Atlas: Automate Online Service Configuration in Network Slicing

Qiang Liu, University of Nebraska-Lincoln
Nak Jung Choi, Nokia Bell Labs
Tao Han, New Jersey Institute of Technology
Emerging Applications

- Resource competition **degrades** end-to-end performance
  - independent optimization, e.g., RAN and TN, fails in guaranteed performance
Network Slicing

- Enable customized **end-to-end slice** for each application
  - performance and functional isolation, SLA guarantee
  - customization in performance, function, security, etc.
Configure individual slice settings to maintain SLA
- for example, cross-domain resources, attributes*
- High-dim contextual states, e.g., traffic, users
- long configuration interval, e.g., hours (non-markov)

*Generic Network Slice Template, V 7.0, GSM Association, June 2022
State-of-the-Art

- **Offline approaches**
  - design the policy in offline environments, e.g., simulator or dataset
  - offline approaches \([1, 2]\) suffer simulation-to-reality discrepancy
  - the discrepancy between offline simulators and real-world networks

---

[1] Marquez, C., et. al. How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency. Mobicom 2018 (pp. 191-206).
[2] Salvat, J.X., et. al. Overbooking network slices through yield-driven end-to-end orchestration. CoNEXT 2018 (pp. 353-365).
Online approaches

- learn the policy via interacting with real-world networks
- online ML methods [3] suffer safety and sample-efficiency issue
- **sample-efficiency**: long configuration interval in real-world networks
- **safety**: unpredictable configuration actions from DNN-parameterized policies

[3] Shi, J., et. al. Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation. *NSDI* 2021 (pp. 887-901).
The first integrated offline-online network slicing

- Atlas automates service configuration of individual slices
- Atlas achieves safe and sample-efficient learn-to-configure in three integrated stages
  - stage 1: learning-based simulator, for reducing sim-to-real discrepancy
  - stage 2: offline training, for training an offline policy
  - stage 3: online learning, for learning the online policy
Stage 1

- Learning-based Simulator
  - **objective**: automatically reduce the simulation-to-reality discrepancy
  - **action**: adjust the simulation parameters, e.g., base pathloss
  - **rationale**: these parameters might not accurate enough

Simulation Parameters

- Online Collection
- Slice Performance
- Discrepancy Calculation
- Simulation Parameters
Stage 1

- **Learning-based Simulator**
  - **problem**: minimize KL divergence between simulation and system measurement
  - **challenge**: unknown correlation between KL divergence and high-dim simulation parameters
  - **solution**: new Bayesian learning method
    - scalable Bayesian neural network
    - parallel Thompson sampling
Stage 2

- **Offline Training**
  - **objective**: offline train a policy in the augmented simulator
  - **problem**: minimize resource usage under requirement of percentile QoE
  - **challenge**: unknown correlation between slice QoE and configuration parameters
  - **solution**: constraint-aware method and Bayesian learning method
Stage 3

Online Learning

- **objective**: online learn the policy in real-world networks
- **rationale**: resolve the sim-to-real discrepancy eventually
- **problem**: minimize resource usage under requirement of percentile QoE
Online Learning

- **challenge**: assure safety (SLA violation) under limited online transitions
- **solution**:
  - sample-efficient GP model to learn sim-to-real gap only
  - conservative acquisition function with regret bound
  - hybrid multiplier update with both offline and online transitions
System Implementation

- **Testbed**
  
  | Role | Description |
  |------|-------------|
  | **User** | OnePlus 9 5G |
  | **Agent** | PyTorch 1.5 (128x64x32) |
  | **RAN** | OpenAirInterface w/ USRP (LTE B7) |
  | **TN** | OpenDayLight w/ SDN switch |
  | **CN** | OpenAir-CN w/ CUPS |
  | **Edge** | Dockers collocated with SPGW-U |

- **Virtualization**
  
  - **RAN**: FlexRAN (exclusive PRB assignment) + customized MCS offset
  - **TN**: OpenFlow with configurable bandwidth via “meter”
  - **CN**: isolated SPGW-U container per slice
  - **EN**: docker container via “docker update”

- **Applications**
  
  - Video analytics at the edge
  - send 540p image to edge server
  - the server run ORB to extract features
  - requirement: 300ms round-trip latency
Stage 1 Performance

- Atlas reduces sim-to-real discrepancy
  - obtains 81.2% discrepancy reduction under 0.12 parameter distance
  - more than 24.5% reduction than existing Bayesian optimization method (GP)

| Methods       | Sim-to-Real Discrepancy | Parameter distance | Best simulation parameters |
|---------------|-------------------------|--------------------|---------------------------|
| Original Simulator | 1.38                    | 0                  | [38.57, 5.0, 9.0, 0.0, 0.0, 0.0, 0.0] |
| Aug. Simulator, GP | 0.31                    | 0.16               | [38.57, 1.44, 7.48, 5.07, 9.23, 6.02, 6.47] |
| Aug. Simulator, Ours | 0.26                    | 0.12               | [38.76, 0.68, 8.93, 5.03, 8.93, 2.16, 3.10] |

Table 4: Details of offline learning-based simulator
Stage 2 Performance

- Atlas trains the policy with reduced resource usage
  - obtains up to 47.5% usage reduction than existing solutions
  - better Pareto boundary performance

Shi, J., Sha, M. and Peng, X., 2021. Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation. NSDI 21.
Stage 3 Performance

- Atlas reduces usage and QoE regret
  - obtains up to 63.9% reduction on the regret of resource usage
  - obtains up to 85.7% reduction on the regret of slice QoE
  - results show the necessity of integrating three stages

Shi, J., Sha, M. and Peng, X., 2021. Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation. NSDI 21.
End-to-end slicing is necessitated to assure diversified performance of slices.

We proposed Atlas, the first integrated offline-online network slicing system that automates the service configuration of individual slices.

Atlas addressed practical challenges of online machine learning, i.e., safety and sample-efficiency, by designing three interrelated stages.

We prototype Atlas in end-to-end slicing testbed with extensive performance evaluation.

GitHub: https://github.com/int-unl/Atlas.git
Qiang Liu
Assistant Professor
School of Computing
University of Nebraska–Lincoln
qiang.liu@unl.edu
https://cse.unl.edu/~qliu/
Simulation and Configuration Space

- **Simulation Space**
  - selected according to its impact on the sim-to-real discrepancy

- **Configuration Space**
  - selected according to its impact on the performance of slice users

- Atlas can handle more simulation and configuration space

| Configuration    | Meaning                        | Range   |
|------------------|--------------------------------|---------|
| bandwidth_UL     | maximum uplink PRBs            | [0, 50] |
| bandwidth_DL     | maximum downlink PRBs          | [0, 50] |
| mcs_offset_UL    | uplink MCS offset [24]         | [0, 10] |
| mcs_offset_DL    | downlink MCS offset [24]       | [0, 10] |
| backhaul_bw      | transport bandwidth (Mbps)     | [0, 100]|
| cpu_ratio        | CPU ratio of docker            | [0, 1.0]| Table 2: Network configuration space

| Parameters       | Meaning                                           |
|------------------|---------------------------------------------------|
| baseline_loss    | base loss in pathloss model (dBm)                |
| enb_noise_figure | noise by non-ideal transceivers (dBm)            |
| ue_noise_figure  | noise by non-ideal transceivers (dBm)            |
| backhaul_bw      | additional transport bandwidth (Mbps)            |
| backhaul_delay   | additional transport delay (ms)                  |
| compute_time     | additional server compute time (ms)              |
| loading_time     | additional loading time in UE (ms)               |

Table 3: Simulation parameter space