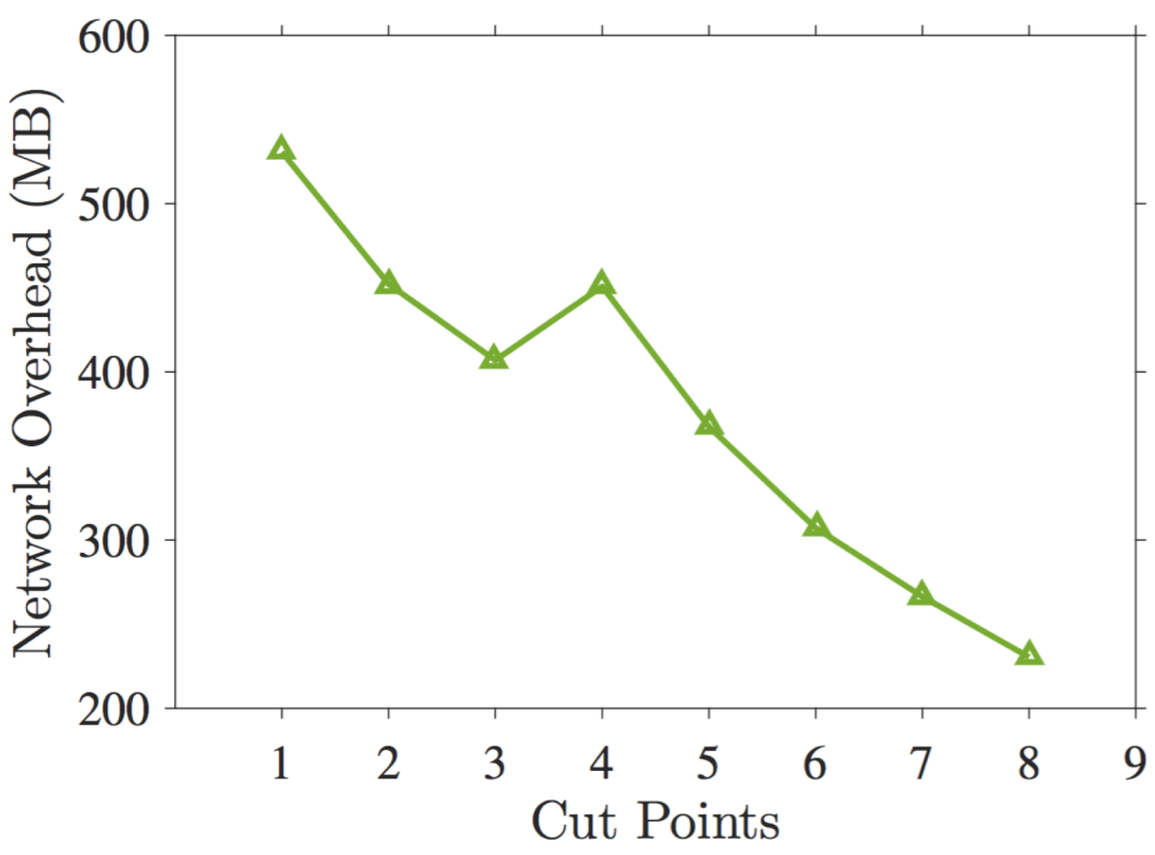






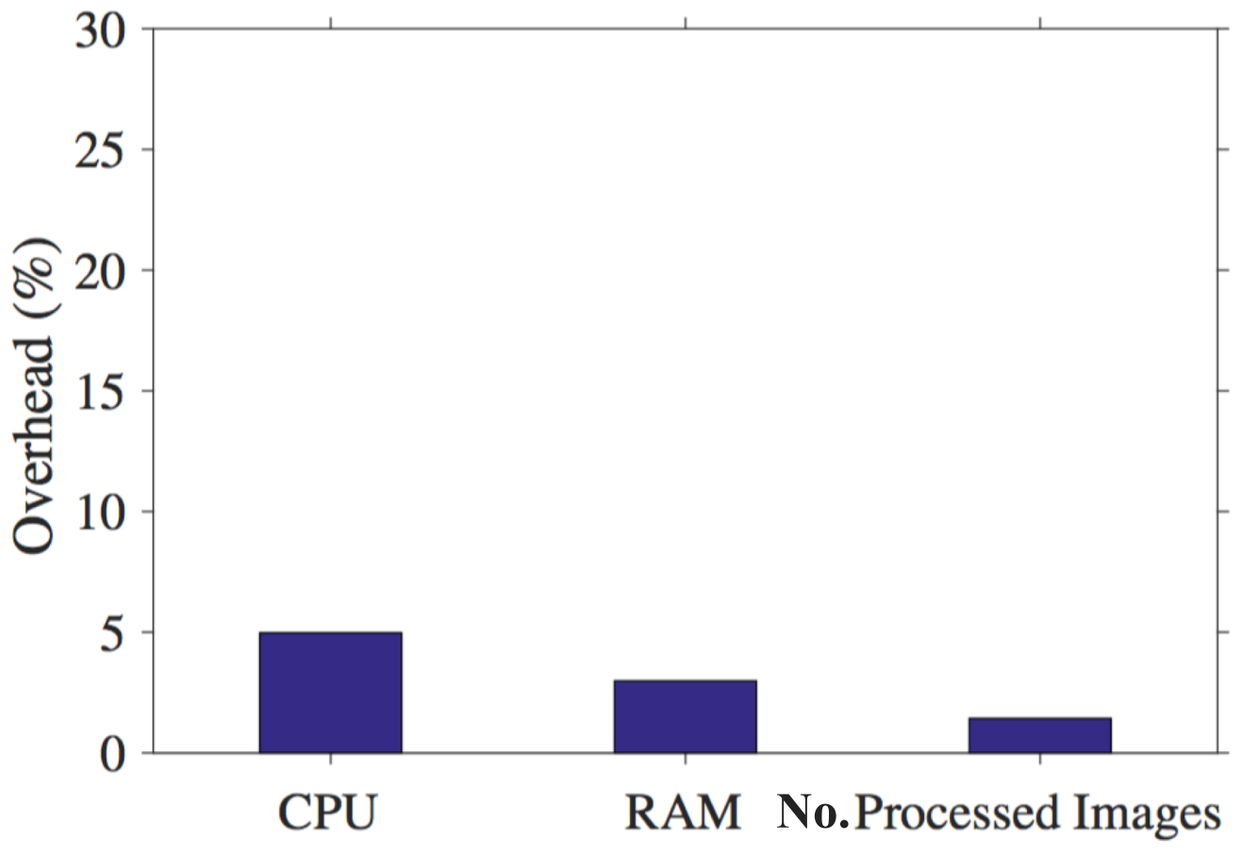
Cut Points Affect Network Overhead

- Put more computations on device 1 can reduce network overhead



Low Virtualization Overhead

- Setup: with and without Docker
- Overhead from Docker
  - Less than 5%



Low Communication Overhead

- Setup: without Docker and distributed speedup
- Overhead from distributed computing using TensorFlow
  - 10%