Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach

He, Ying, F. Richard Yu, Nan Zhao, Victor C. M. Leung and Hongxi Yin. “Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach.” *IEEE Communications Magazine* 55 (2017): 31-37.
doi: 10.1109/MCOM.2017.1700246

Review by Tugrul Yatagan

yatagan@itu.edu.tr

April 2018

Outline

1. Introduction
2. System Description
3. Proposed Framework
4. Deep Reinforcement Learning
5. Problem Formulation
6. Conclusion

Introduction

- Innovations in networking, caching and computing for smart cities.
- The basic principle of SDN is to separate the control plane from the data plane, enabling the ability to program the network via a centralized software defined controller with a global view of the network.
- Network functions virtualization (NFV) enables abstraction of physical networking resources and flexible sharing of resources by multiple users through isolating each other.
- In-network caching;
  - Efficiently reduce duplicate content transmission in networks.
- Cloud computing;
  - Enable access to a shared pool of computing resources.
- An integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve the performance of applications for smart cities is proposed.

Introduction

- Framework of integrated networking, caching, and computing with software-defined and virtualized networks for smart cities.
- Big data deep reinforcement learning approach is proposed for resource allocation policy.
- Infrastructure is abstracted and virtualized into multiple customized virtual networks. (energy, health, vehicular, IoT, etc.)
- Variety of smart city applications can be provided;
  - Healthcare assistance.
  - Security and safety.
  - Intelligent transportation.
  - Traffic monitoring.
  - Waste management.

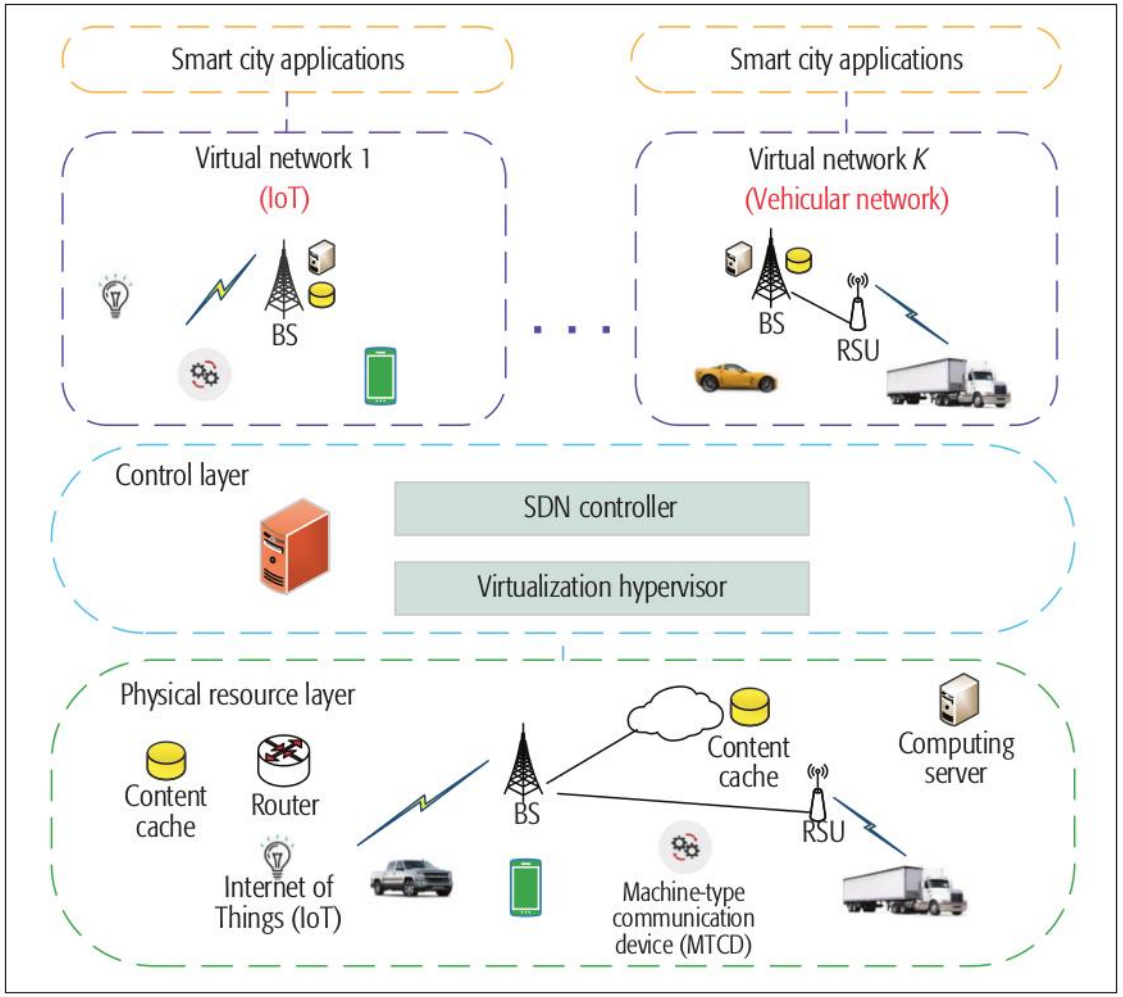

Figure 1. A framework of integrated networking, caching, and computing for smart cities.

System Description

Software Defined and Virtualized Networks

- SDN; flexibility, programmability, and centralized controller.
  - Multi-radio environment;
    - Optimize channel allocation and network selection and reduce interference.
  - Multi-hop environments;
    - Improve packet routing decisions and mobility.
- Wireless network virtualization;
  - Management of network architecture and network resources.
  - A common physical wireless network can be virtualized into several virtual ones.

System Description

Information Centric Smart Cities

- Huge amounts of information-rich and safety-critical data.
- Poor quality wireless links and high mobility in some applications.
- In-network caching;
  - Reduce duplicate content transmission in networks.

Smart Cities with Mobile Edge Computing

- Data centers are usually far away from the end user.
- Transmitting a large amount of data from the device to the cloud may not be feasible or economical.
- Low latency and location awareness are two obvious characteristics of fog computing.
- Edge computing;
  - Deploy computing resources closer to end devices.

Proposed Framework

Framework

- Abstract, allocate, and optimize; networking, caching and computing resources for different smart city applications.
- Physical wireless network can be virtualized into several virtual ones.
- Each virtual network includes base stations (BSs), roadside units (RSUs), access points (APs), mobile edge computing (MEC) servers and content caches.
- Multiple virtual networks with different network services could be embedded in the same platform.
- Controller can manage the network nodes equipped with caching and computing capacities when user access the wireless network.
- Some smart city applications want to access or process the data, the contents may hit the in-network cache which decrease latency and decrease traffic redundancy.

Proposed Framework

Smart City Use Cases

Joe visits a city and finds an interesting monument.

- Downlink;
  - With his augmented reality (AR) glasses, Joe notices that there is an introductory video about this monument.
  - He issues a video content request.
  - The virtual BS will check whether or not its associated cache has the requested content.
  - If cache miss, virtual BS will extract the current video content and construct a computation task.
  - Virtual BS will transmit the task to the MEC server to execute.
- Uplink;
  - Joe takes a video of it, puts some comments on it, and streams it to a remote server via a virtual BS.
  - The virtual BS extracts the video content, and estimates the possibility that other people want to see this video.
  - If the cached version does not match the new user's device, transcoding will be performed by the MEC server.

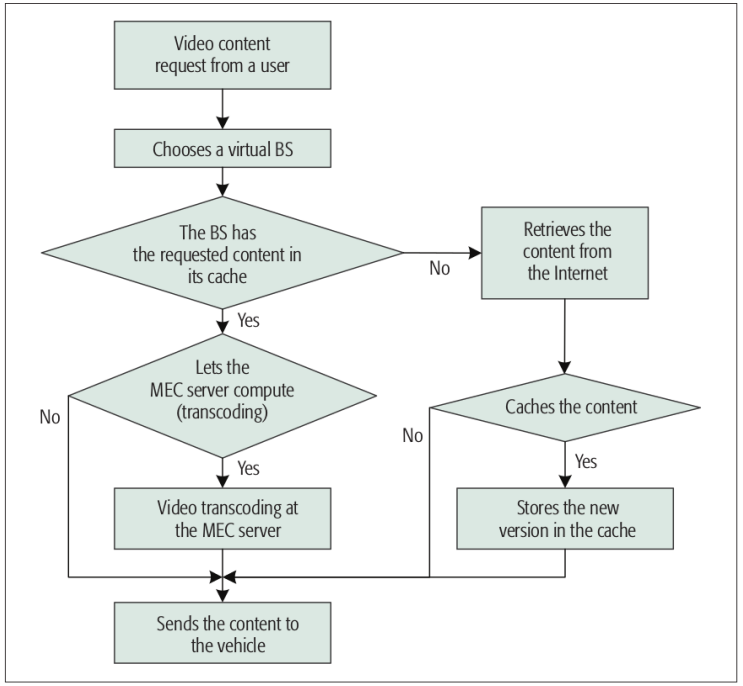

Figure 2. The procedure of tourism streaming services in a smart city network with mobile edge computing and caching.

Deep Reinforcement Learning

- Networking, caching, and computing jointly increases the complexity of the system.
- Difficult to solve the resource allocation problem using traditional approaches.
- Deep reinforcement learning approach is used to solve the optimization problem.

Reinforcement Learning

- Branch of machine learning where agent learns to take actions that would yield the most reward by interacting with the environment.
- It cannot learn from samples provided by an experienced external supervisor.
- It has to operate based on its own experience despite facing significant uncertainty about the environment.

Deep Q-Learning

- The agent that uses a neural network to represent Q-function is called the Q-network.
- $Q(x, a; \theta)$. The parameter θ stands for the weights of the neural network.
- Q-network is trained by updating θ at each iteration.

Problem Formulation

- There are K BSs, m MEC servers, and C content caches that are virtualized and managed to provide services for smart cities.
- BSs' downlink channel conditions, the MEC servers' computation abilities, and the caches' states are all dynamically changing.
- Since we consider realistic scenarios;
  - Controller faces a large amount of system states.
  - It has to make a decision on which virtualized resources will be assigned to a specific user according to the system's current state.
- It is difficult to solve this complicated task using a traditional methods.

Problem Formulation

- Deep Q-learning is capable of receiving complex high-dimensional data as input, and yielding an optimal action for each input data.
- Controller sends the constructed system state (status from each BS, MEC server, and content cache) to the agent.
- The deep Q agent;
  - Tries to take a random action to explore the unknown environment with a probability E.
  - Maximizes the reward based on the knowledge already obtained with a probability 1 – E.
- The agent has to decide;
  - Which BS is assigned to the user.
  - Whether or not the requested content should be cached in the BS.
  - Whether or not the computation task should be offloaded to the MEC server.
- The system reward is the MVNO's revenue;
  - MVNO rents the wireless spectrum and the backhaul bandwidth.
  - MVNO needs to pay for the usage of spectrum, computation fee.

Conclusion

- TensorFlow is used for simulations to implement deep reinforcement learning.
- With the increase of the number of episodes, the total utility increases until it reaches a relatively stable value.
- The total utility of different scenarios in the proposed scheme is very low at the beginning of the learning process.
- The total utility is lowest when virtualization is not considered compared to other scenarios.

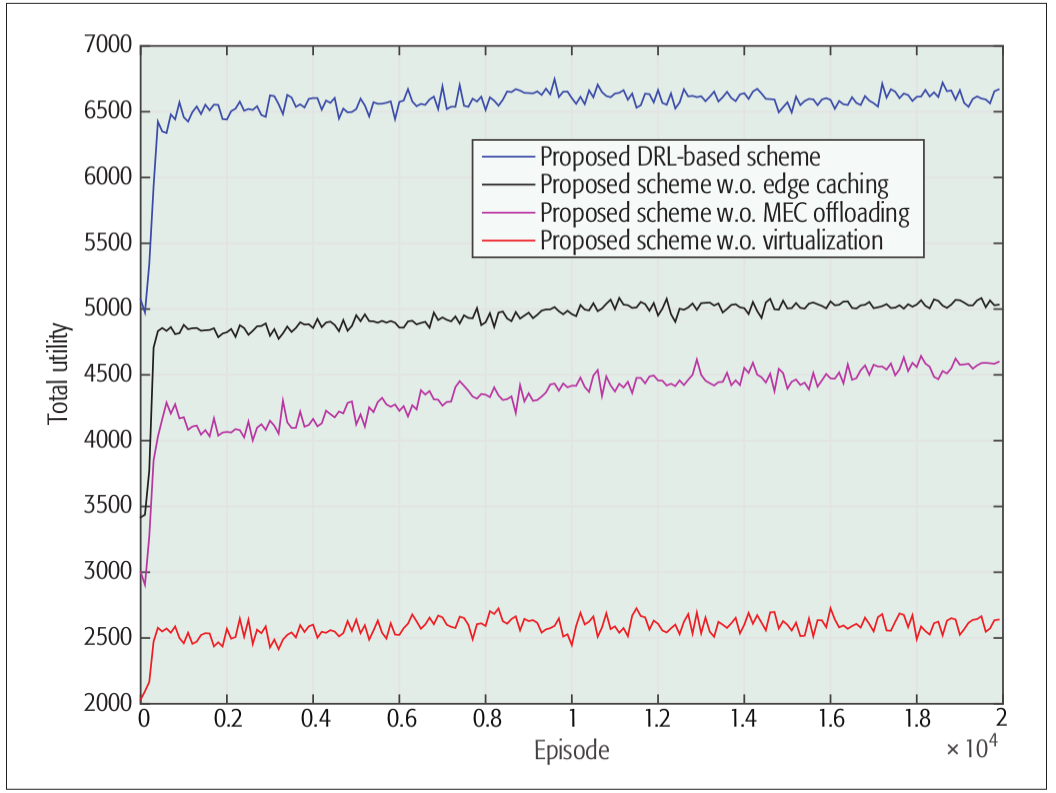

Figure 3. Convergence performance of different schemes.

Conclusion

Learning Rate

- Determines to what extent newly acquired information overrides old information.
  - 0 makes the agent learn nothing.
  - 1 makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities).
- Convergence is faster when the learning rate is 0.001 compared to the case when the learning rate is 0.0001 and 0.00001.
- Larger learning rate will result in local optimum instead of global optimum.

CPU Cycle

- The required number of CPU cycles has no effect on the total utility of the proposed scheme without MEC offloading.
- The total utility decreases exponentially with the increase of the required number of CPU cycles for each task.
  - Larger number of required CPU cycles will consume high computation energy and lower gain of computation utility.

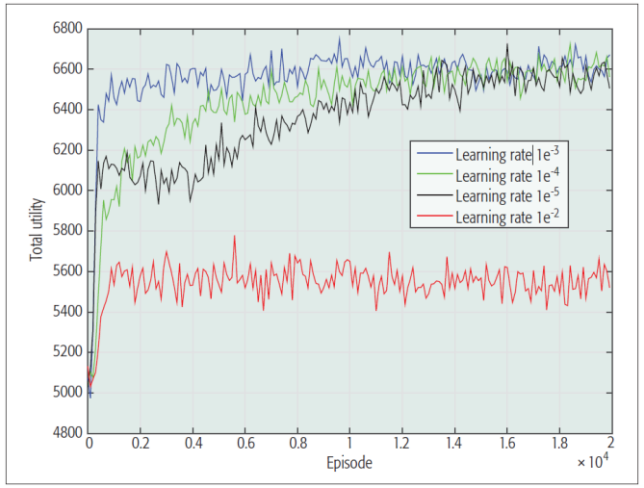

Figure 4. Convergence performance with different learning rates.

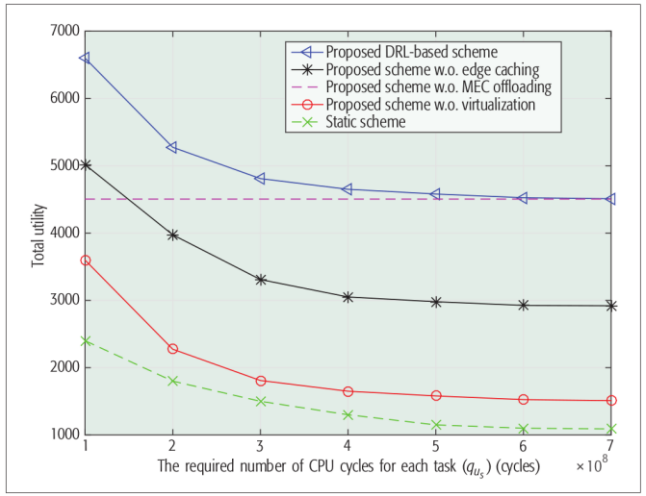

Figure 5. Effects of the required number of CPU cycles for each task.

Thank you for your time.

Any questions?