A Multi-Tenant System for Deploying Deep Neural Networks in a Thing-to-Cloud Continuum

Chia-Ying Hsieh

cyinghsieh@gmail.com

Advisor: Cheng-Hsin Hsu

Networking and Multimedia Systems Lab, CS,
National Tsing Hua University

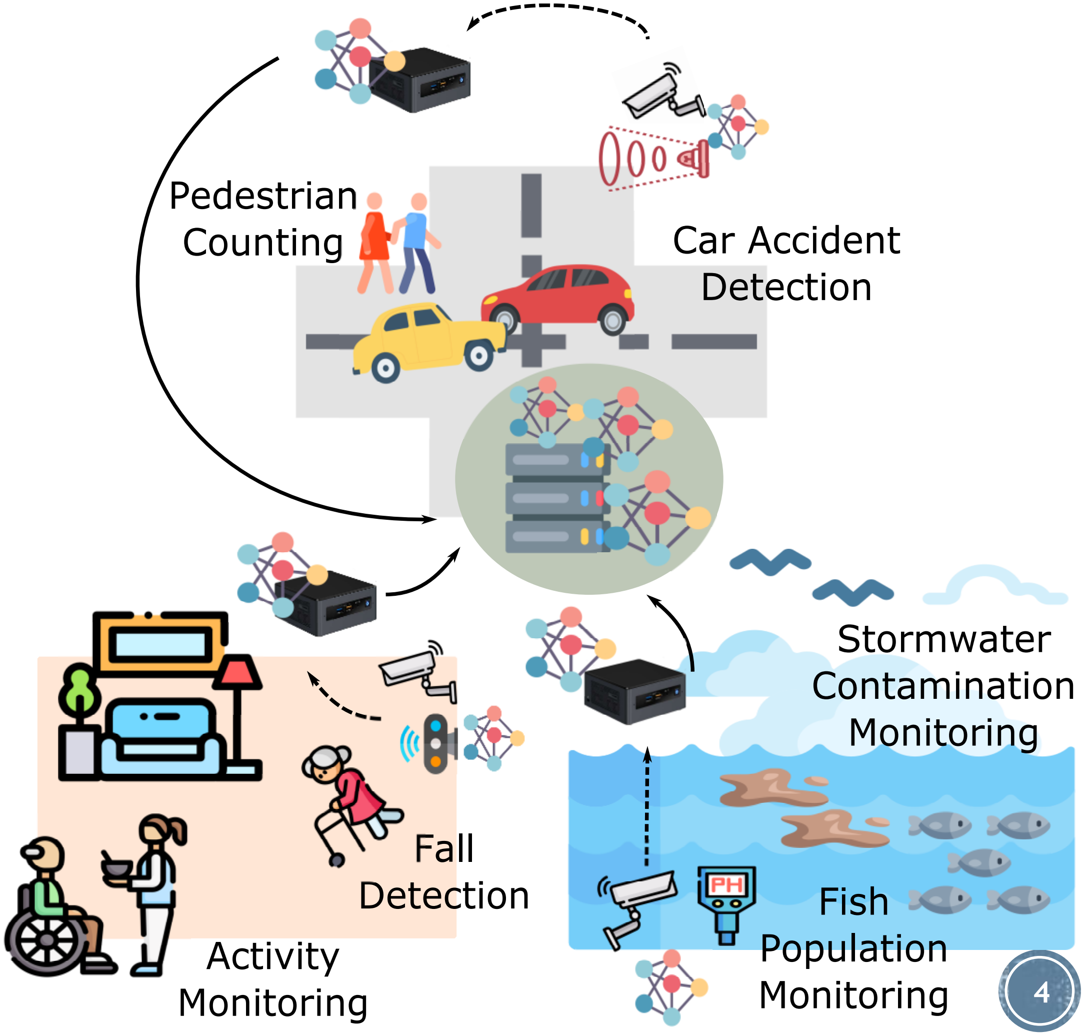
Outline

- Motivations
- Challenges & Goal
- Related Work
- System Overview
- Planning Phase
- Operation Phase
- Implementations
- Evaluations
- Conclusion & Future Work

Motivations

Numbers of IoT Sensors are Growing Rapidly

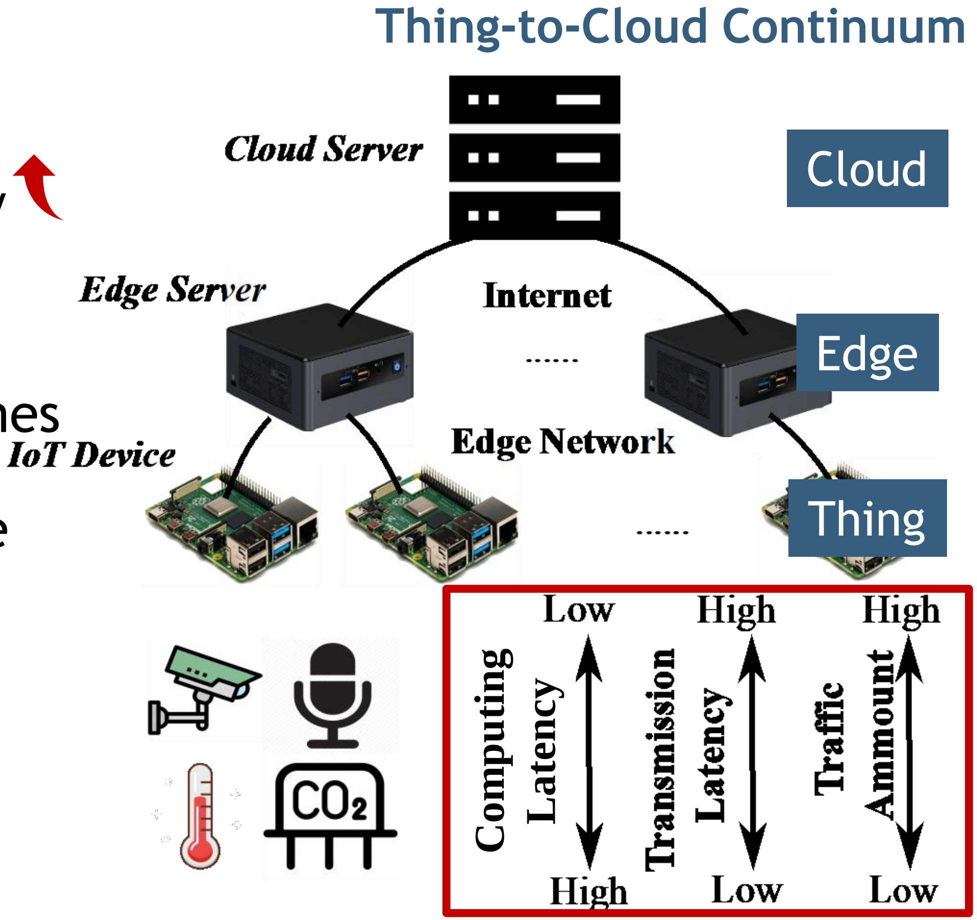
- “By 2025, there will likely be more than 27 billion IoT connections.” [1]
- The data generated from the sensors need to be analyzed to export the information
  - IoT analytics are needed
  - DNNs are popular for analyzing data



[1] Satyajit Sinha. (2021) State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 billion globally, cellular IoT now surpassing 2 billion. [Online]. Available: (https://iot-analytics.com/number-connected-iot-devices/)

DNN Deployment Problem

- Deploying DNN
  - On IoT device: computing latency ↑
  - On cloud server: transmission latency ↑
- DNN deployment decision becomes a key research problem to guarantee the Quality-of-Service (QoS) of IoT analytics
  - Trade-offs between analytics accuracy, computation and transmission latencies, network overhead, etc.



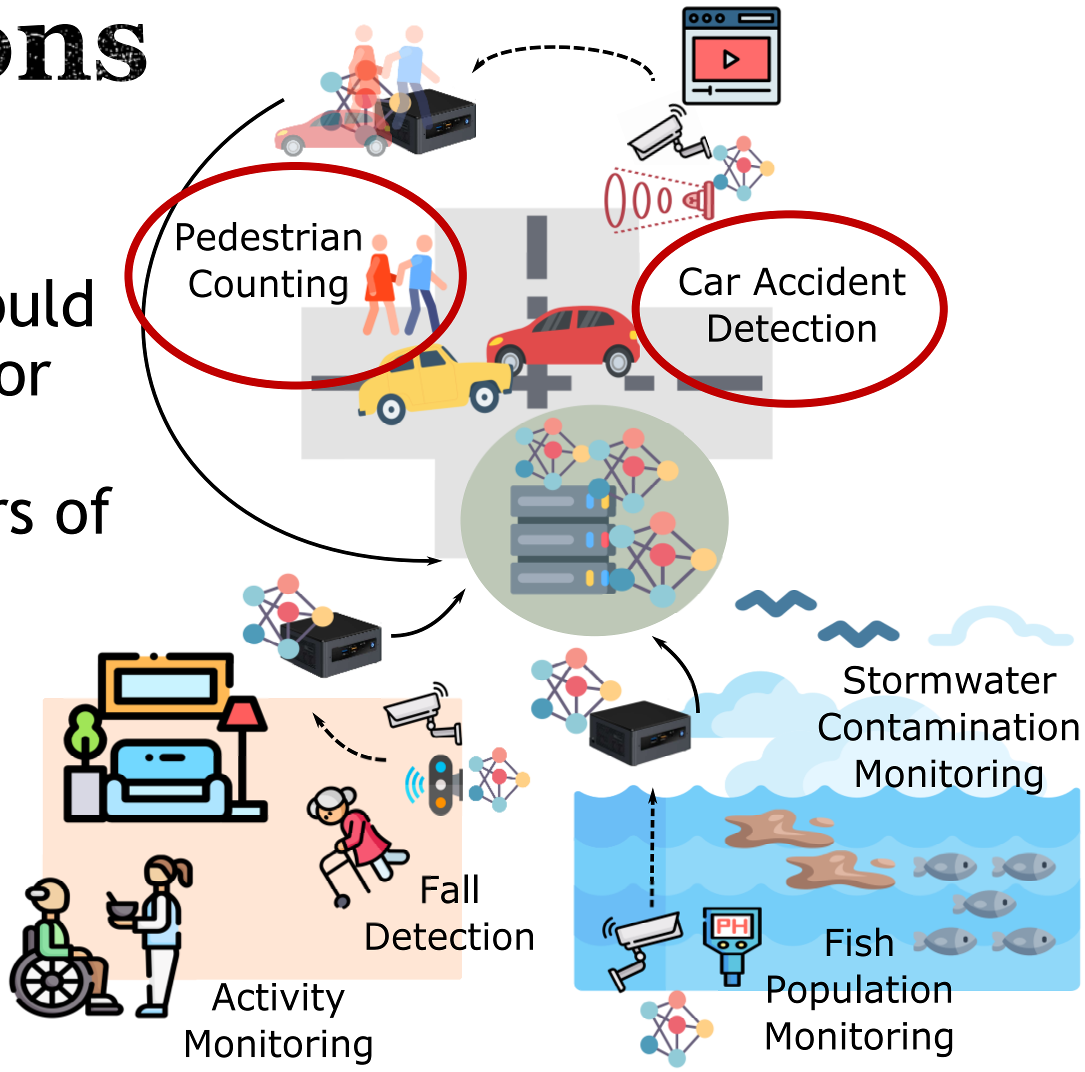
Goals and Challenges

Goals and Challenges

- Goals
  - Maximize # served requests
  - Dynamically choose, deploy, monitor, and control IoT analytics under resource constraints
- Challenges
  - Limited resources
  - Diverse QoS requirements
  - Different request arrival patterns
  - Dynamic environments

Observations

Multiple analytics could share the same sensor data or even some common prefix layers of their DNNs



Proposed Thing-to-Cloud (T2C) System Features

- Limited resources

Multi-task [ML’97]

- Diverse QoS requirements

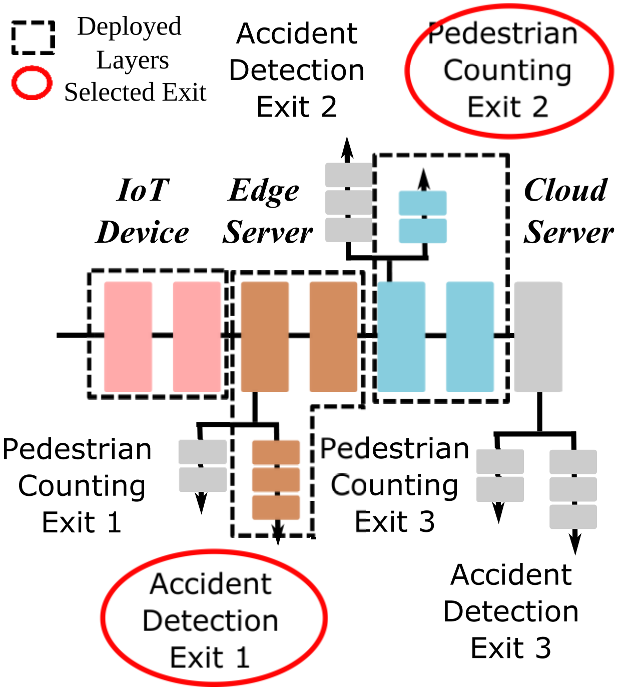
Early exit [ICPR’16]

- Different request arrival patterns

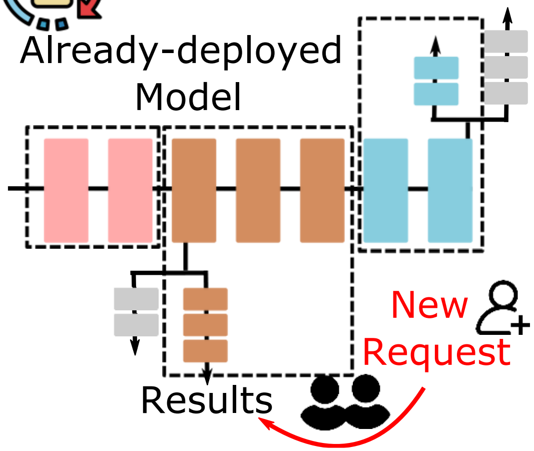
Hitchhiking

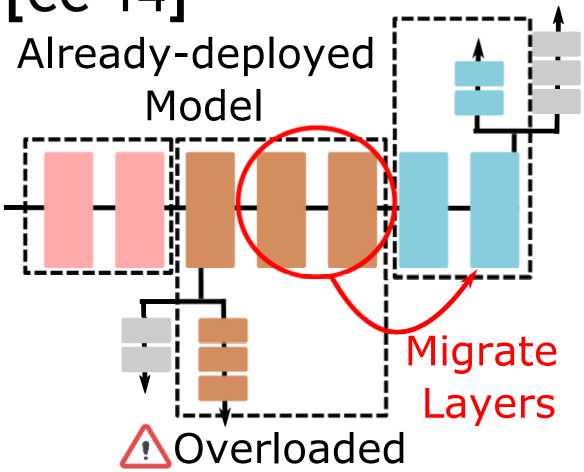
- Dynamic environments

Reconfiguration [CC’14]



Deployed Layers
Selected Exit
Accident Detection Exit 2
Pedestrian Counting Exit 2
IoT Device
Edge Server
Cloud Server
Pedestrian Counting Exit 1
Pedestrian Counting Exit 3
Accident Detection Exit 1
Accident Detection Exit 3





[ML’97] R. Caruana. Multitask learning. Springer Machine Learning, 28(1):41-75, July 1997.

[ICPR’16] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proc. of IEEE International Conference on Pattern Recognition (ICPR’16)*, pages 2464–2469, Cancun, Mexico, December 2016.

[CC’14] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri. Run time application repartitioning in dynamic mobile cloud environments. IEEE Transactions on Cloud Computing, 4(3):336-348, September 2014.

Related Work

DNN-based IoT Analytics Deployment

Name	Multi-tenant	Distributed Inference	Early Exit	Multi-task	Reconfiguration
[MobiSys'17]	✓				
[SIGARCH'17]		✓			
[WC'19]		✓	✓		
[ATC'18]				✓	
[SEC'20]		✓	✓	✓	
[IC2E'21]		✓			✓
Ours	✓	✓	✓	✓	✓

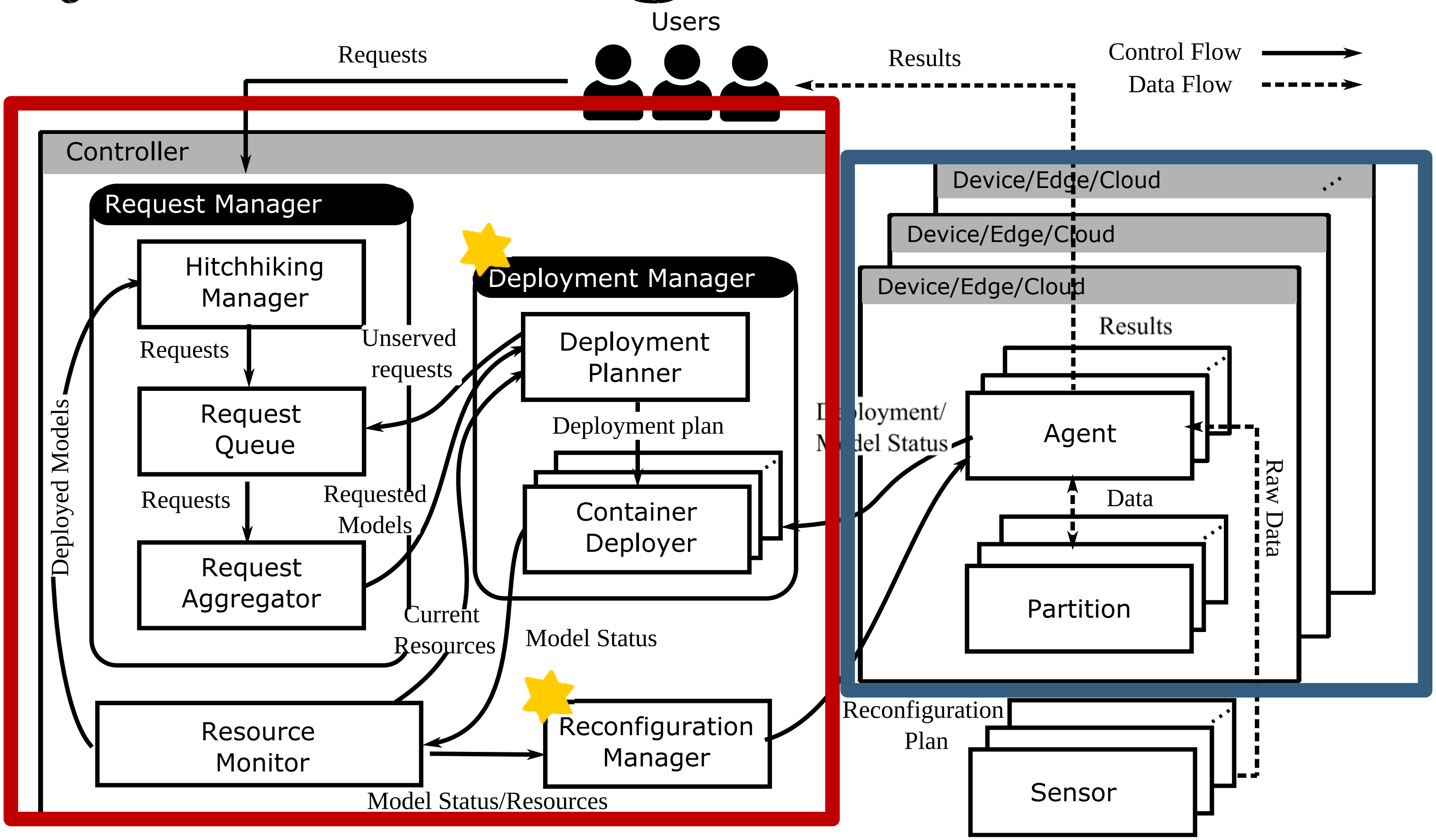
1. Topology of thing-edge-cloud has not been fully studied yet
2. Focus on one-time request

Reference

- [MobiSys’17] DeepEye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware. In Proc. of the Annual International Conference on Mobile Systems, Applications, and Services (MobiSys’17), pages 68–81, Niagara Falls, New York, June 2017.
- [SIGARCH’17] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ACM SIGARCH Computer Architecture News, 45(1):615–629, April 2017.
- [WC’19] E. Li, L. Zeng, Z. Zhou, and X. Chen. EdgeAI: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications, 19(1):447–457, October 2019
- [ATC’18] A. Jiang, D. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. Kozuch, P. Pillai, D. Andersen, and G. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In Proc. of Internatioal USENIX Annual Technical Conference (ATC’18), pages 29–42, Boston, MA, July 2018.
- [SEC’20] M. Chao, R. Stoleru, L. Jin, S. Yao, M. Maurice, and R. Blalock. AMVP: Adaptive CNN-based multitask video processing on mobile stream processing platforms. In Proc. of IEEE/ACM Symposium on Edge Computing (SEC’20), pages 96–109, Virtual, November 2020.
- [IC2E’21] A. Majeed, P. Kilpatrick, I. Spence, and B. Varghese. NEUKONFIG:Reducing edge service downtime when repartitioning DNNs. In in Proc. of IEEE International Conference on Cloud Engineering (IC2E’21), pages 118–125, San Francisco, CA, October 2021

System Overview

System Design



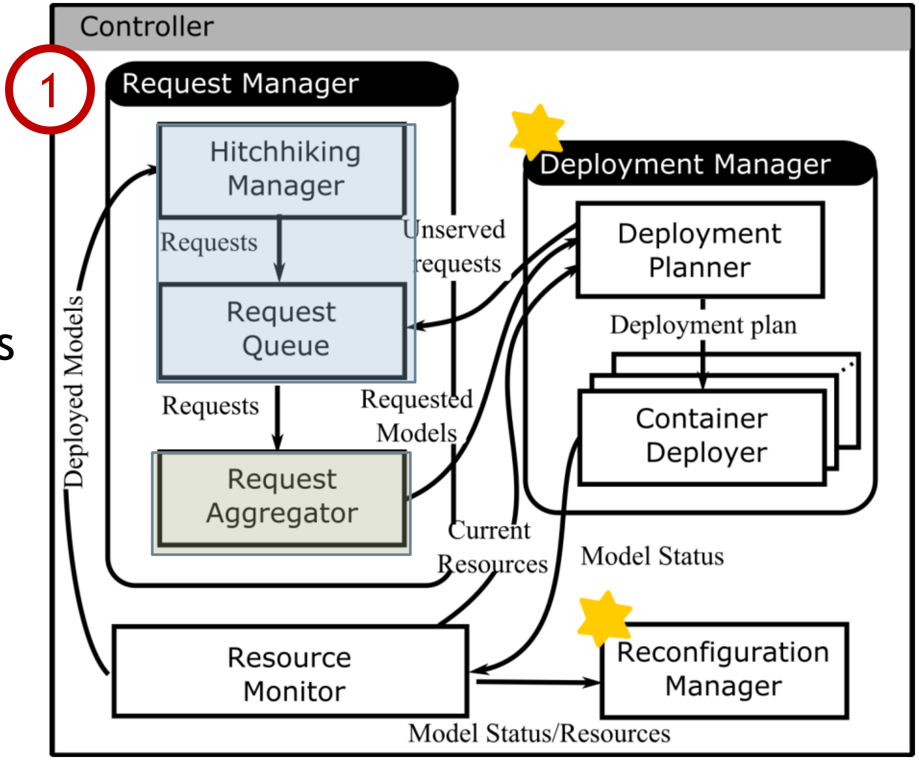
Controller

- Manage requests
- Monitor system
- Make deployment decisions

Computing devices (device/edge/cloud)

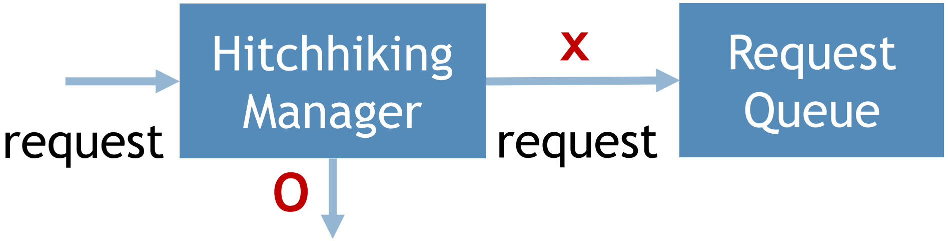
- Execute DNN layers

Controller



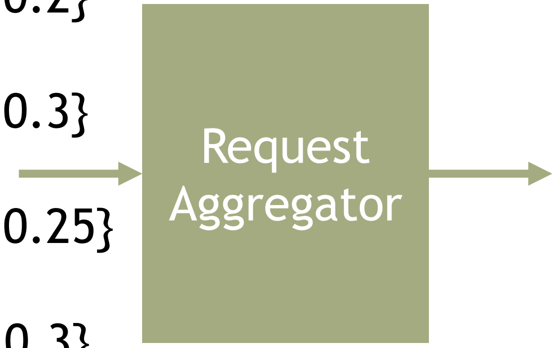
1

- **Request manager**: handle requests
  - Hitchhike requests to deployed models
  - Store requests
  - Aggregate requests



deployed model's results

Request1:
{task A1, accuracy 70%, latency 0.2}
Request2:
{task A1, accuracy 65%, latency 0.3}
Request3:
{task A3, accuracy 90%, latency 0.25}
Request4:
{task B2, accuracy 80%, latency 0.3}



A1: accident detection
A3: pedestrian counting
B2: fall detection

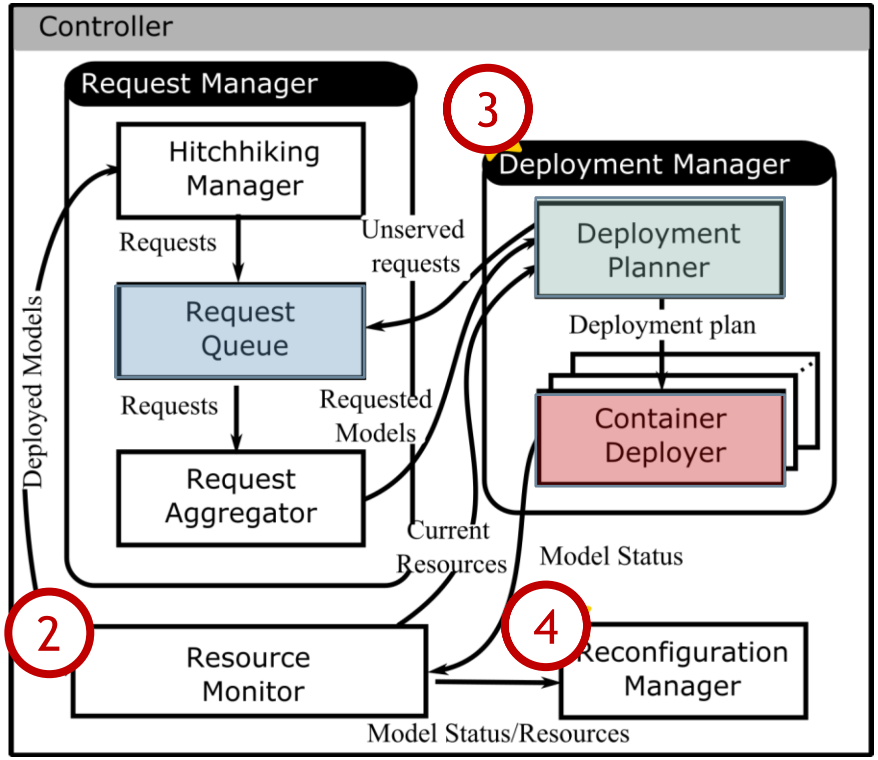
Model A (Intersection)

- task A1
- task A3
- requests 1, 2, 3

Model B (Senior house)

- task B2
- request 4

Controller



2 **Resource monitor**: keep track of the system

- Deployed models
- Computing/Networking resources

3 **Deployment Manager**: make deployment decisions and deploy the models

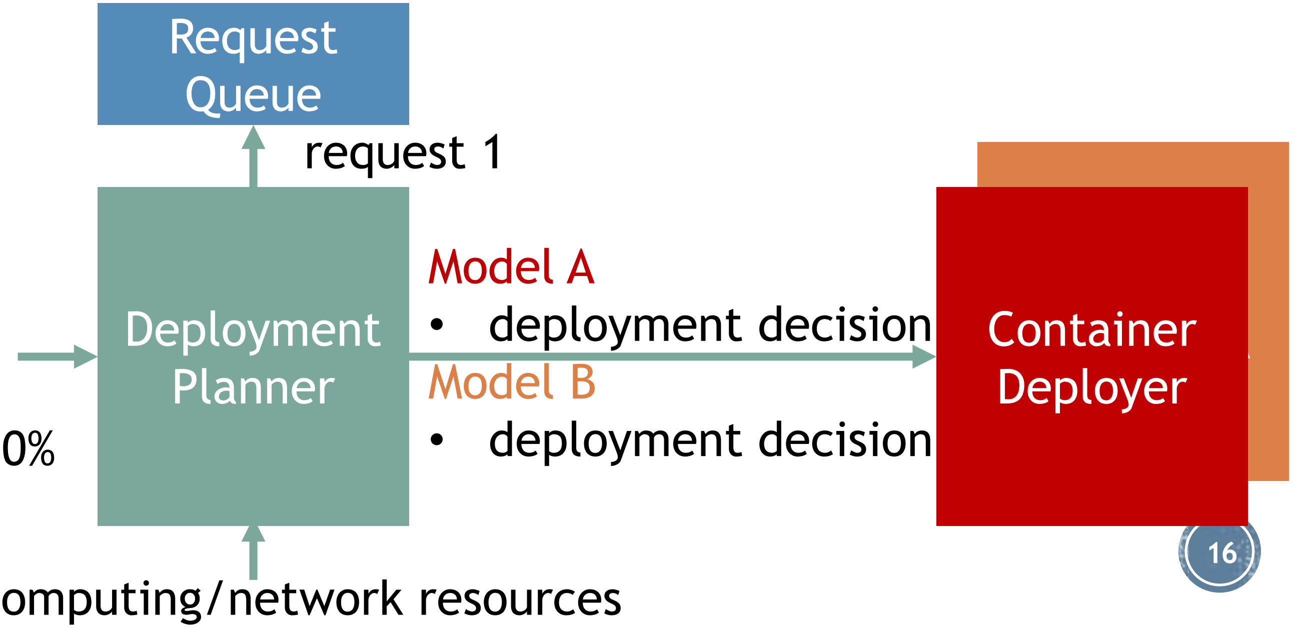
4 **Reconfiguration manager**: check runtime performance of deployed models and make reconfiguration decision

Model A

- task A1
- task A3
- requests 1, 2, 3

Model B

- task B2: accuracy 80%
- requests 4


Request Queue

request 1

Deployment Planner

Model A

- deployment decision

Model B

- deployment decision

Container Deployer

computing/network resources

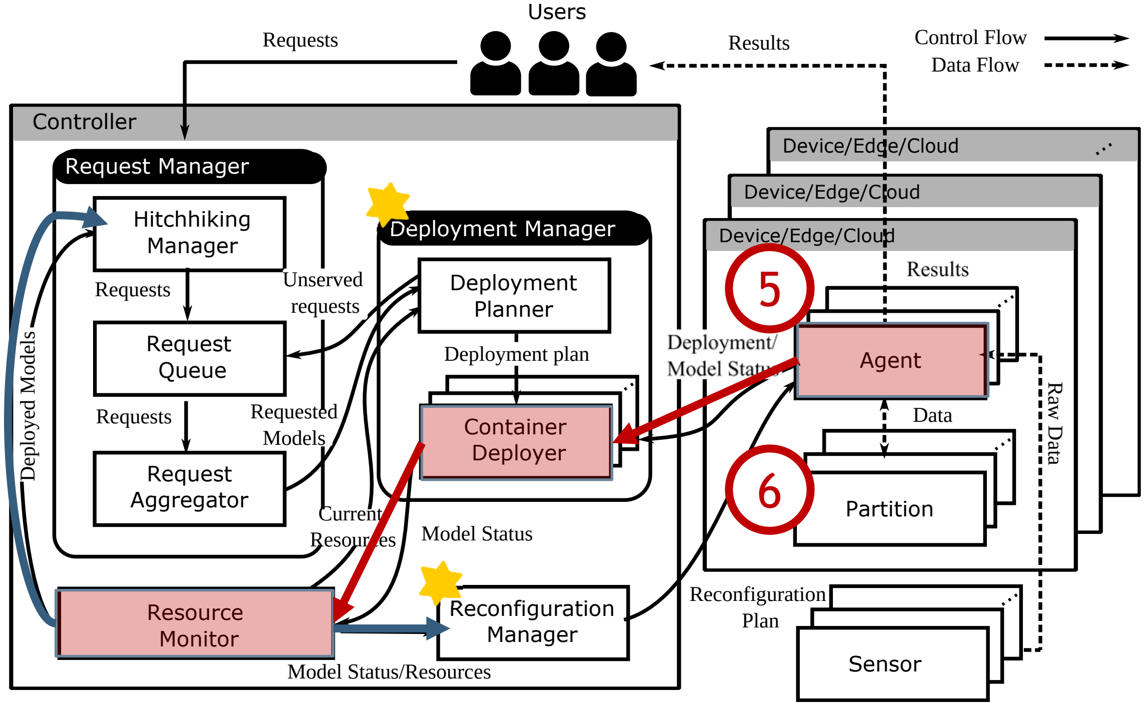
Computing Devices

5

- **Agent**: Manage DNN partition
  - Communicate with controller (model status / reconfiguration command, etc.)
  - Receive raw / intermediate data from other computing devices
  - Send intermediate data to other computing devices
  - Send inference results to users

6

- **Partition**: execute assigned DNN layers



Model A

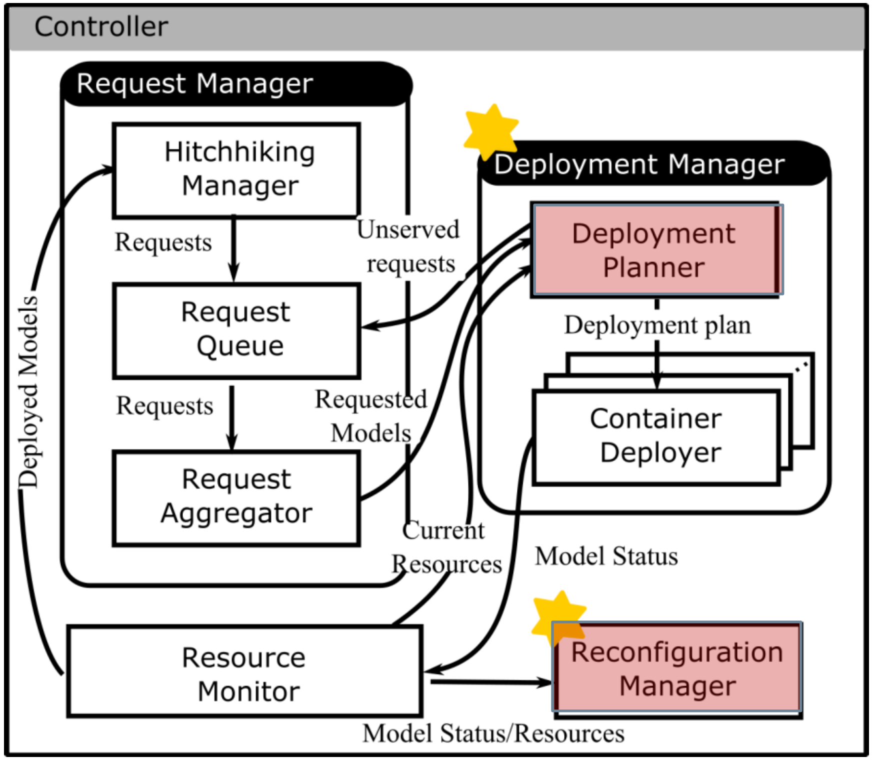
- latencies = {0.2, 0.3, 0.25…}
- status: running/finished/ reconfiguring

Research Problems

Goals of DNN Deployment

Derive *the* ***optimal deployment plans*** *for the DNNs that serve the most requests*

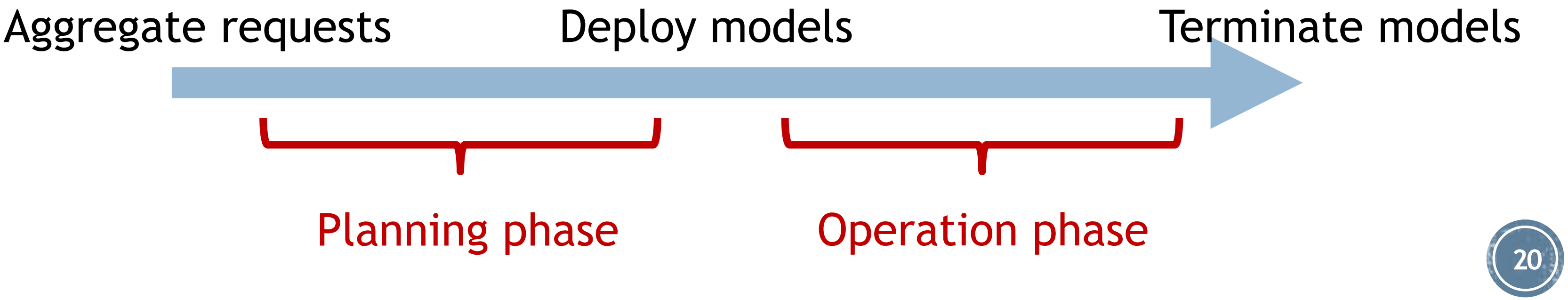
- Deployment plan
  - Determine **where to deploy** the DNN model partitions and **which exit point** to run
- Challenges
  - Resource demands of DNNs are heterogenous
  - Resource availability in the thing-to-cloud continuum is diverse and limited
  - Environment at runtime is dynamic



Deployment Plan Decision-Making Process

We devide the deployment plan decision-making process into two phases

- Planning phase: *genearte deployment plan* given resource levels -> **Deployment planning problem**
- Operation phase: check the QoS of deployed models and *reconfigure the deployment plan* at runtime if needed -> **Dynamic reconfiguration problem**



Aggregate requests

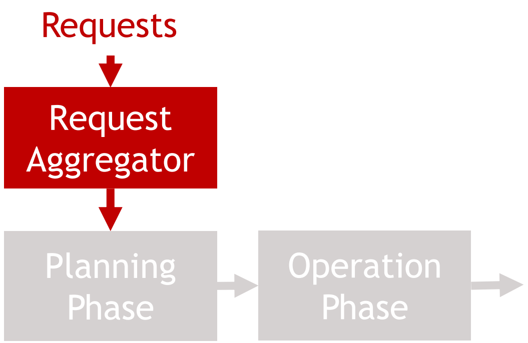
Deploy models

Terminate models

Planning phase

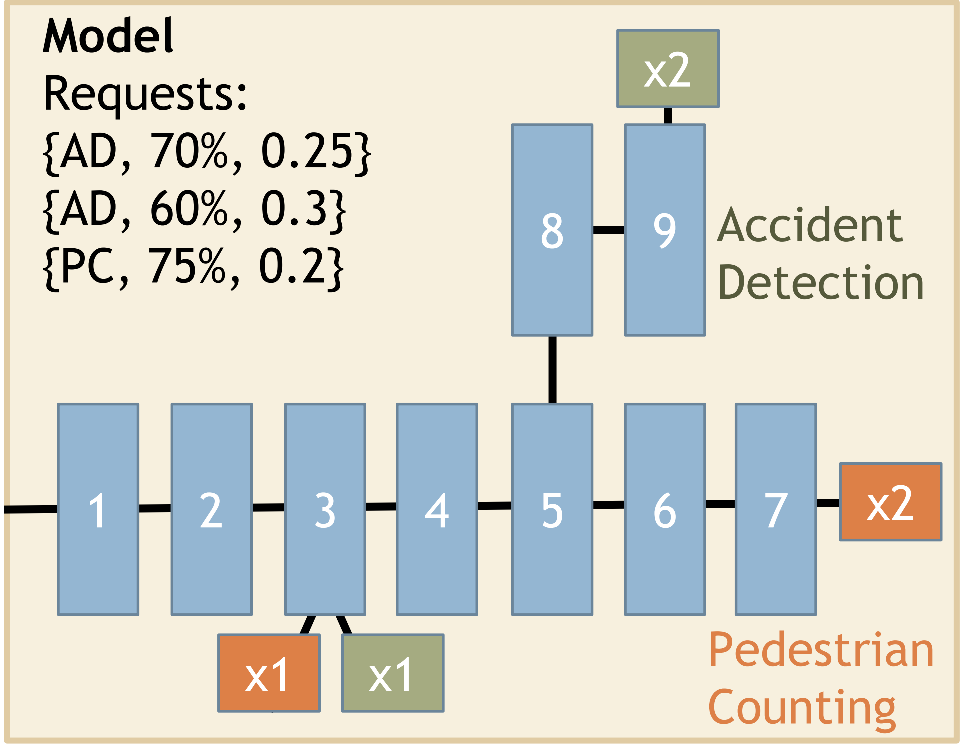
Operation phase

Planning Phase

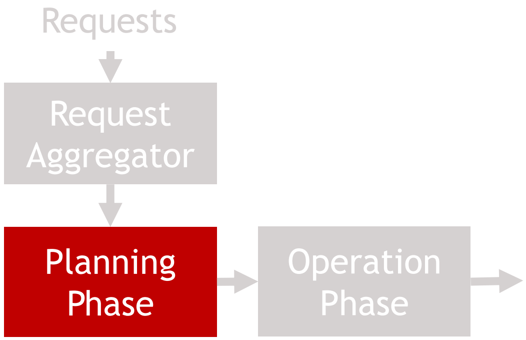


Multi-task Model Requests

- Each model contains several tasks
- Each task contains a multiple exit points and layers
- Each tasks has several requests with accuracy and latency requirements

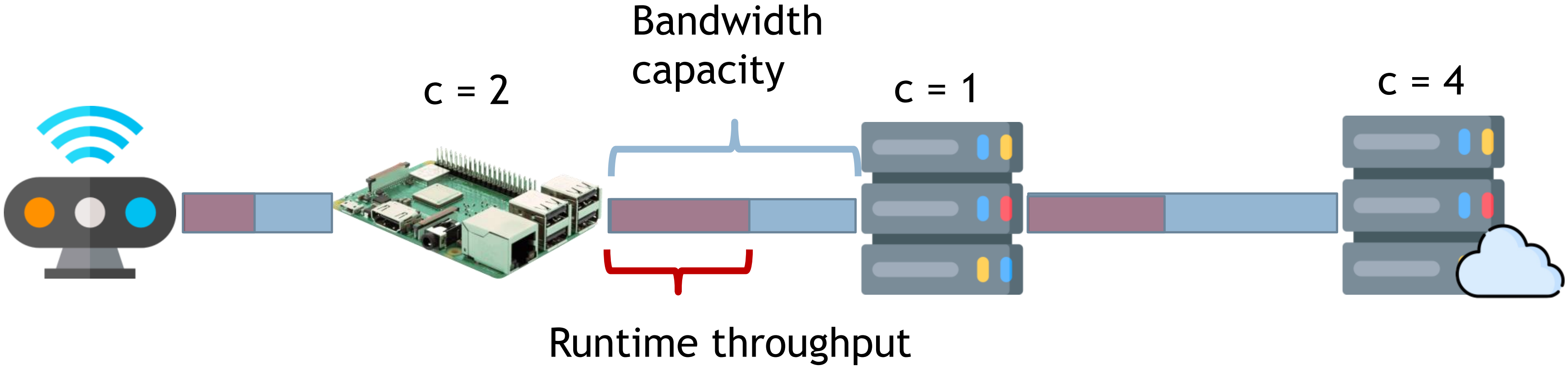


Model
Requests:
{AD, 70%, 0.25}
{AD, 60%, 0.3}
{PC, 75%, 0.2}



Resource Status

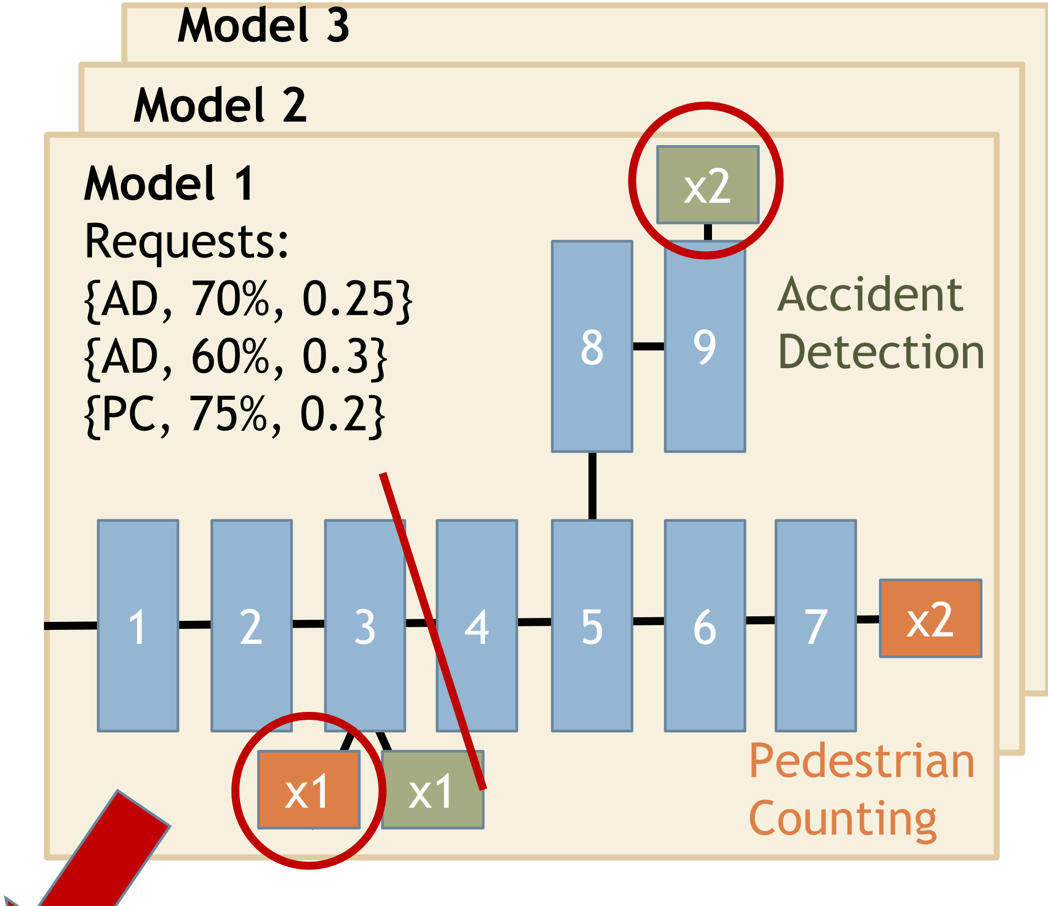
- Computing powers indicate the number of partitions that can be deployed on that device
- Network resources
  - Bandwidth capacity
  - Runtime throughput

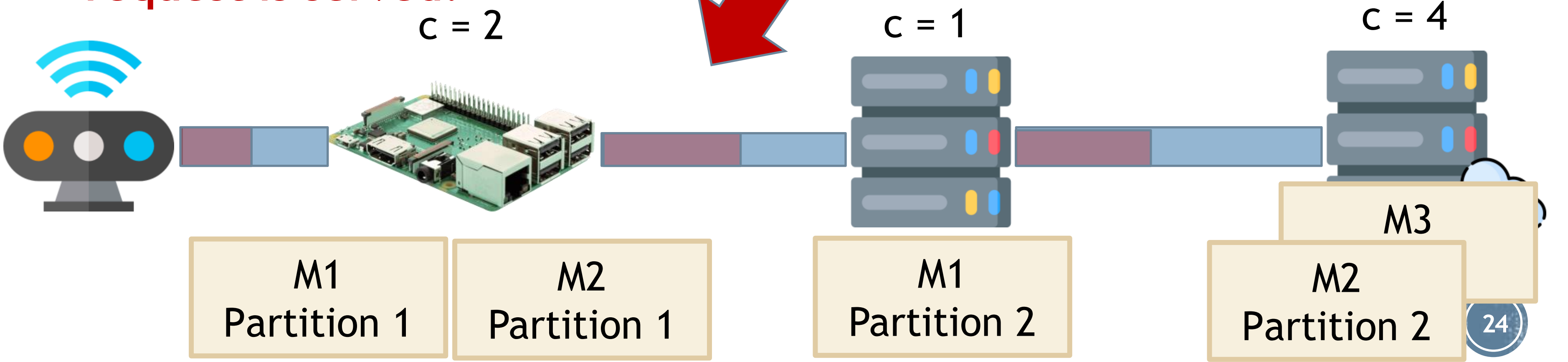


Deployment Planning Problem

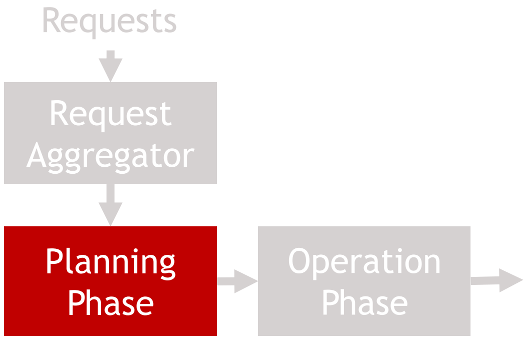
- We have a set of models to deploy
- Select exit points and partition points for each model
- **Maximize the number of served requests**

How to determine a request is served?



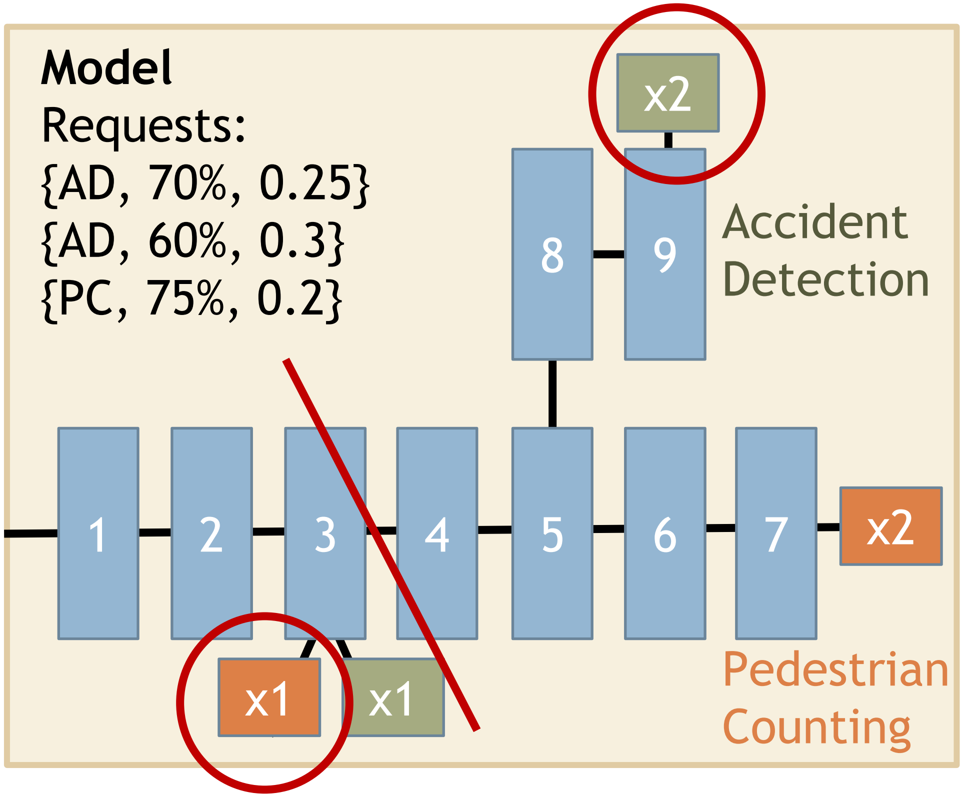


Performance Prediction



A request is considered served if both latency and accuracy requirements are met

- Latency
  - Computing latency (lookup table)
    - Layers
    - Exit layers
  - Transmission latency
- Accuracy
  - Based on exit point:
    task1 x1 -> 75%
    task2 x2 -> 90%

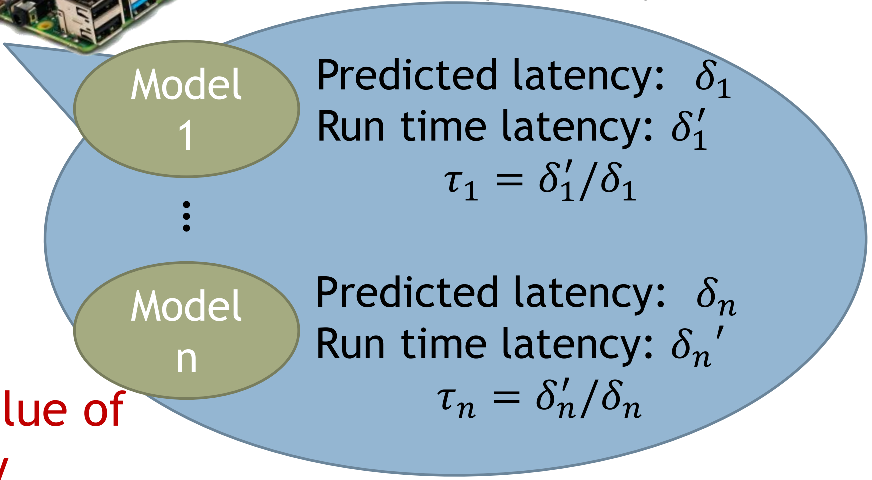


Model Requests:
{AD, 70%, 0.25}
{AD, 60%, 0.3}
{PC, 75%, 0.2}

Scaling Factors

- Lookup table may not capture the impact of runtime environment
- Leverage historical logs to update the prediction at runtime

$\tau_c = mean(\{\tau_1, \dots \tau_n\})$



Model 1

Predicted latency: δ_1
Run time latency: δ_1'
$\tau_1 = \delta_1' / \delta_1$

⋮

Model n

Predicted latency: δ_n
Run time latency: ${\delta_n}'$
$\tau_n = \delta_n' / \delta_n$

Latency Prediction

Y: lookup table value of computing latency

- Computing latency

Identify the layers running on device/edge/cloud based on partition points

$$\overbrace{\sum_{l=1}^{\substack{min(p_{m,t},\\ L_{m,t,x})}} \tau_c Y_{m,\mu'(l)}}^{\text{Computing latency on Device}} + \overbrace{\sum_{\substack{l=min(p_{m,t},\\ L_{m,t,x})+1}}^{\substack{min(p'_{m,t},\\ L_{m,t,x})}} \tau_c' Y'_{m,\mu'(l)}}^{\text{Computing latency on Edge}} + \overbrace{\sum_{\substack{l=min(p'_{m,t},\\ L_{m,t,x})+1}}^{L_{m,t,x}+1} \tau_c'' Y''_{m,\mu'(l)}}^{\text{Computing latency on Cloud}} +$$

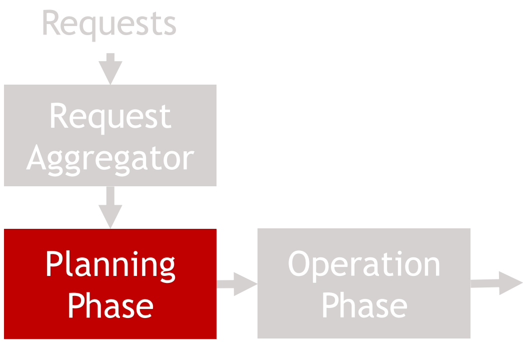
$$\tau_c \hat{Y}_{m,t} \underbrace{\mathbf{1}_{\substack{p_{m,t}< \\ L_{m,t,x}}} + \tau_c' \hat{Y}'_{m,t} \mathbf{1}_{\substack{L_{m,t,x} \le \\ p_{m,t}}} \mathbf{1}_{\substack{p'_{m,t}< \\ L_{m,t,x}}} + \tau_c'' \hat{Y}''_{m,t} \mathbf{1}_{\substack{L_{m,t,x} \le \\ p'_{m,t}}}}_{\text{Computing latency of exit layers}}$$

- Transmission latency

Layer output size

$$s_{m,t,l} = \frac{Z_{m,t,l}}{0.9(b-n)}$$

Capacity-throughput, 0.9 makes room for background traffic



Throughput

- Number of served requests

$$h_t = \sum_{r=1}^{R_t} \mathbf{1}_{\delta_r \geq \bar{\delta}_t \text{ and } a_r \leq \bar{a}_r}$$

Latency Accuracy

Formulation

- Maximize throughput under computing resource constraints

$$\underset{\mathbf{p},\mathbf{x}}{maximize} \sum_{m=1}^{M} \sum_{t=1}^{T_m} h_t$$

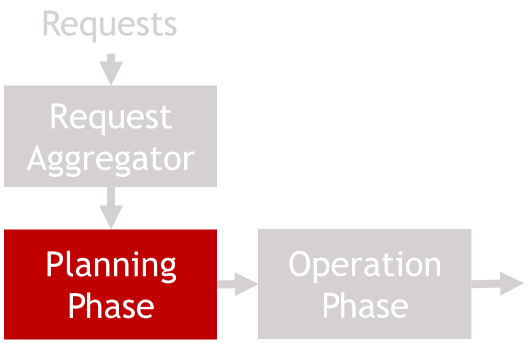
$$s.t.: \sum_{m=1}^{M} \hat{c}_m \leq c; \quad \text{Device}$$

$$\sum_{m=1}^{M} \hat{c}'_m \leq c'; \quad \text{Edge}$$

$$\sum_{m=1}^{M} \hat{c}''_m \leq c''. \quad \text{Cloud}$$

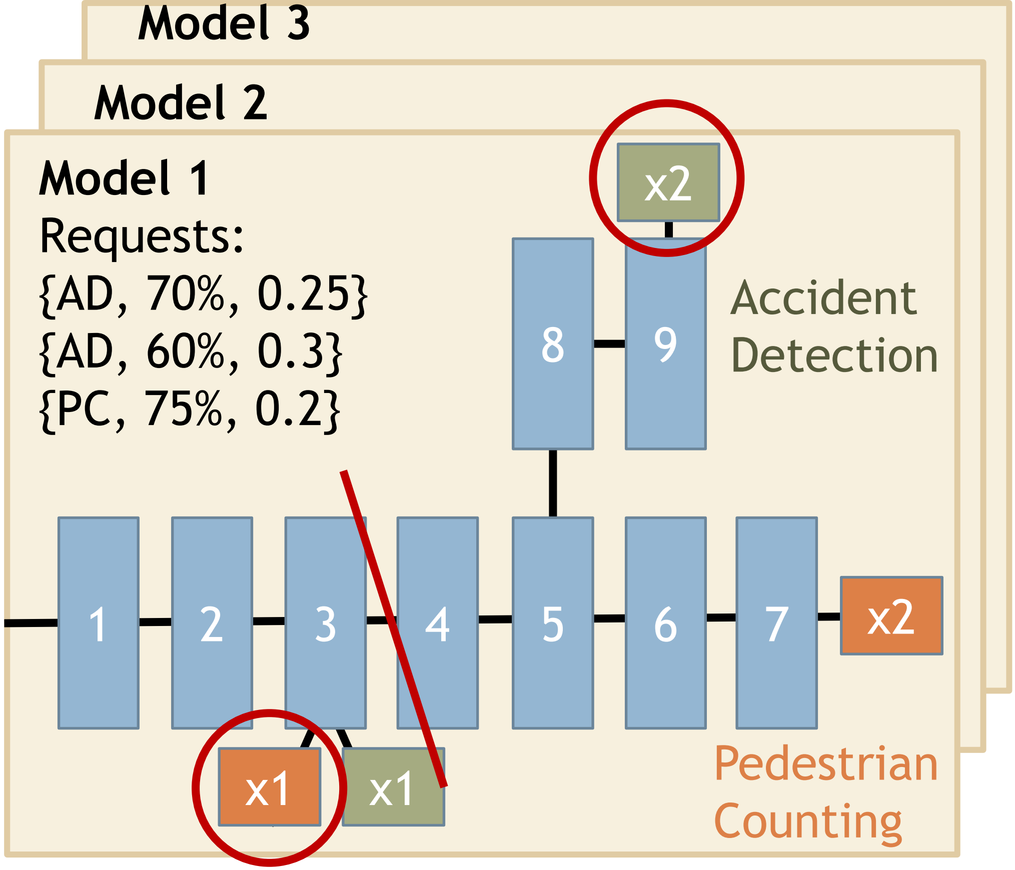
Based on partition points, we can determine if a computing device is participate in the execution of model m

Deployment Planning Algorithm



- Serve models with more requests first
- Satisfy accuracy requirements (exit points) and choose the latency (partition points) with highest throughput

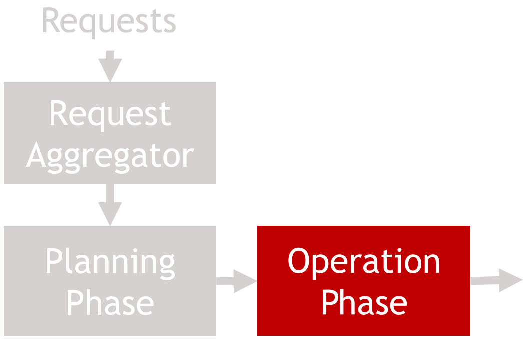
1. Sort models by number of requests
2. For each model
   (1) Select exit point based on accuracy requirements
   (2) Select partition points with max satisfied #requests
   (3) Update available resources


Model 1
Requests:
{AD, 70%, 0.25}
{AD, 60%, 0.3}
{PC, 75%, 0.2}

If there are multiple partition points with the same throughput, select the one with lowest latency

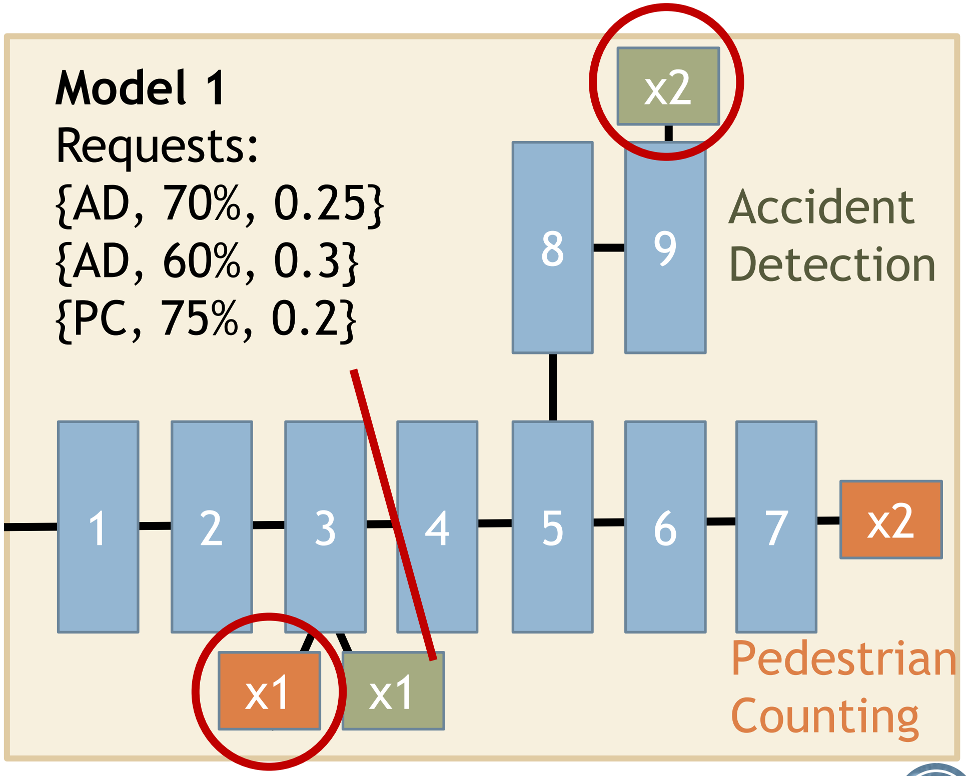
Operation Phase

Dynamic Reconfiguration Problem

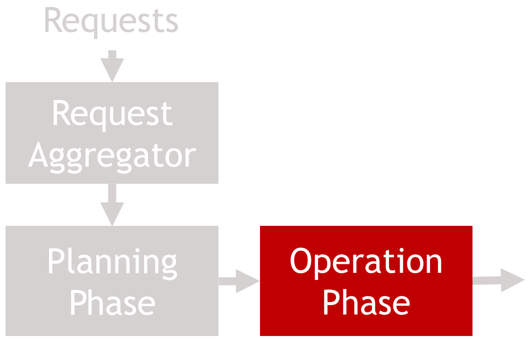


- A performance drop is detected on a deployed model m
- Select new exit points and partition points to maxmize the throughput
- The computing resources are limited to those already assigned in the planning phase

Search space is reduced to single model and fixed devices

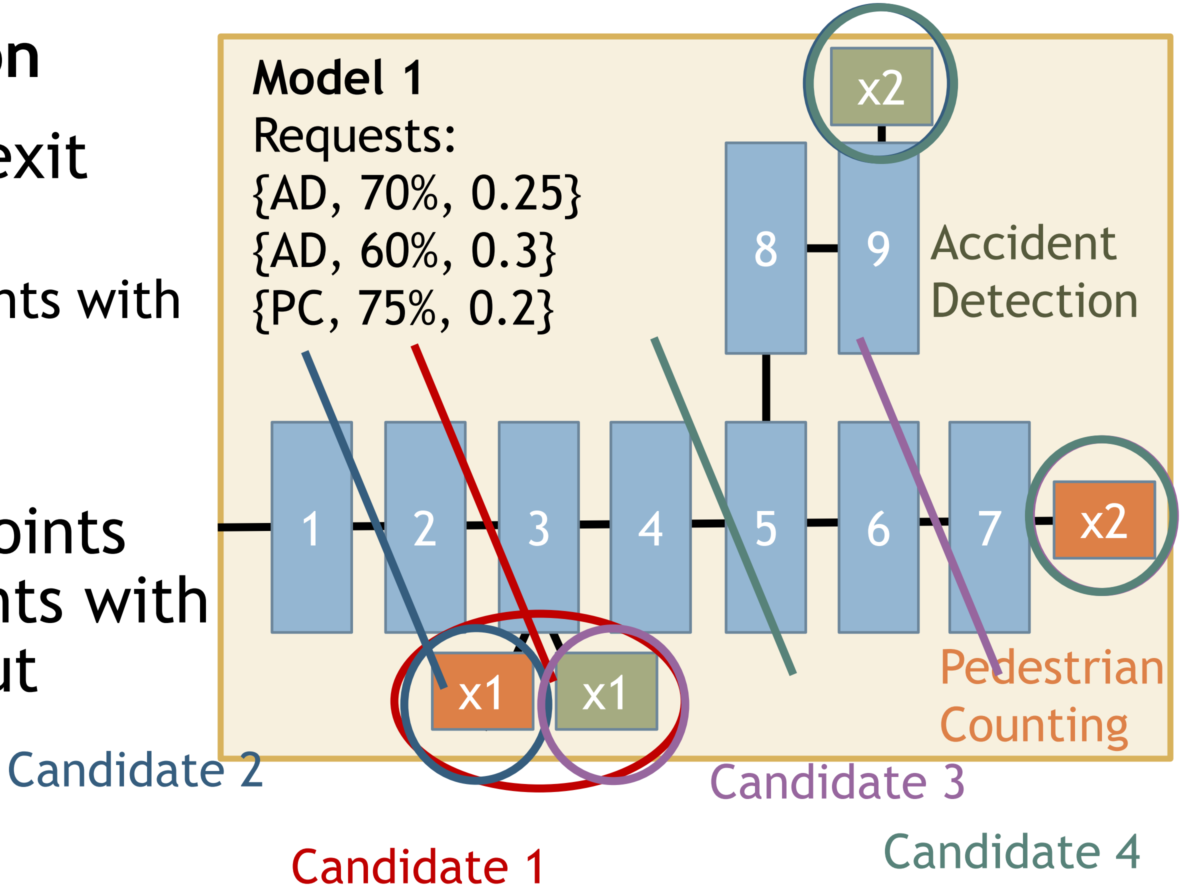


Dynamic Reconfiguration Algorithm



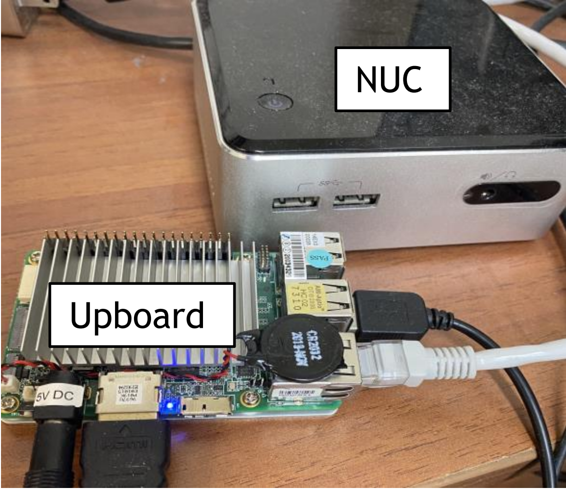
Find optimal decision

1. Find all possible exit points
   (i) Select partition points with lowest latency
   (ii) Predict throughput
2. Return the exit points and partition points with highest throughput

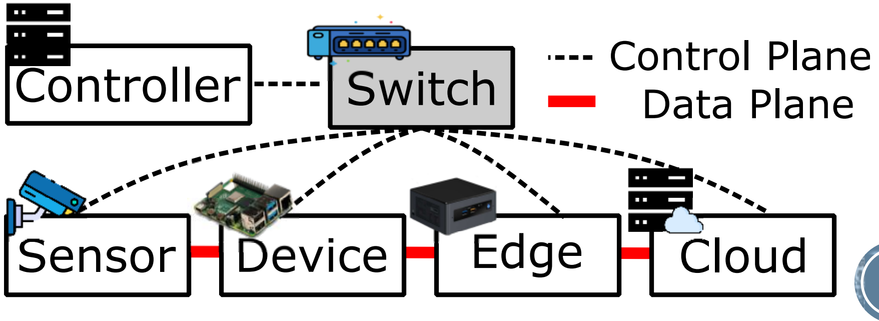

Model 1
Requests:
{AD, 70%, 0.25}
{AD, 60%, 0.3}
{PC, 75%, 0.2}

Implementations

Testbed

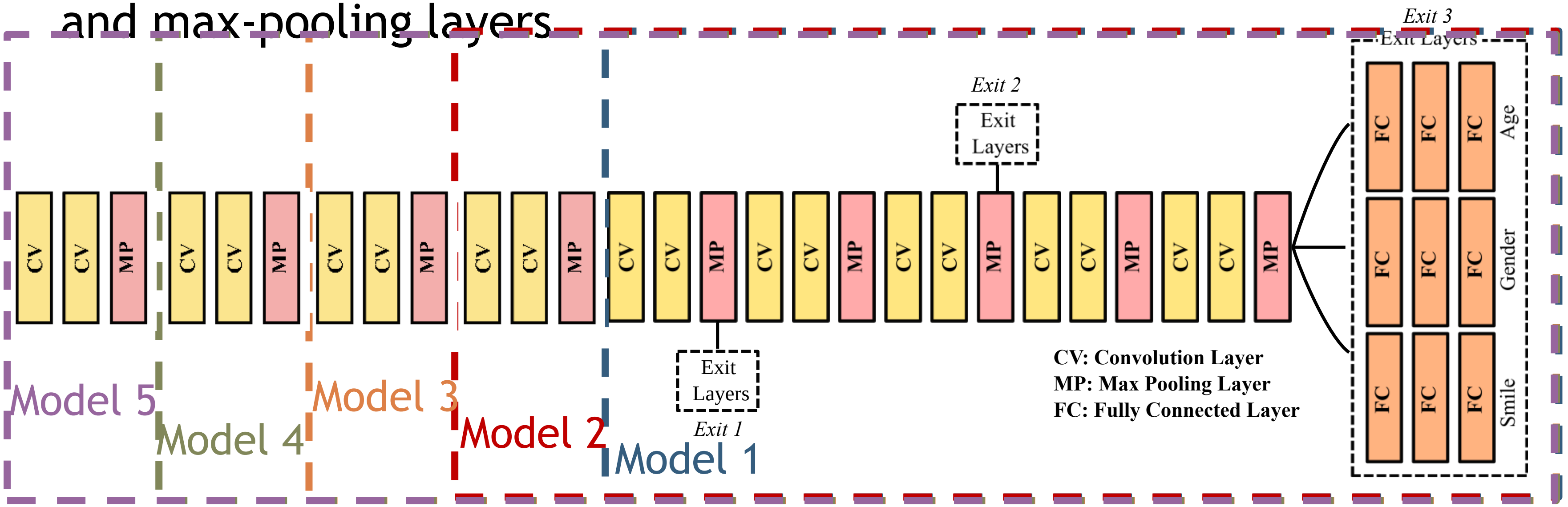


- Computation devices
  - IoT Device (Upboard): Intel Atom CPU @1.44 GHz and 4 GB RAM
  - Edge (NUC): Intel i3 CPU @1.7 GHz and 16 GB RAM
  - Cloud (PC): Intel i7 CPU @3.4 GHz and 24 GB RAM
- Open-sources tools
  - Kubernetes
  - Docker
  - ZeroMQ
- Network topology
  - System communicates to the computing devices via **control plane**
  - Computing devices send data via **data plane**



Multi-task Networks

- Pytorch based Age-Smile-Gender multi-task DNNs [Informatica’18]
- Enable early exits with BranchyNet [ICPR’16] structure
- Run inference for 100 times and use the average computing time to build the lookup table
- Create 5 models by varying the number of convolution and max-pooling layers



CV: Convolution Layer
MP: Max Pooling Layer
FC: Fully Connected Layer

[Informatica’18] S. Viet and C. Bao. Effective deep multi-source multi-task learning frameworks for smile detection, emotion recognition and gender classification. Informatica, 42(3), September 2018.

[ICPR’16] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In Proc. of IEEE International Conference on Pattern Recognition (ICPR’16), pages 2464–2469, Cancun, Mexico, December 2016.

Compared Algorithms

- Baselines
  - Neurosurgeon (NEU) [SIGARCH’17]: select partition points for shortest latency
  - Edgent [WC’19]: select exit points with highest accuracy that has partition points achieving required latency requirement (we forced it to choose the exit points that satisfying the accuracy requirements)
- T2C with different features

Algms / Features	NEU	Edgent	$T2C_{M}$	$T2C_{MH}$	$T2C_{MHE}$	$T2C_{MHER}$
Multi-task			✓	✓	✓	✓
Hitchhiking				✓	✓	✓
Early exit		✓			✓	✓
Reconfiguration						✓

[SIGARCH’17] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ACM SIGARCH Computer Architecture News, 45(1):615–629, April 2017.

[WC’19] E. Li, L. Zeng, Z. Zhou,and X. Chen. EdgeAI: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications, 19(1):447–457, October 2019.

Evaluations

Evaluation Setup

- Experiment duration: 5 min
- Run (# repeats): 5
- # Models: 5
- Sliding window of scaling factors: 1 min
- Aggregation period
  - Planning phase: 10 s
  - Operation phase: 20 s
- Reconfiguration period: 45 s
- Request arrival rate (Poisson arrival rate of a Public 311 call trace)
  - Planning phase: {1X, 20X, 40X, 60X, 80X}
  - Operation phase: {40X, 60X, 80X, 100X, 120X}

- Network link:
  - Sensor-IoT (**Bluetooth**): 1 Mbps, latency 1ms
  - IoT-Edge (**WiFi**): 24 Mbps, latency 3ms
  - Edge-Cloud (**Broadband**): 40 Mbps, latency 5 ms
- Computing powers:
  - Device: 3
  - Edge: 3
  - Cloud: 7

Perforamance Metrics

- **Throughput:** the number of served requests
- **Satisfied ratio:** the % of requests meeting both accuracy and latency requirements

- **Latency:** the inference time of IoT analytics
- **Accuracy:** the accuracy of IoT analytics
- **Queuing time:** the waiting time of a each request in the request queue

- **CPU utilization:** the fraction of algorithm running time on our Intel i7 PC @3.6 GHz
- **Deployment time:** time difference between deployment plan generation and container launch
- **Data plane throughput:** the bandwidth consumption among DNN partitions
- **Control overhead:** the bandwidth consumption of control messages

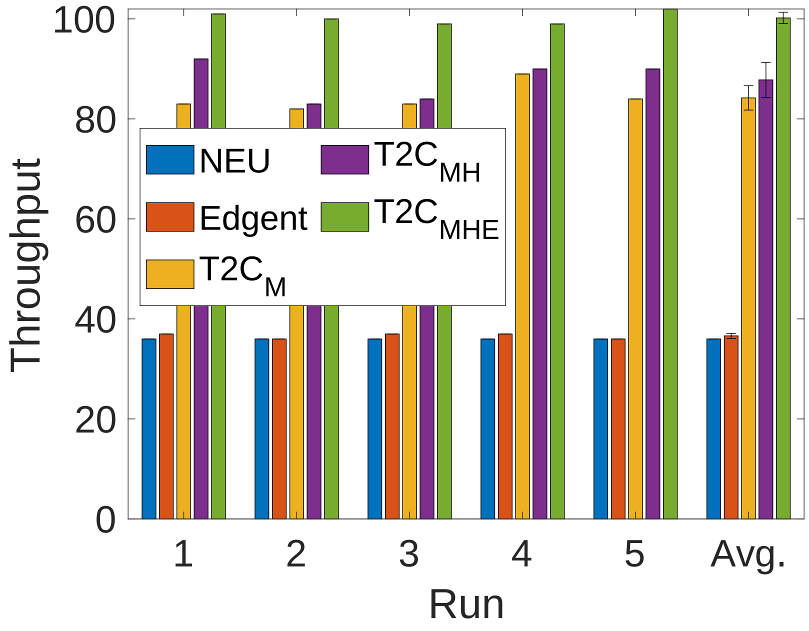
Results: Planning Phase

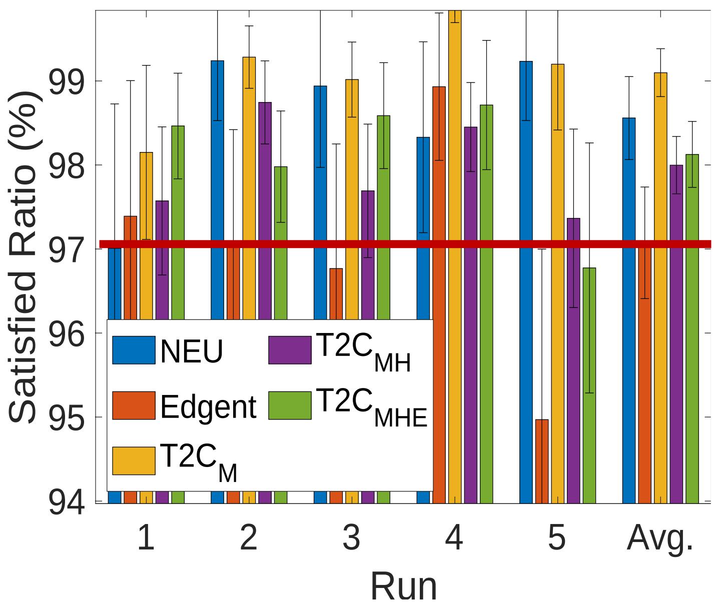
Multi-task and Hitchhiking Improve Throughput without Sacrificing Satisfied Ratio

Planning Phase → Operation Phase

M: multi-task
H: hitchhiking
E: early exit

- T2C$_{M}$ and T2C$_{MH}$ achieves 2.34X and 2.44X of throughput compared to NEU
- T2C$_{MHE}$ delivers 2.74X of throughput compared to Edgent
- Our algorithms satisfy > 97% requests at runtime





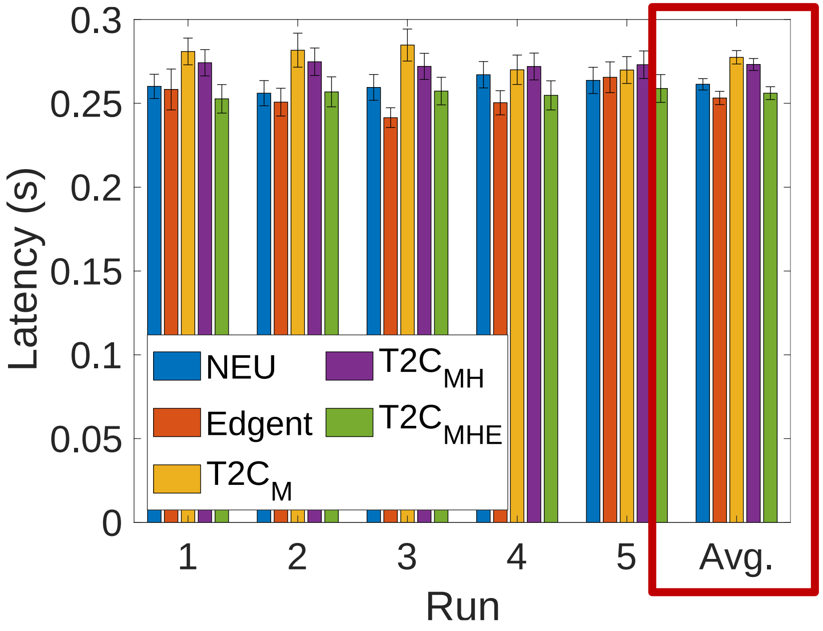
Planning Phase

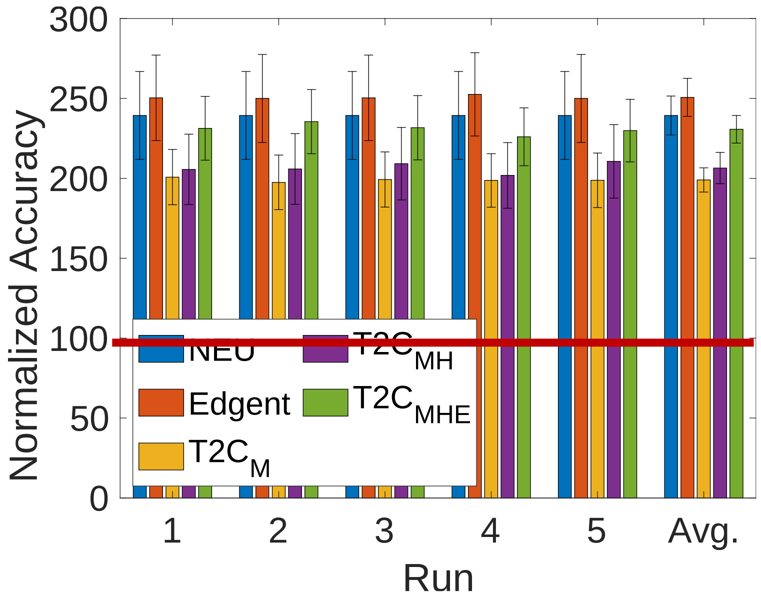
Operation Phase

Early Exits Reduce Latency while Satisfying Accuracy Requirements

M: multi-task
H: hitchhiking
E: early exit

- T2C$_{MHE}$ and Edgent result in 3.85% lower latency compared to NEU and 7.68% lower latency compared to T2C$_{M}$



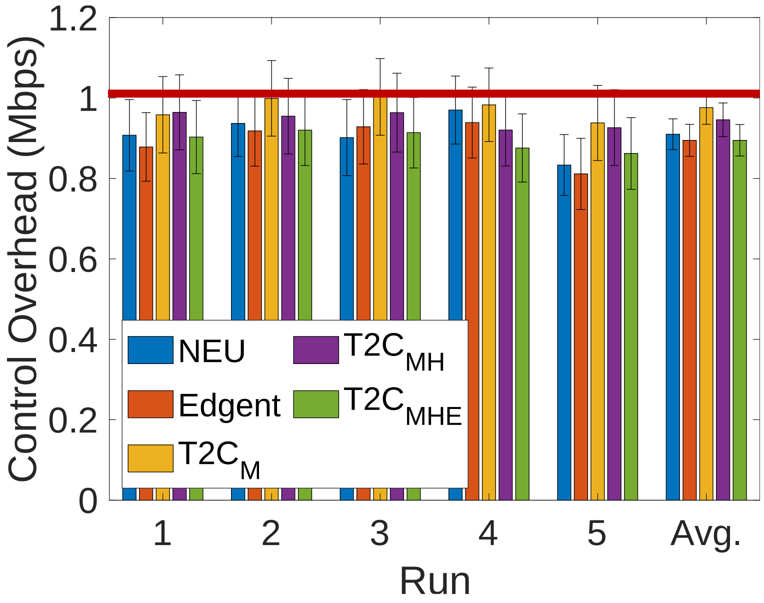


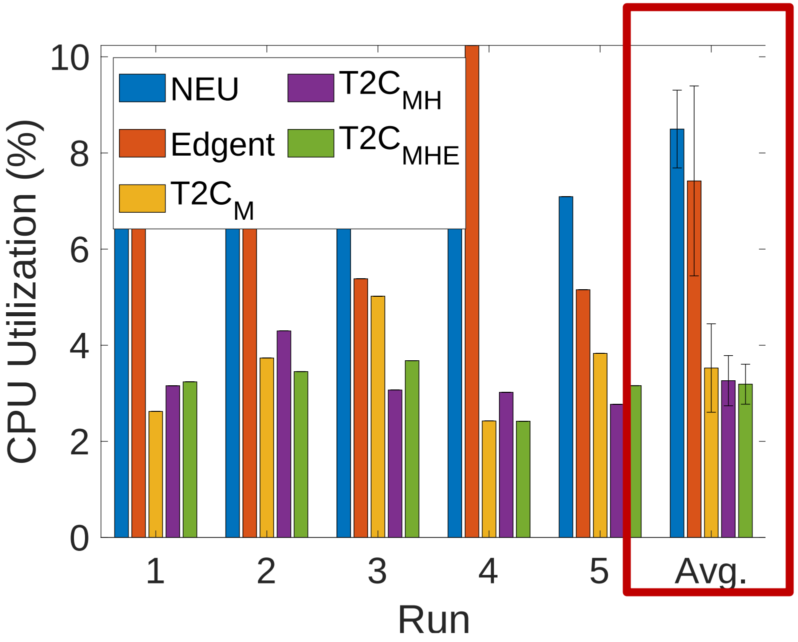
Planning Phase → Operation Phase

Our System Incurs Acceptable Overhead

M: multi-task
H: hitchhiking
E: early exit

- All algorithms incur ~1Mbps control overhead -> Acceptable
- T2C$_{MHE}$ obtains 37.86% and 42.86% of CPU utilization compared to NEU and Edgent



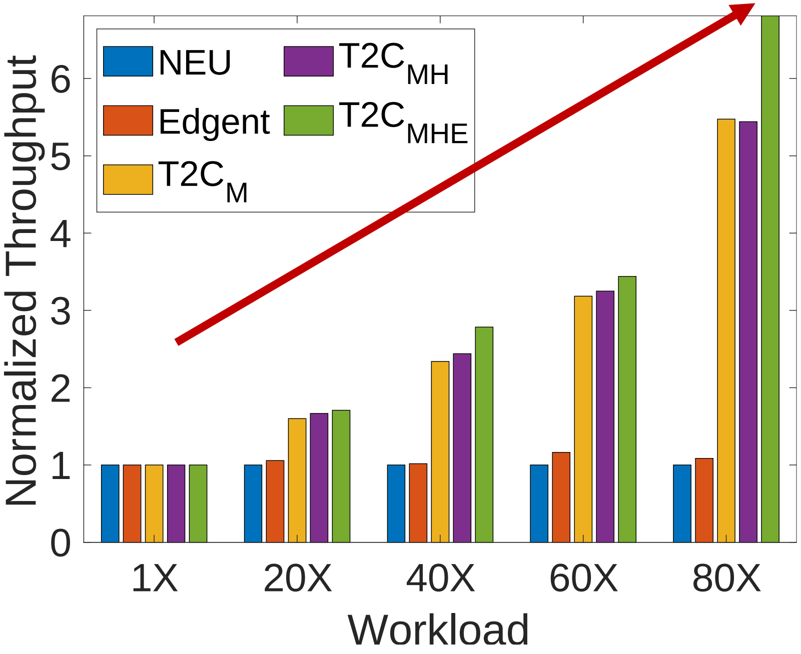


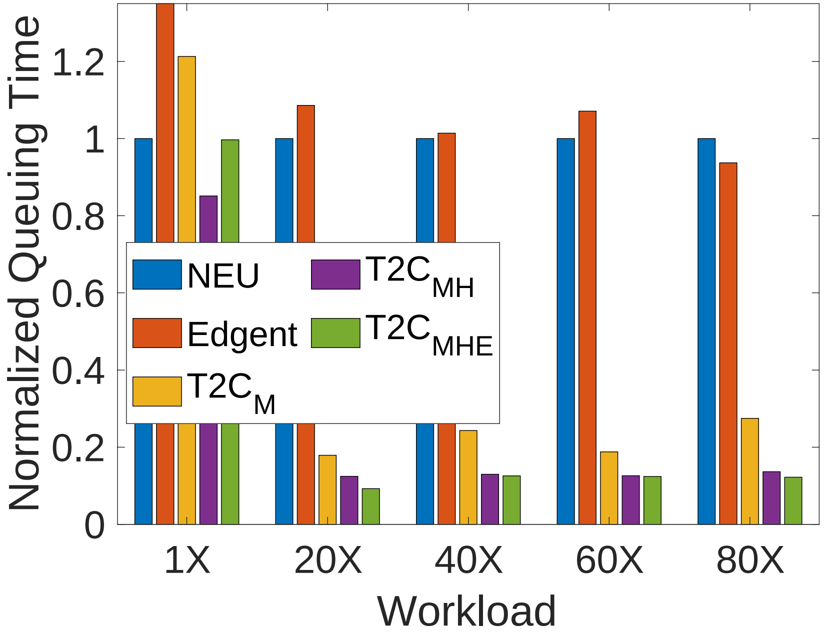
Our System Scales Well under Heavy Workload

Planning Phase → Operation Phase

M: multi-task
H: hitchhiking
E: early exit

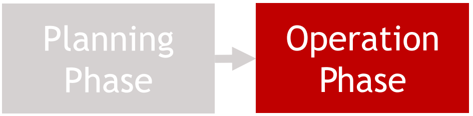
- T2C$_{MH}$ and T2C$_{MHE}$ achieve at most 5.4X and 6.8X of throughput compared to NEU
- Queuing time of T2C algorithms is as low as 12.21% of that of NEU





Results: Operation Phase

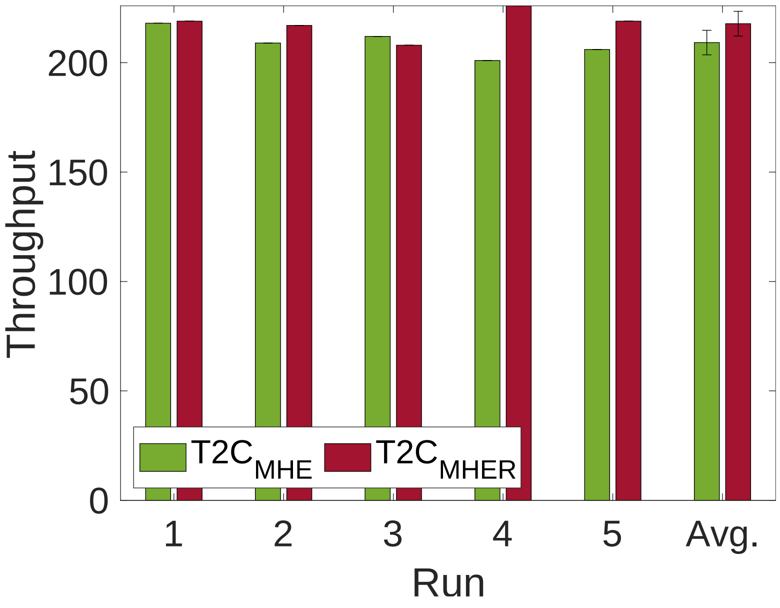
$T2C_{MHER}$ Obtains Highest Throughput and Satisfied Ratio

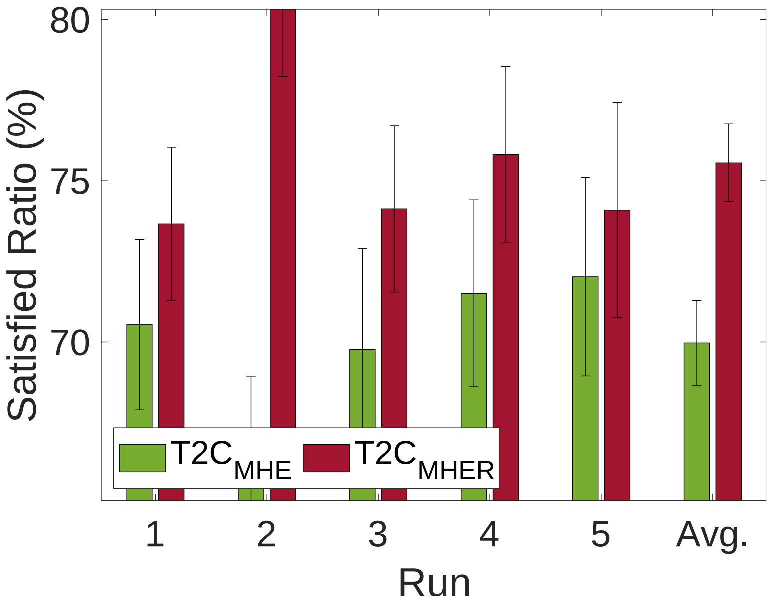


MHE: Without reconfiguration

MHER: With reconfiguration

- To trigger reconfiguration, we increase the workload and aggregation period to accumulate more requests in a model
- We cut network bandwidth by half to emulate ***network congestion*** after the 1st deployment decisions are made (@20s)





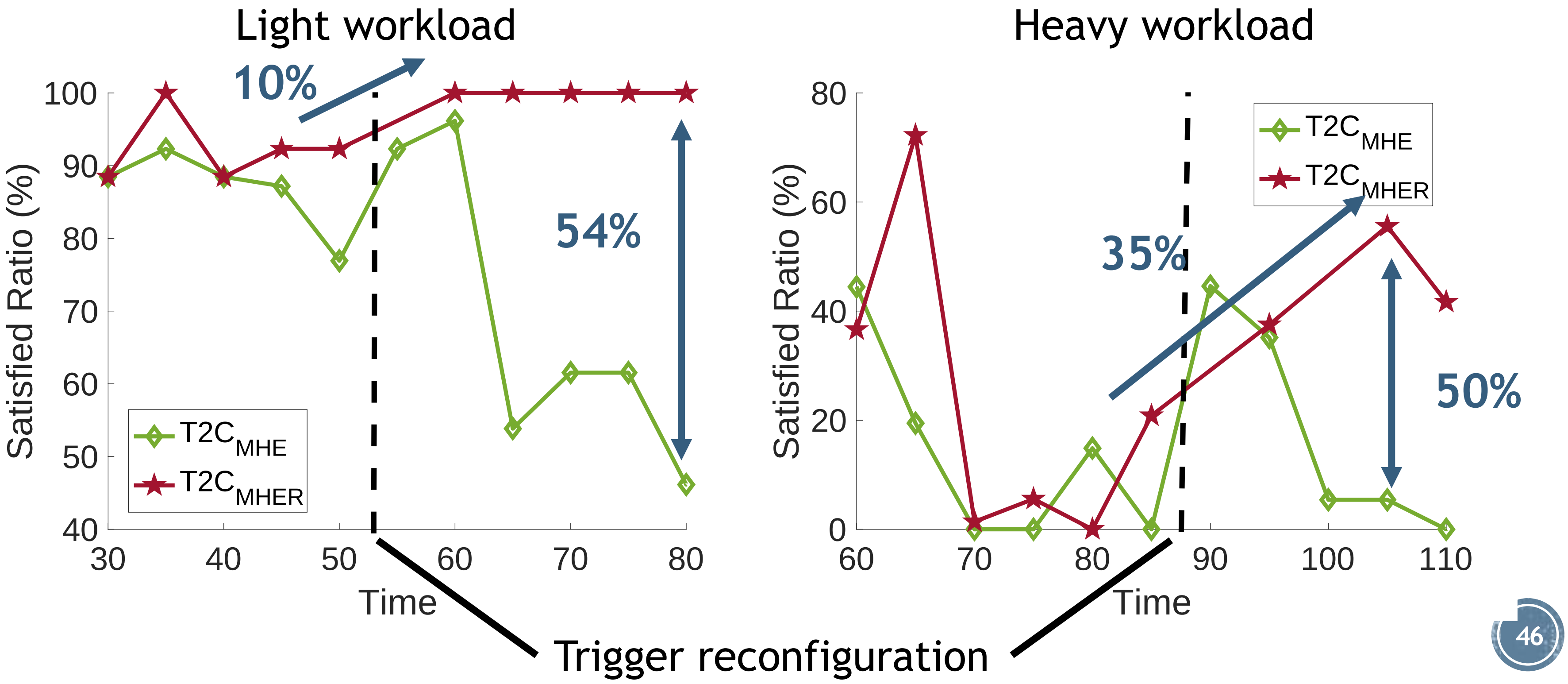
Reconfiguration Improves Satisfied Ratio

Planning Phase → Operation Phase

MHE: Without reconfiguration

MHER: With reconfiguration

- Zoom into the performance of a single model
- Light workload: 4 models
- Heavy workload: up to 13 models



Light workload

Heavy workload

Trigger reconfiguration

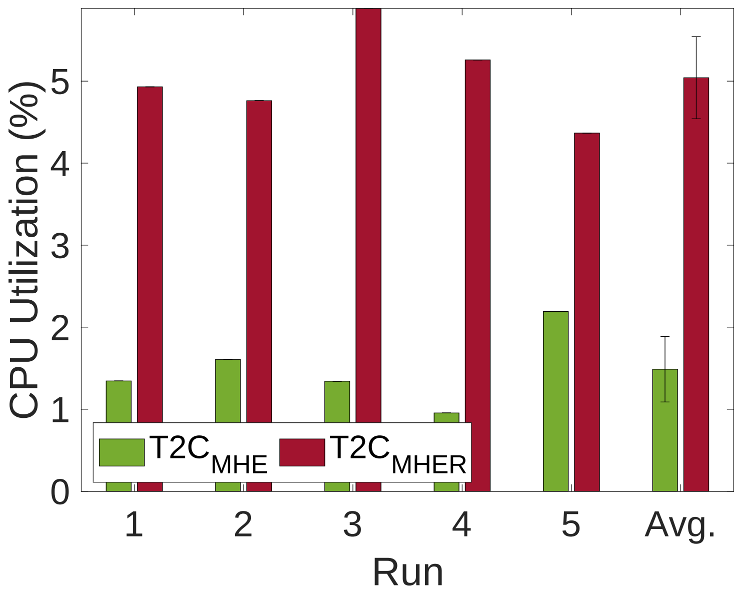
T2C$_{MHER}$ Incurs Acceptable Overhead

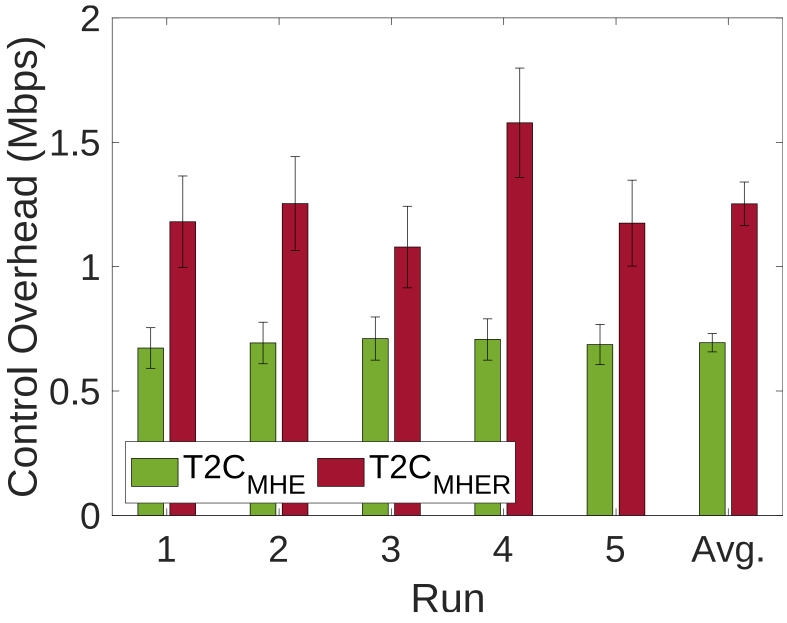
Planning Phase → Operation Phase

MHE: Without reconfiguration

MHER: With reconfiguration

- T2C$_{MHER}$ has a CPU utilization of ≤ 5%, which is only a few ms for each invocation
- T2C$_{MHER}$ consumes extra bandwidth for reconfiguration, but the average throughput is < 1.5 Mbps





Summary

Multi-task and Hitchhiking improve throughput

Early exit reduces latency

Reconfiguration increases satisfied ratio

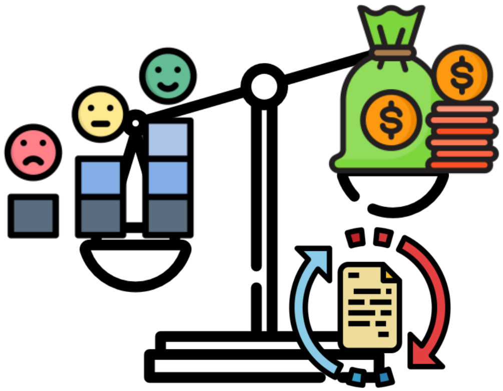
- $T2C_{MHE}$ achieves 6.8X throughput at low latency
- $T2C_{MHER}$ improves up to 35% satisfied ratio

Conclusion & Future Work

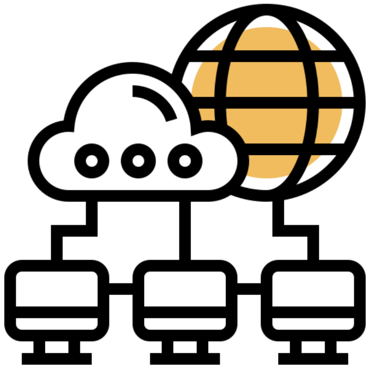
Conclusion

- Propose the T2C system for deploying DNNs in a thing-to-cloud continuum
- Leverage multi-task, hitchhiking, early exits, and reconfiguration
- Divide the deployment decision-making process into planning and operation phase
- Solve the deployment planning problem and dynamic reconfiguration problem
- T2C_{MHE} achieves 6.8X throughput at low latency
- T2C_{MHER} improves up to 35% satisfied ratio

Future Work



Cost of reconfiguration



Multiple devices
in each infrastructure layer

Adaptive aggregation period

Thank you for listening

Chia-Ying Hsieh (cyinghsieh@gmail.com)

Thanks for the help of Prof. Hsu, Prof. Venkatasubramanian, Praveen Venkateswaran, Tung-Chun Chang, Chao-Wen Chen, Kuan-Yu Lee, and all lab mates

- Publications
  - **C. Hsieh**, P. Venkateswaran, N. Venkatasubramanian, and C. Hsu T2C: A Multi-User System for Deploying DNNs in a Thing-to-Cloud Continuum. In Proc. of IEEE International Conference on Mobility, Sensing and Networking (MSN'22), December 2022, invited paper.
  - P. Venkateswaran, K. Benson, **C. Hsieh**, C. Hsu, S. Mehrotra, and N. Venkatasubramanian REAM: A Framework for Resource Efficient Adaptive Monitoring of Community Spaces. Pervasive and Mobile Computing, September 2021.
  - T. Chang, G. Bouloukakis, **C. Hsieh**, C. Hsu, and N. Venkatasubramanian SmartParcels: Cross-Layer IoT Planning for Smart Communities. In Proc. of ACM/IEEE Conference on Internet of Things Design and Implementation (IoTDI'21), May 2021.
  - T. Chang, G. Bouloukakis, **C. Hsieh**, C. Hsu, and N. Venkatasubramanian SmartParcels–A What-if Analysis and Planning Tool for IoT-Enabled Smart communities. In Proc. of ACM/IEEE Conference on Internet of Things Design and Implementation (IoTDI'21), Demo Session, May 2021.
  - **C. Hsieh**, Y. Li, C.Hsu, Y.Kuo, C. Chen, C. Hsu, and J. Sheu Stream Processing of Software-Defined Video Analytics on a Smart Campus. In Proc. of IEEE International Conference on Big Data Intelligence and Computing (DataCom'19), Demo Session, Kaohsiung, Taiwan, October 2019.

Multi-task Networks

- Originally proposed to share weights of prefix layers of different tasks [ML’97]
- Trained several DNNs for multiple video analytics with different number of shared layers to trade accuracy and complexity [ATC'18]
- Trained popular networks and replaced the classifier (fully-connected layers) and some of the convolution layers with that of the target tasks [SEC’20]

Deploy multi-task model in the thing-to-cloud continuum has not been thoroughly studied

[ML’97] R. Caruana. Multitask learning. Springer Machine Learning, 28(1):41-75, July 1997.

[ATC’18] A. Jiang, D. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. Kozuch, P. Pillai, D. Andersen, and G. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *Proc. of Internatioal USENIX Annual Technical Conference (ATC’18)*, pages 29–42, Boston, MA, July 2018.

[SEC’20] M. Chao, R. Stoleru, L. Jin, S. Yao, M. Maurice, and R. Blalock. AMVP: Adaptive CNN-based multitask video processing on mobile stream processing platforms. In *Proc. of IEEE/ACM Symposium on Edge Computing (SEC’20)*, pages 96–109, Virtual, November 2020.

Early Exits and Deployment

- Infer with prefix layers as long as the accuracy reaches the target [ICPR'16]
- Dynamically selected partition and exit points between an IoT device and an edge server [WC’19]
- Partitioned the DNNs and guaranteed that the local device has at least an exit point [MobiCom’20]

1. Only considered the decision between thing-edge or edge-cloud, the topology of thing-edge-cloud has not been fully studied yet
2. Focus on one-time request

[ICPR’16] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proc. of IEEE International Conference on Pattern Recognition (ICPR’16)*, pages 2464–2469, Cancun, Mexico, December 2016.
[WC’19] E. Li, L. Zeng, Z. Zhou, and X. Chen. EdgeAI: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communi- cations*, 19(1):447–457, October 2019
[MobiCom’20]S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane. Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proc. of the Annual International Conference on Mobile Computing and Networking (Mobi- Com’20)*, pages 1–15, London, UK, April 2020.

Reconfiguration

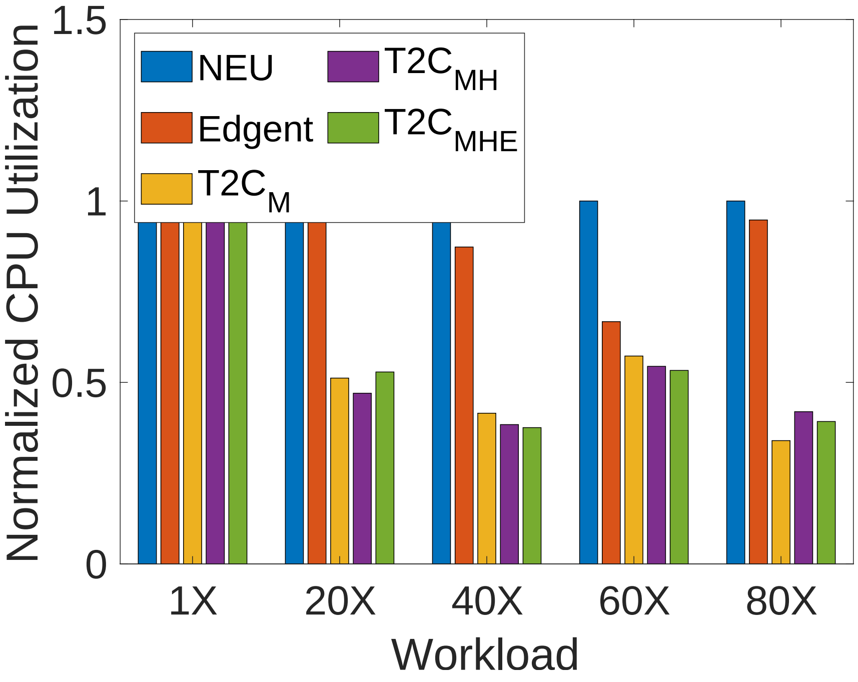
- Reconfigured mobile apps during cloud offloading to maximize the execution speed [CC’14]
- Investigated if reconfiguration of DNN is useful [CLOUD’21]
- Reduced service downtime of reconfiguration [IC2E’21]

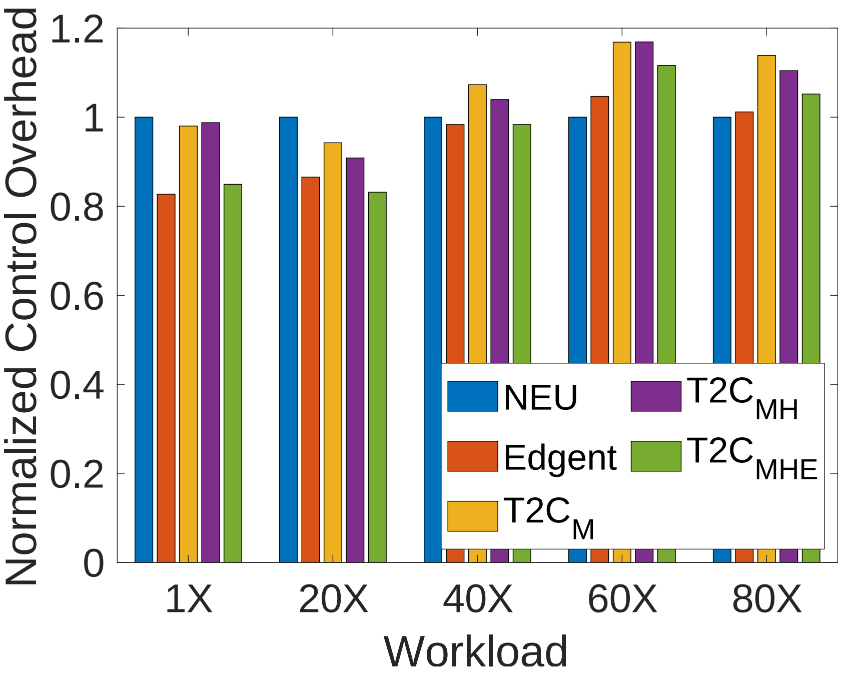
Reconfiguration of DNNs has not been integrated in a deployment system

[CC’14] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri. Run time application repartitioning in dynamic mobile cloud environments. IEEE Transactions on Cloud Computing, 4(3):336–348, September 2014.
[CLOUD’21] F. McNamee, S. Dustdar, P. Kilpatrick, W. Shi, I. Spence, and B. Varghese. The case for adaptive deep neural networks in edge computing. In Proc. of IEEE International Conference on Cloud Computing (CLOUD’21), pages 43–52, Chicago, IL, September 2021.
[IC2E’21] A. Majeed, P. Kilpatrick, I. Spence, and B. Varghese. NEUKONFIG:Reducing edge service downtime when repartitioning DNNs. In in Proc. of IEEE International Conference on Cloud Engineering (IC2E’21), pages 118–125, San Francisco, CA, October 2021.

Overhead under Different Workload Compared to NEU

- T2C algorithms do not incur significantly more overhead when workload is heavier



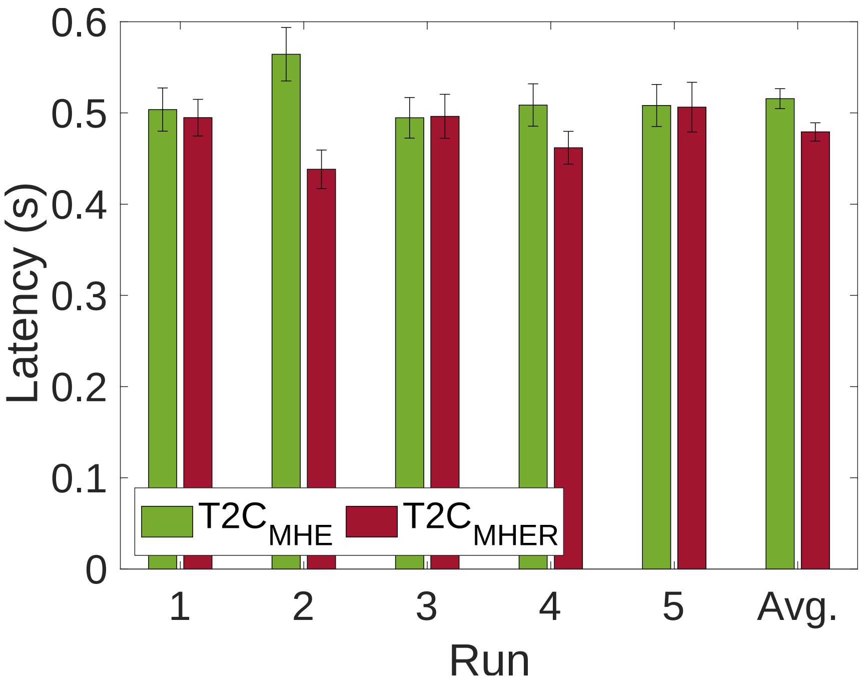


Our system scales well under heavy workload

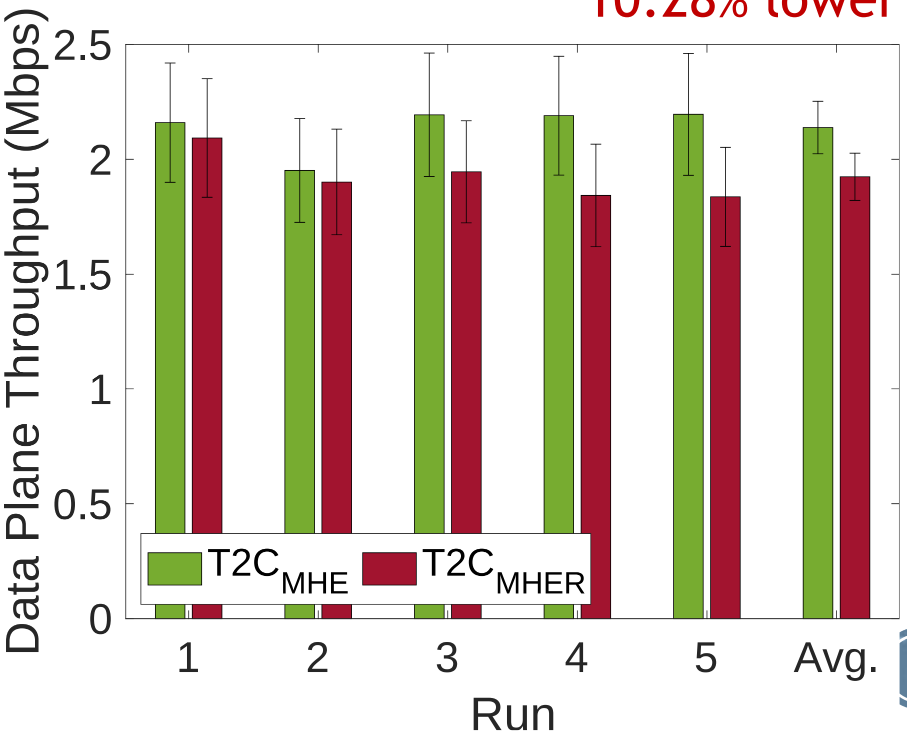
Latency Reduction by Reconfiguration

- T2C$_{MHER}$ changes the partition points to reduce latency

7.67% lower

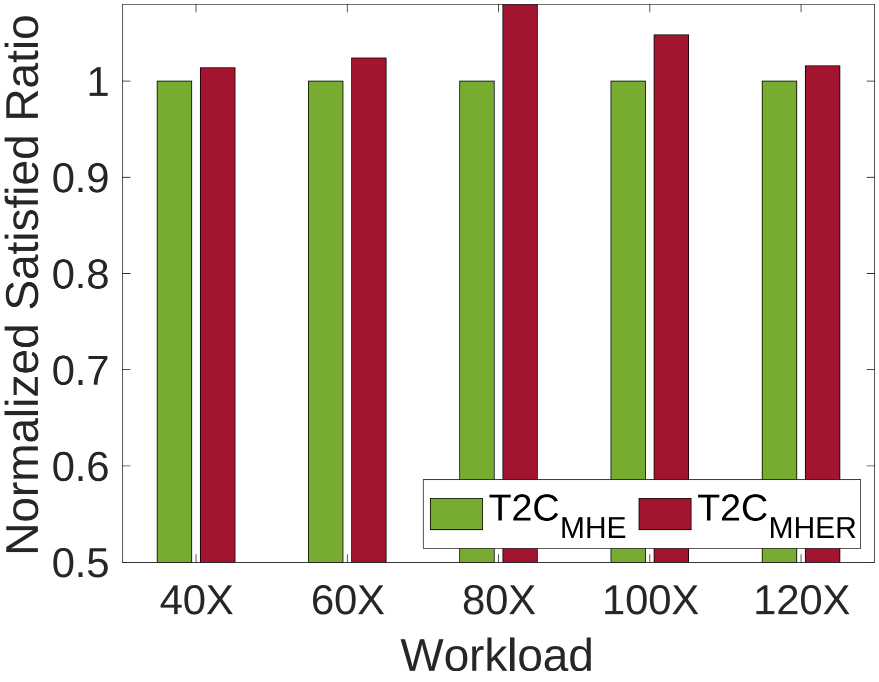


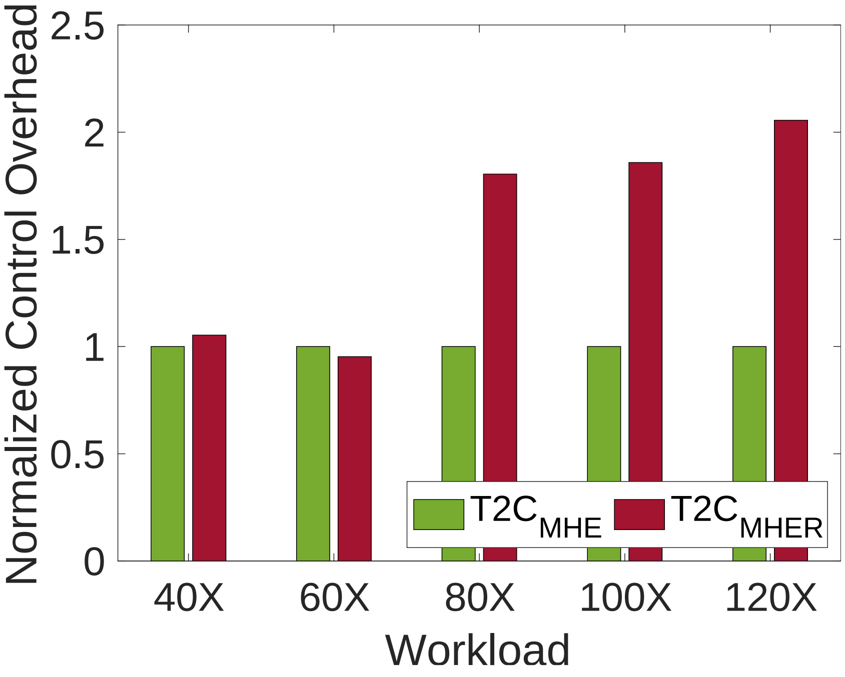
10.28% lower



Performance under Different Workload

- T2C$_{MHER}$ always achieves better satisfied ratio under different workload
- Higher control overhead but still < 1.5 Mbps


Normalized Satisfied Ratio
40X
60X
80X
100X
120X
Workload
🟩 T2C$_{MHE}$
🟥 T2C$_{MHER}$


Normalized Control Overhead
40X
60X
80X
100X
120X
Workload
🟩 T2C$_{MHE}$
🟥 T2C$_{MHER}$