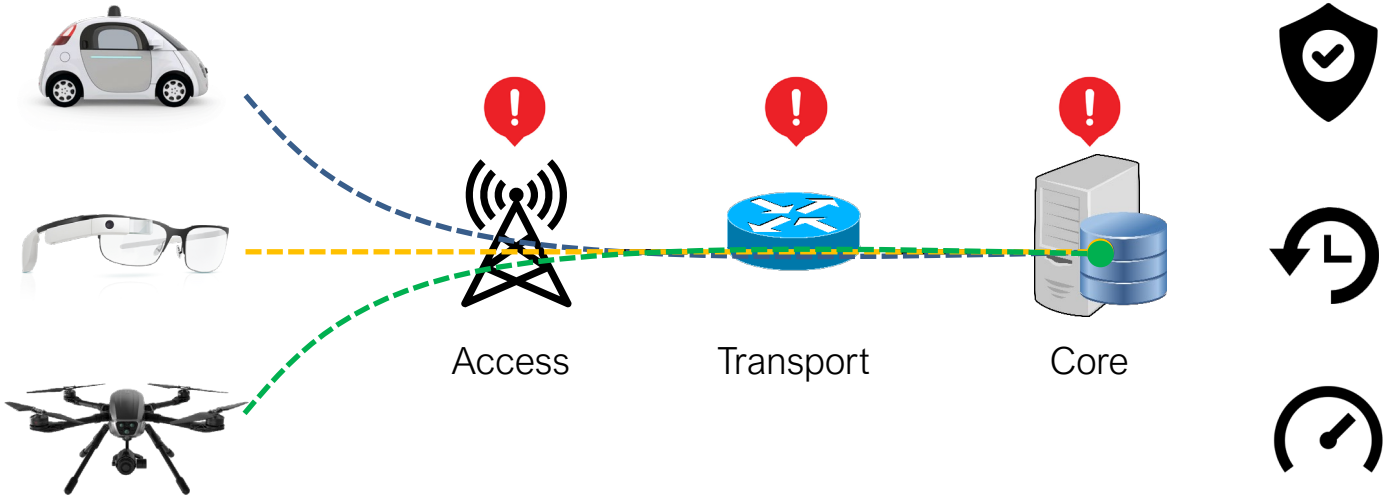
OnSlicing: Online End-to-End Network Slicing with Reinforcement Learning

Qiang Liu, University of Nebraska-Lincoln
Nakjung Choi, Nokia Bell Labs
Tao Han, New Jersey Institute of Technology

IN OUR GRIT, OUR GLORY™

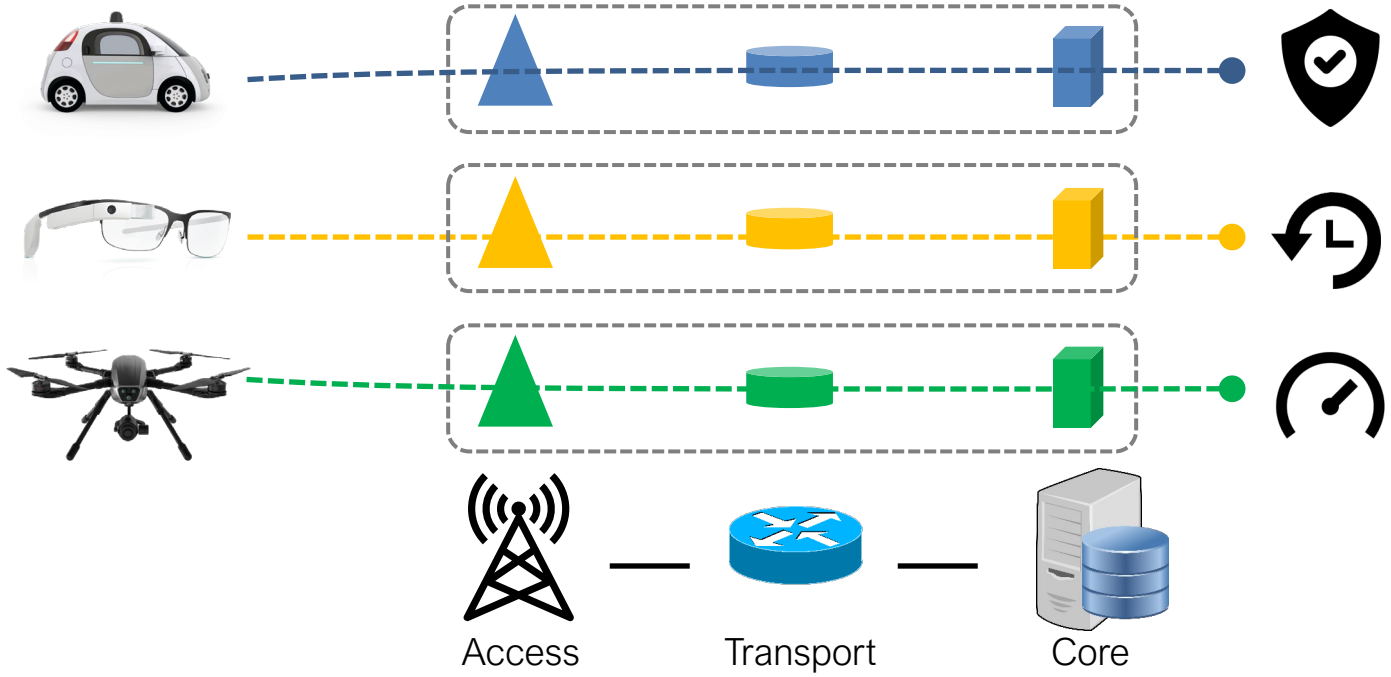
End-to-End Applications

- Resource competition degrades end-to-end performance of applications
  - Independent optimization in RAN, Transport, Core, MEC



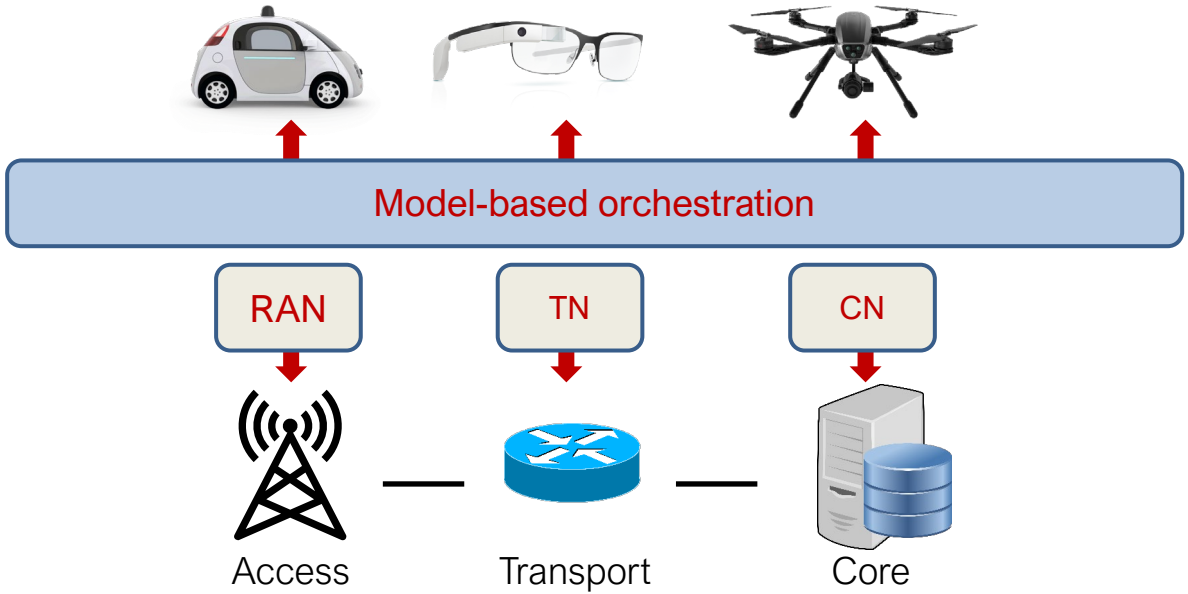
End-to-End Network Slicing

- Enabling dedicated end-to-end slice for each application
  - Performance isolation, SLA guarantee
  - Customization in performance, function, security, etc.



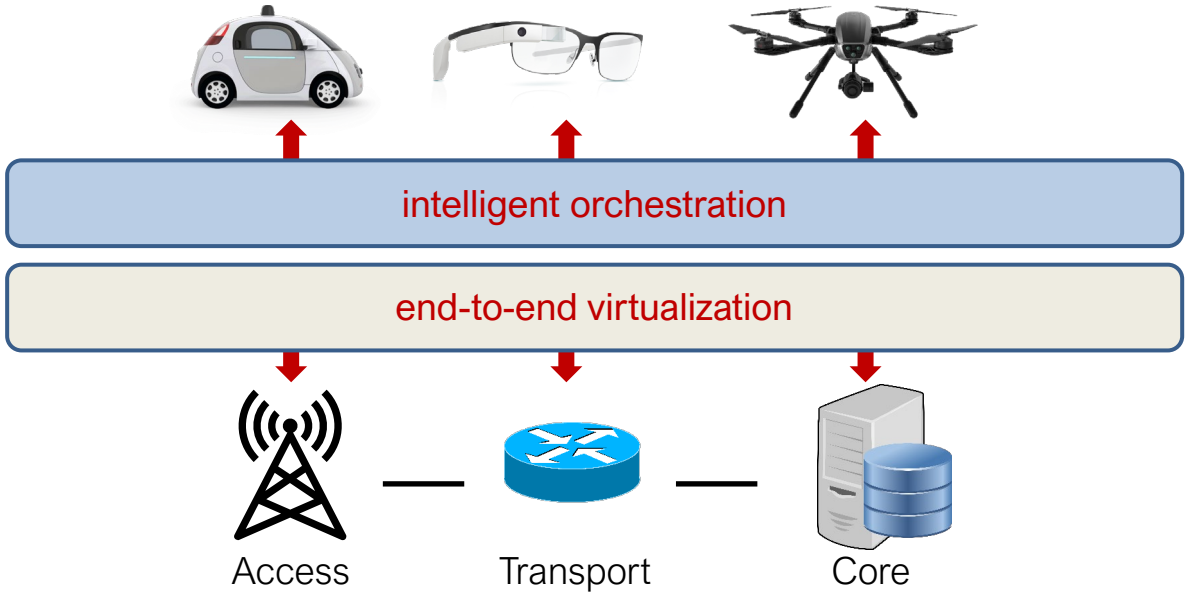
State-of-the-Art

- Individual domain virtualization
  - Fail to optimize end-to-end performances
- Model-based resource orchestration
  - Fail to capture complex correlations and network dynamics



OnSlicing

- The first online end-to-end network slicing system
  - End-to-end virtualization platform, i.e., RAN, TN, CN, Edge
  - Intelligent model-free cross-domain resource orchestration (multi-domain, multi-slice)



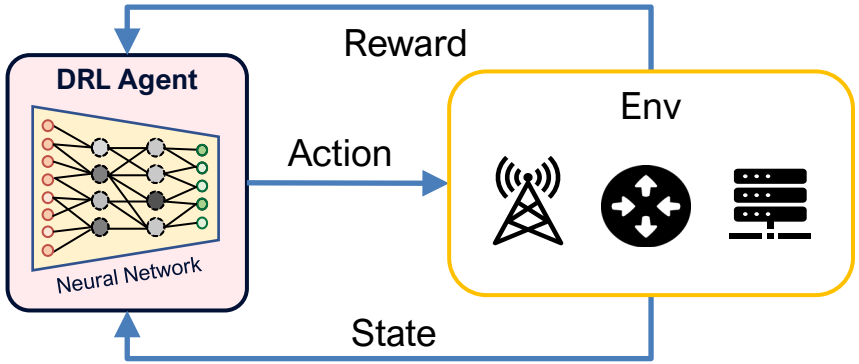
Resource Orchestration

❖ Objective

- Minimize cross-domain resource usage
- Satisfy the end-to-end performance requirements of slices and resource capacity of infrastructures
- **Degree of freedom**: virtual resources in multiple domains

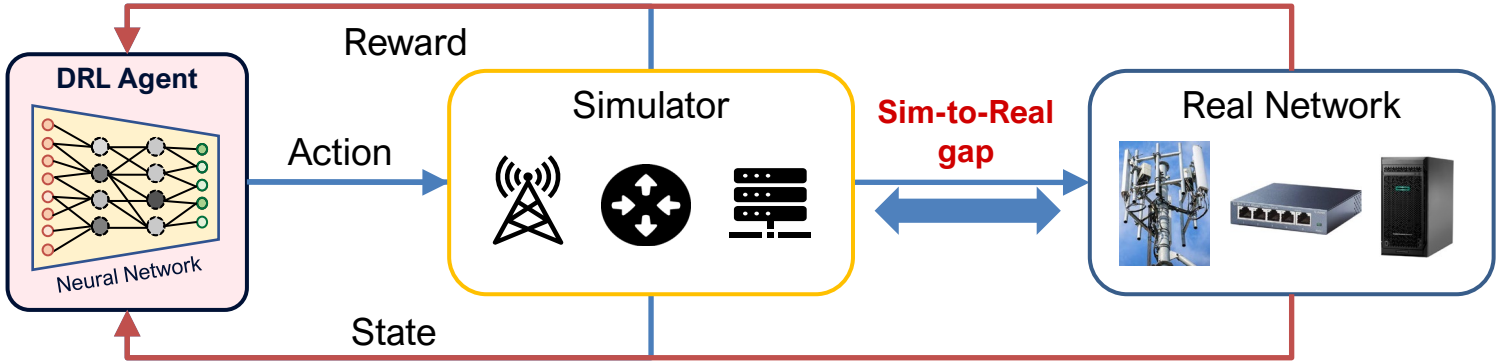
❖ Deep Reinforcement Learning

- Model-free approach, learn to orchestrate automatically
- Handle high-dim network states and complex correlations



Simulation-to-Reality Discrepancy

- **Offline-train and online deploy**
  - Train DRL agent in offline simulator, deploy DRL agent in online real networks
- **Simulation-to-reality discrepancy**
  - Offline simulator fails to fully represent the complex real networks
  - Performance degrades if the offline trained policy is directly applied to manage real networks
- **Online Deep Reinforcement Learning**
  - Online learn to orchestrate from real networks



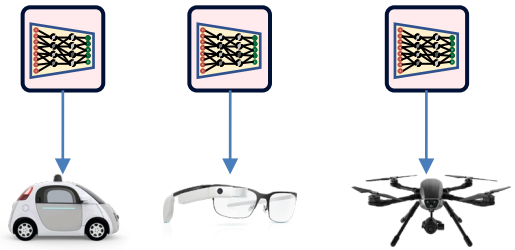
* The simulation-to-reality discrepancy has been identified in multiple fields, e.g., robots, network.

Challenges

- **Performance assurance of slices**
  - Unawareness of constraints in policy update
  - Random action exploration of DRL violates slice SLAs
  - Unconstrained DRL exploitation
- **Scalability in distributed infrastructures**
  - Constraints in infrastructures, e.g., resource capacity
- **Poor policy at early learning stage**
  - DRL agents need lots of experiences to achieve acceptable performance

DRL Agent Design

- Individualized DRL agent for each slice
  - Reduce problem complexity
  - Efficient to learn inherent slice characteristics



- Smooth policy improvement
  - Constrain the maximum policy update w/ proximal policy optimization (PPO)
- State space
  - Avg. channel condition, radio resource usage, server workload, cumulative cost, etc.
- Action space
  - **RAN:** UL/DL BW, MCS offset, Sche. Alg., **TN:** BW, Path, **CN/Edge:** CPU/RAM
- Reward function
  - Negative total resource usage
- Cost function
  - Violation of slice SLA

Learning with near-zero violations

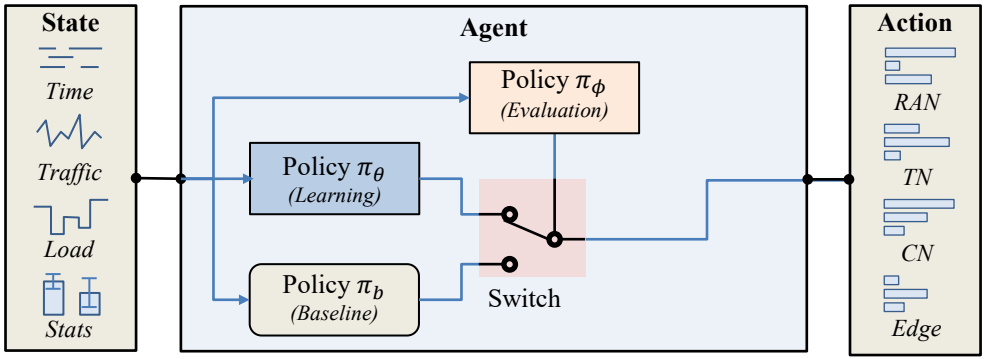
❖ Constraint-aware policy update method

- Dynamic incorporate slice SLA into objective
- Lagrangian primal-dual method

$$\max_{\pi_\theta} \quad \mathbb{E}_{\pi_\theta}\left[\sum_{t\in\mathcal{T}} r(\mathbf{s}_t, \mathbf{a}_t)\right]$$
$$s.t. \quad \mathbb{E}_{\pi_\theta}\left[\frac{1}{T}\sum_{t\in\mathcal{T}} c(\mathbf{s}_t, \mathbf{a}_t)\right] \le C_{\max},$$

❖ Proactive baseline switching mechanism

- Switch to a baseline policy if predicted to violate slice SLA
- Predict the statistics of cost value with variational inference (VI)

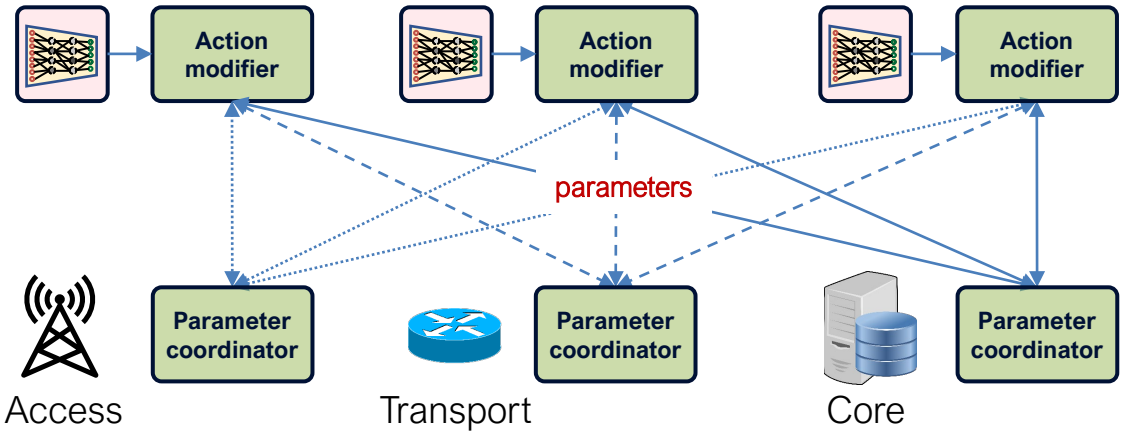


Learning in Distributed Networks

- Satisfy resource capacity in infrastructures
  - The DRL agent of slice makes decision independently

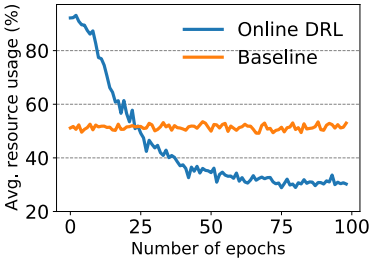
$$\min_{\hat{\mathbf{a}}^i} \sum_{i\in\mathcal{I}} \{|\hat{\mathbf{a}}_t^i - \mathbf{a}_t^i|_2^2 + c(\mathbf{s}_t^i, \hat{\mathbf{a}}_t^i)\}$$
$$s.t. \sum_{i\in\mathcal{I}} \hat{\mathbf{a}}_t^{i,k} \le L_{\max}^k, \forall k \in \mathcal{K},$$

- Action modification in each agent
  - Modify the action according to coordinating parameters
- Parameter coordination in each infrastructure
  - Update coordinating parameters based on resource usage

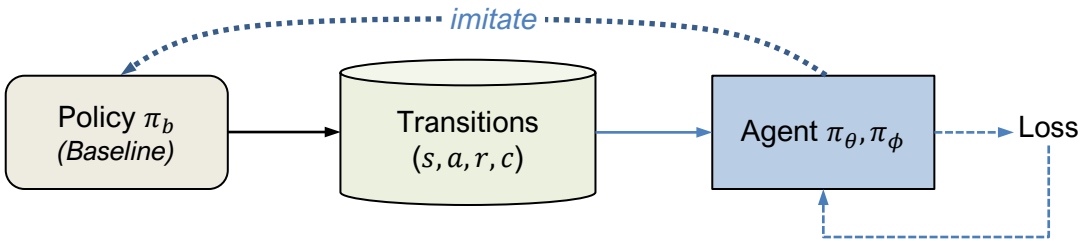


Learning from Baseline

- **Poor policy at early learning stage**
  - Online learning from scratch is expensive and risky



- **Behavior cloning**
  - Offline imitate the baseline policy before online learning phase
  - Minimizing the action differences with supervised learning

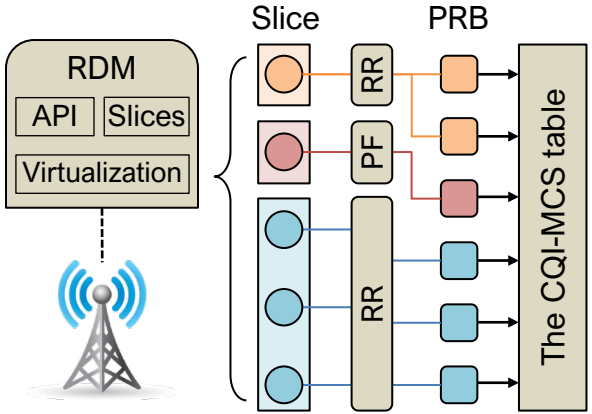


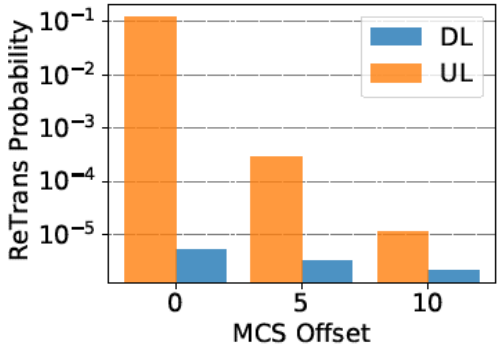
Virtualization - RAN

- ## Limited features in FlexRAN
  - Allocating physical resource blocks (PRBs) in MAC layer
  - Controls in data rate but not reliability
- ## Radio domain manager (RDM)
  - Based on FlexRAN framework
  - New CQI-MCS mapping table* for each slice

Table 7.2.3-1: 4-bit CQI Table

CQI index	modulation	code rate x 1024	efficiency
0	out of range		
1	QPSK	78	0.1523
2	QPSK	120	0.2344
3	QPSK	193	0.3770
4	QPSK	308	0.6016
5	QPSK	449	0.8770
6	QPSK	602	1.1758
7	16QAM	378	1.4766
8	16QAM	490	1.9141
9	16QAM	616	2.4063
10	64QAM	466	2.7305
11	64QAM	567	3.3223
12	64QAM	666	3.9023
13	64QAM	772	4.5234
14	64QAM	873	5.1152
15	64QAM	948	5.5547

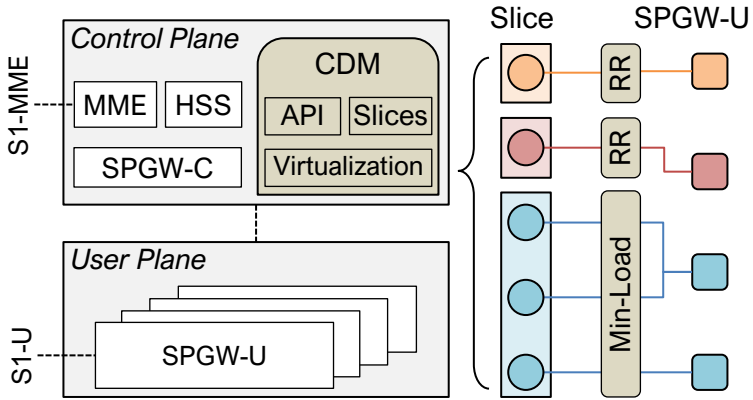




*The bit error rate (BER) is reduced if adopting a lower modulation and coding scheme under the same power allocation

Virtualization - CN

- **Unsupported virtualization**
  - All traffics share the data plane
  - No isolation for different slices
- **Core domain manager (CDM)**
  - Isolated data plane (SPGW-U/UPF) with CUPS architecture
  - Associate containerized a SPGW-U pool to each slice
  - Determining association upon the initial attachment request of slice users



* The virtualization of transport network and edge computing are accomplished based on OpenFlow and Docker containers, respectively.

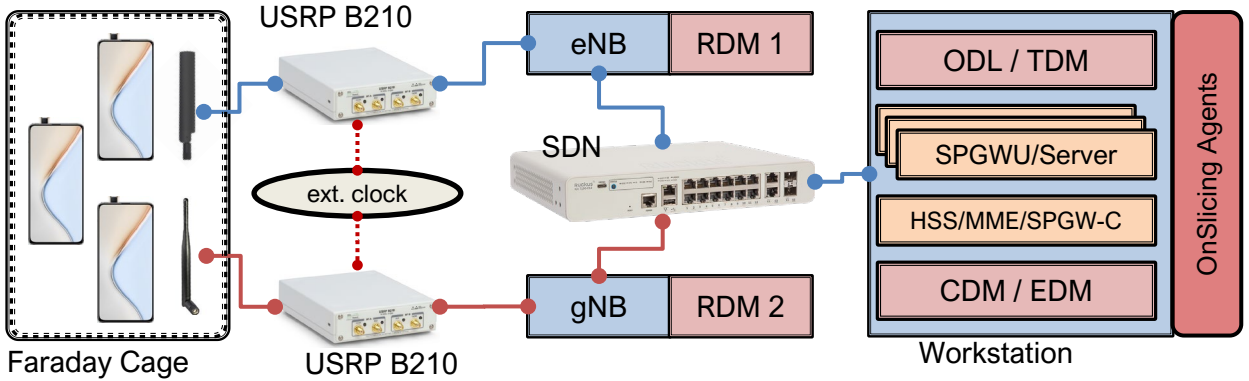
System Implementation

❖ Testbed

Slice: 3 slices (each is emulated a phone)	**Agent**: PyTorch 1.5 (128x64x32)
RAN: OpenAirInterface w/ USRP (LTE & 5G NSA)	**TN**: OpenDayLight w/ OpenFlow SDN switch
CN: OpenAir-CN w/ CUPS	**Edge:** Dockers collocated with SPGW-U

❖ Mobile application w/ traffic traces

- **MAR**: client offloads 540p images, server detects features and sends results back (latency)
- **HVS**: server streams 1080p video (FPS)
- **RDC**: server controls IoT devices (reliability)



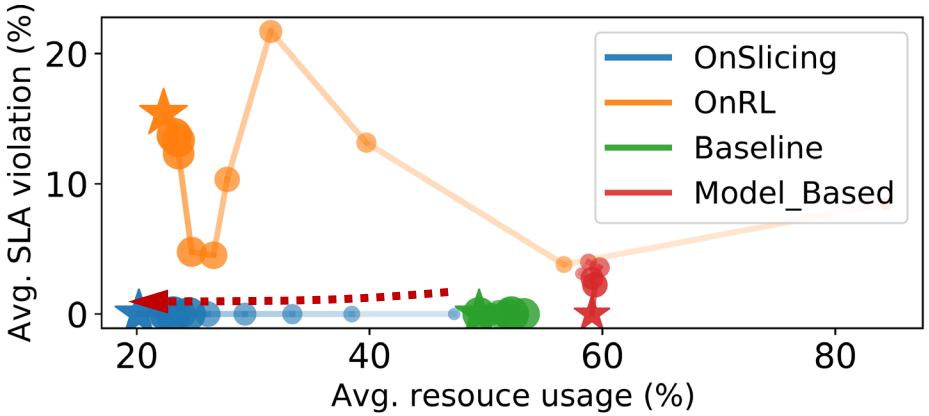
Overall Performance

❖ Comparison

- **Model_Based**: build the problem with approx. math models
- **Baseline**: regress the correlation w/ sk-learn toolbox
- **OnRL**: original for video telephony, enhanced to assure SLA w/ reward shaping

❖ Learning trajectories

- OnSlicing continuously learns from real networks
- OnSlicing achieves the best performance, i.e., resource usage and SLA violation



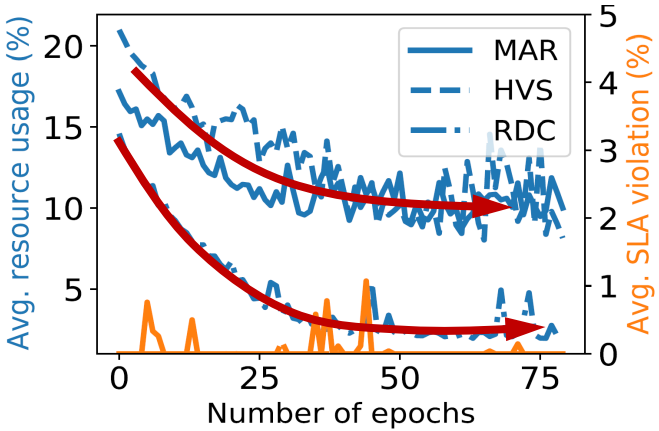
Learning Performance

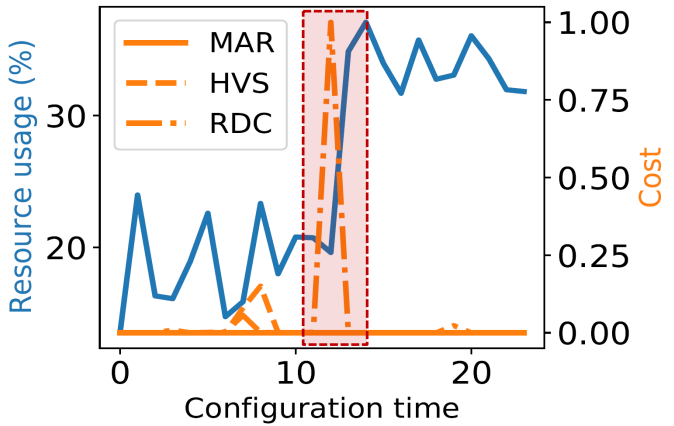
❖ Online safe learning

- OnSlicing learns to reduce resource usage smoothly
- OnSlicing learns the intrinsic slice characteristics
- OnSlicing has near-zero violation throughout learning
- OnSlicing proactively switch to baseline for SLA guarantee

Method	Avg. res. usage (%)	Avg. SLA viol. (%)
OnSlicing	29.07	0.06
OnSlicing-NE	30.81	0.33
OnSlicing-NB	29.64	2.94
OnSlicing Est. Noise	52.91	1.03

Table 2: Avg. performance of baseline switching methods



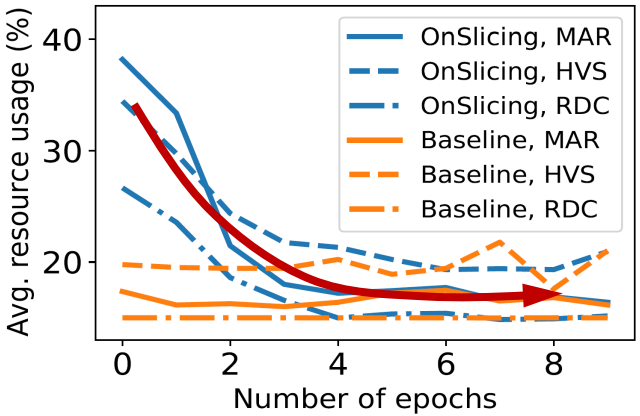


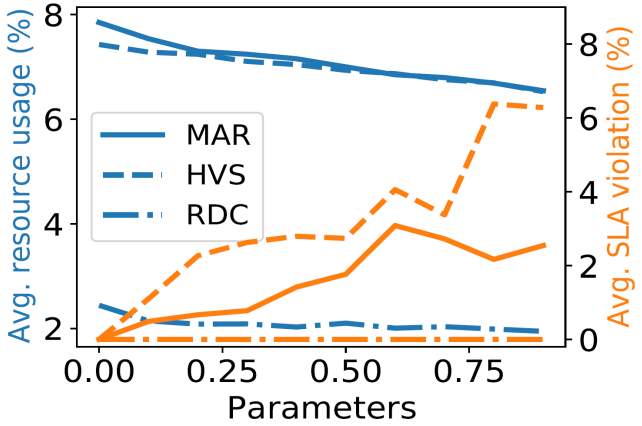
Learning Performance

- **Offline imitate learning**
  - OnSlicing quickly imitate the baseline
- **Online distributed coordination**
  - OnSlicing reacts to coordinating parameters
  - Coordination is essential to maintain slice SLA

Methods	Usage (%)	Viol. (%)	Interact num.
OnSlicing	20.2 ± 0.23	0.00 ± 0.00	1.83 ± 0.61
OnSlicing-projection	18.2 ± 0.50	3.66 ± 2.49	1.00 ± 0.00
OnSlicing Md. Noise	23.8 ± 1.56	2.57 ± 1.66	2.16 ± 1.08

Table 3: Performance of action modifications





Summary

- End-to-end slicing is necessitated to assure the diverse end-to-end performance of applications
- We proposed OnSlicing, the first online end-to-end network slicing system
- OnSlicing addressed the practical challenges of online DRL, i.e., performance assurance, scalability in distributed networks and poor policy at early stage.
- OnSlicing enables the efficient resource virtualization in RAN, TN, CN and Edge.
- We prototype OnSlicing in end-to-end testbed and evaluated its performance w/ extensive experiments

Thank you!
Q&A

Qiang Liu
Assistant Professor
School of Computing
University of Nebraska–Lincoln
qiang.liu@unl.edu
https://cse.unl.edu/~qliu/

Intention Blank

Performance in 5G NSA

- 5G NSA settings
  - Static MCS 9 for stability
  - 40 MHz, B78, TDD
- 5G promise better performance
  - Lower resource usage and zero SLA violation

Networks	Avg. res. usage (%)	Avg. SLA violation (%)
5G NR	43.5 ± 3.27	0.00 ± 0.00
4G LTE	45.9 ± 4.48	0.66 ± 1.42

Table 4: OnSlicing performance in 4G LTE and 5G NSA