
Service Placement for Mobile Edge Computing using Pointer Networks

Expose: M. Sc


Name: Anagha Namboodiripad

Student-ID: 2609726

Course and study regulations: Distributed Software Systems

Supervisor: Florian Brandherm

18. Juli 2020



Inhaltsverzeichnis

1	Introduction	3
2	Motivation	5
3	Background	6
4	Background	8
5	Preliminaries	9
5.1	Sequence-to-Sequence Model	9
5.2	Attention mechanism	10
6	Pointer Networks	12
7	Models	13
8	Implementation	15
9	Related Work	17
10	Evaluation	18
11	Execution plan	19

1 Introduction

The advent of technology has made life easier to a large extent. Internet is one such decisive technology that emerged in the late 1960's and revolutionized the lives of people starting from the basic necessities to utmost luxuries. The capability of providing a plethora of mechanisms for information dissemination, and a medium that lets users interact with each other irrespective of their geographical location, led to never-ending advancements in the field of Internet technology. The popularity of smartphones and other mobile devices that avail the services of Internet, further boosted its growth, as users desired more enhanced computing services. The emergence of the Internet technology led to migration of communication and collaboration applications into the Internet 'cloud' [10], which in simple words is a set of servers performing a service. This is when the term 'cloud' was used for the first time. Clouds are servers that are located in data centers throughout the world and lets users access and store applications or files through the internet. Cloud computing is the technology of offering computing capability in the form of services such as storage, infrastructure, software, data etc. through the cloud for the users. This facilitates ease in terms of increased storage as clouds offer large storage capacities, reduced cost as there is no need of maintaining or acquiring these services, increased data security and easy recovery of data as there is no worry about damage or loss of data. The concept of a cloud further evolved into an edge cloud which are flexible and scalable as they are placed at the edge of the network.

Beginning from an era of cloud computing, where the computations were centralized, the world is now witnessing a shift towards Edge Computing, which brings the power of cloud computing closer to the edge of the network. The idea of providing computational services such as storage, networking, analytics, and intelligence, etc. through clouds has proved to offer faster innovation, lower operating costs and flexible usage of resources. But after the advent of the Internet Of Things (IoT), there is an explosive growth of internet-connected devices that use huge amounts of data which further need computational capabilities with extremely low latency requirements. Cloud computing fails to efficiently tackle these issues because an increased load on the network increases the processing time and response time for the applications running in the cloud. Additionally, moving the data back and forth from the customer end to the clouds needs a significant amount of bandwidth, which further inflates latency. To overcome these drawbacks, Edge computing pushes the data to the edge of the network and sends the processed bulk of data to the cloud for storage and management. This diminishes the requirement of high bandwidth, as the data is not traversed through the network to a distant cloud for processing, thereby considerably reducing the latency.

Mobile edge computing nodes (MEC nodes) are attached to densely situated base stations or access points. These nodes exhibit considerable computational capabilities such as CPU power, RAM, battery life, etc. When a user moves around, the Service Placement Problem (SPP) involves finding the optimal edge node that can afford to satisfy the user application requirements. Every MEC node has limited computational capabilities and hence it is also significant to ensure that an optimal MEC node can accommodate the migrated service. Hence, the distribution of services across these nodes is not an easy proposition. A failure to find an optimal MEC node can lead to degradation in latency or inefficient use of resources.

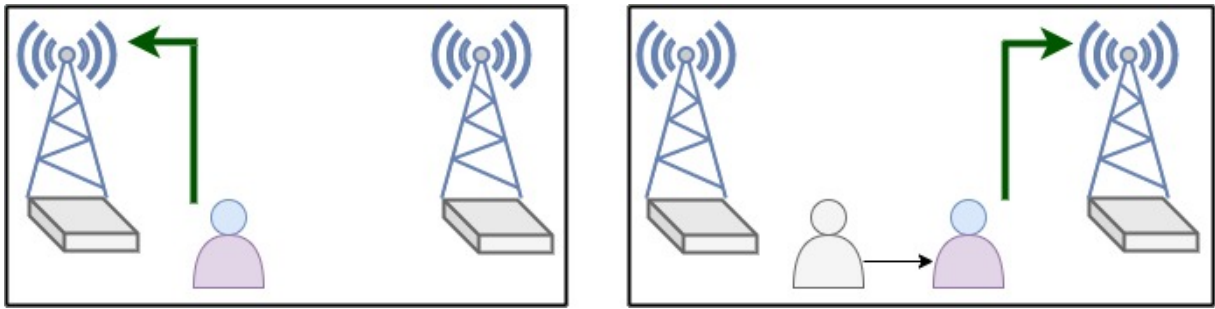


Abbildung 1.1: SPP in a mobile edge computing environment



2 Motivation

3 Background

The thesis focuses on the implementation of a neural network architecture called Pointer Networks, that addresses SPP in a mobile edge computing environment. Pointer Networks are derived from the popular Sequence-to-Sequence learning model which is further combined with the Bahandau attention mechanism proposed by Bahandau in [1]. It was first introduced by Vinayls [9] for solving problems such as Convex Hull, Delaunay Triangulation and Travelling Sales Person Problem(TSP). These problems are termed as Combinatorial Optimization problems, where the dictionary size of the output expected from the neural network depends on the length of the input provided to the neural network. Pointer Networks predict outputs that are pointers to positions in the input sequence that are fed to the neural network encoder. This is achieved by adjusting the Bahandau attention mechanism [1] where traditionally, the attention is combined with the encoder hidden states to form a context vector at every step of decoding. On the other hand, in Pointer Networks the attention is used as a pointer that selects the corresponding indices of the inputs as outputs. Since the output chosen is a member of the corresponding input, the requirement of a fixed-size output dictionary is ruled out. This makes it practical to implement the Combinatorial Optimization problems without having the need to train the model multiple times with varying values for input sequence length.

The paper focuses on obtaining a solution to SPP by selecting an optimal MEC node for a service from a given set of MEC nodes, using Pointer Network. The MEC nodes are passed as inputs to the encoder model, where the availability of these nodes is constrained by its respective capabilities. In order to be able to migrate a service, only the relevant MEC nodes, that has the capability to serve, should be taken into account. Thus, the requisite of considering fixed number of MEC nodes fails to provide an ideal solution. Considering the problem statement similar to Combinatorial Optimization problems, Pointer Networks pose a suitable architecture for SPP. Hence, the primary objective is to evaluate the performance of Pointer Networks in mitigating SPP.

The success of Deep Reinforcement Learning in tackling complex decision-making tasks has gained lot of popularity in the past few years. Using Pointer Networks as the neural network architecture that is combined with a Reinforcement Learning agent to address SPP, is the goal of the thesis. The basic elements in a Reinforcement learning system includes an agent that is responsible for interacting with an environment, an environment that the agent is subjected to, an action that is performed by the agent on the environment and a reward that decides the effectiveness of the corresponding action. Reinforcement learning algorithms are broadly classified as Model-free and Model-based algorithms depending on whether the agent has prior knowledge of the environment. Since for the concerned problem statement, obtaining prior knowledge of the environment is not possible, a Model-free reinforcement learning algorithm called Deep Deterministic Policy Gradient (DDPG), based on Actor-Critic method is used. The Actor is responsible for exploring the environment and the Critic provides feedback on how good the action was.

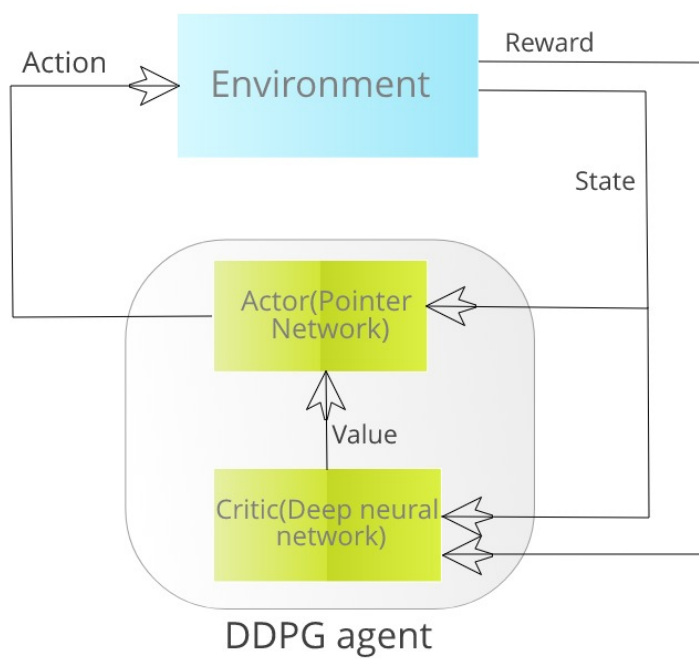


Abbildung 3.1: DDPG based Actor-Critic Reinforcement Learning System

4 Background

The problem of service migration in a mobile edge computing environment is addressed in this thesis. Considering the user location concerning the edge nodes and the availability of resources and its usage in an efficient manner, the application service is migrated to the optimal node.

Machine learning paves the way for systems to derive functions or strategies from a given set of data and formulate solutions, without the need of being explicitly programmed. Undeniably, Machine Learning is a powerful technology for problems where it is difficult to yield solutions manually. The thesis, therefore, focuses on using a Machine Learning strategy called Reinforcement Learning to predict the decision of migrating a service to its optimal MEC node. Additionally, a Reinforcement learning strategy that also incorporates deep learning has proved to achieve high-level performance across many challenging domains. Some proven examples include AlphaGo[6], playing Atari[8].

Considering the problem statement here, the decision such as to which node should the application service be migrated, depends on the number of MEC nodes present in the network. Therefore, the output expected from the neural network, which is the optimal MEC node for migrating an application service, depends on the availability of the MEC nodes that are provided as inputs. Such problems are also called combinatorial optimization problems, where the output of a network is not fixed and depends on the input provided to the network, cannot be solved trivially by a traditional sequence-to-sequence model. Pointer network proves a solution to effectively deal with such problems. A pointer network uses Content-Based Attention Mechanism to predict output that corresponds to the positions in the input sequence. The length of the input decides the number of target classes in the output.

5 Preliminaries

Pointer Networks is a neural network architecture that is devised by combining the Sequence-to-Sequence model with a modified attention mechanism formulated by Bahandau [1].

5.1 Sequence-to-Sequence Model

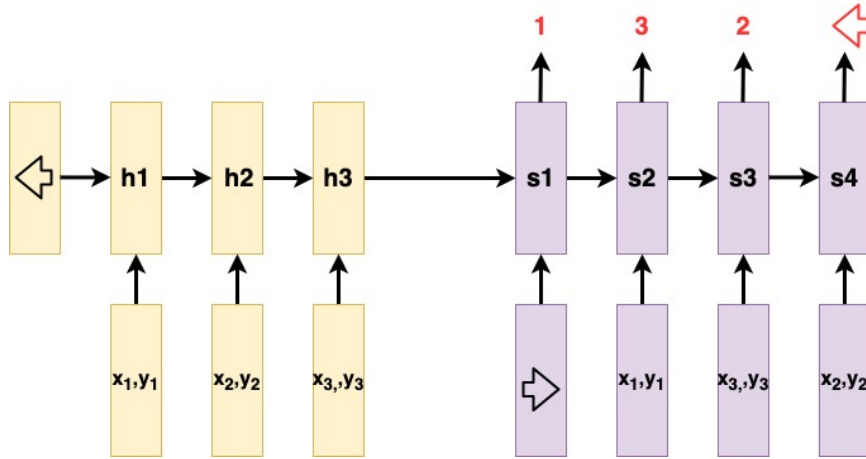


Abbildung 5.1: Sequence-to-Sequence model

Sequence-to-Sequence model is comprised of two Recurrent Neural Networks (RNN), one as an encoder that encodes the input passed to it and the other as a decoder that decodes the encoded input. Fig 5.1 shows an example of the working of a Sequence-to-Sequence model where (x_1, y_1) (x_2, y_2) and (x_3, y_3) are the coordinates of the input data that is fed into the encoder. The output produced by the decoder are 1, 3, 2 respectively, where 1, 3 and 2 are the outputs predicted by the decoder. h_1, h_2, h_3 are the encoder hidden states and s_1, s_2, s_3 are the decoder hidden states respectively. A special symbol indicated as \Rightarrow is used as a start token to commence the process of decoding. For the following stages of decoding, the output from the previous stage is used in combination with the decoder hidden state from the last stage to produce the next output.

$$p(C^P|P : \theta) = \pi_{i=1}^{m(P)} p_{\theta}(C_i|C_1, \dots, C_{i-1}, P : \theta) \quad (1)$$

In the equation $[[9], 1]$, $P_i = P_1, \dots, P_n$ is a sequence of vectors and $C_i = (C_1, \dots, C_{m(P)})$ is the sequence of the corresponding $m(P)$ indices between 1 to $n[9]$. In the inference stage, on passing a sequence P , the corresponding output sequence \bar{C}_P with the highest probability is selected. The process of decoding continues till the network runs into the special symbol indicated by \Rightarrow . Considering the Fig 5.1, when the input co-ordinates x and y are passed as inputs to the encoder, the encoder hidden state value from the last

encoder state, also known as a context vector, that holds the information of all the inputs that are passed through the encoders, is fed to the decoder. This model works fine when the output dictionary C_i is of a fixed size which is equal to n . But the same trained model cannot be used for sizes other than n .

5.2 Attention mechanism

In a Sequence-to-Sequence model, the context vector that contains the information regarding all the inputs fed to the encoder is merely a value drawn from the last encoder hidden state. The drawback in this model is that, the context vector might not be good enough to hold all the relevant information that the decoder needs in order to understand which output pertains to which input. The reason for the mentioned drawback is that the context vector is only absorbed from the last encoder hidden state. This makes Sequence-to-Sequence model incapable of processing large number of inputs, since the valuable information of the inputs in the context vector diminishes as more and more inputs are consecutively passed. The need for having an additional mechanism that retains the relevant information stemmed from this drawback. Thus, the Attention mechanism was introduced. Attention, as the name suggests, provides more pertinence to the input that the output should predict. Instead of using the context vector from the last encoder hidden state, a context vector is generated at every step of decoding. This is achieved by assigning every input with a certain weight, also known as attention weights. The two popularly used Attention mechanisms are namely Bahandau attention [1] and Luong attention [7], which are different from each other in terms of the computation of the attention. Pointer Networks are implemented using the Bahandau attention [1] mechanism which is also known as Align and Translate attention. In this attention mechanism, every hidden state of the encoder and the decoder are involved in creating the context vector, unlike the traditional way of Sequence-to Sequence model where only the last encoder hidden state is used to calculate the context vector. An alignment score is produced for every input-output combination that defines which input should be given more attention to predict an output.

The context vector in Bahandau attention is obtained as shown in the equation [[9],2].

$$\begin{aligned} u_j^i &= v^T \tanh(W_1 h_j + W_2 s_i) \quad \text{where } j \in (1, \dots, n) \\ a_j^i &= \text{softmax}(u_j^i) \\ s_i' &= \sum_{j=1}^n a_j^i h_j \end{aligned} \tag{2}$$

In the equation [[9],2], u_j^i is known as the alignment score which aligns the output i to its best input at index j . The alignment score is obtained by applying the encoder hidden states, depicted as h_j where j is the length n of the input sequence, and the decoder hidden state for output i , depicted as s_i , to an activation function \tanh . The length of the alignment vector is equal to n . W_1, W_2 and v are the parameters that the neural network optimizes during training. On applying softmax activation function over the alignment score, u_j^i is normalised to produce the attention weights a_j^i , which in turn describe the influence of each of the input over output i . The weighted linear combination of the encoder hidden states namely, (h_1, \dots, h_j) and the attention weights a_j^i produces the context vector for the output i . The context vector, output of the decoder from the previous stage and the previous hidden state of the decoder is further used to predict the target output. The working of Bahandau attention mechanism is shown in Fig 5.2. The introduction of attention mechanism in the Sequence-to-Sequence model enhanced the amount of knowledge needed by the decoder to predict an output. But the dictionary size of the output is still dependent on the input and therefore, the drawback of the Sequence-to-Sequence model still persists.

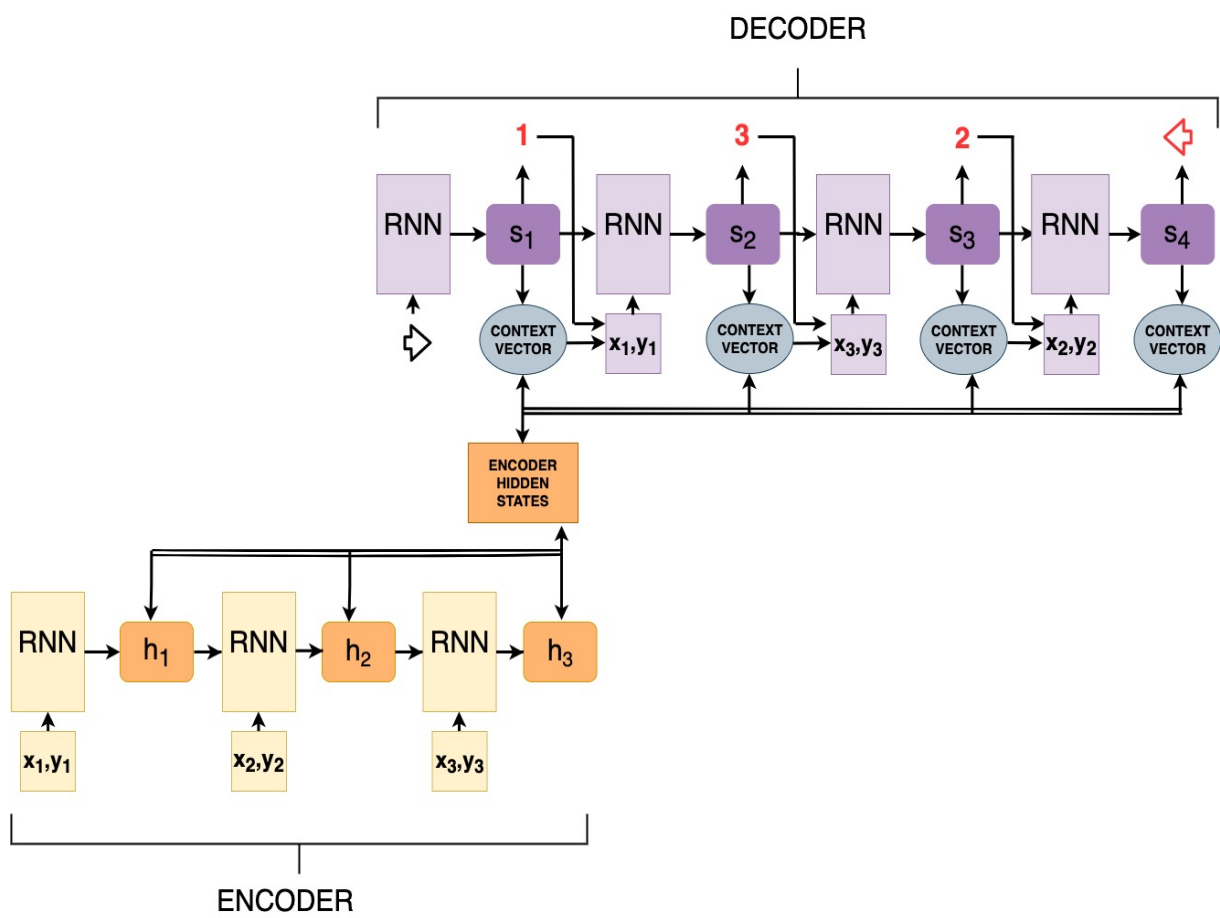


Abbildung 5.2: Bahdanau Attention Mechanism

6 Pointer Networks

Pointer Networks is a simplified augmentation to the Bahandau Attention mechanism [1]. In the equation [[9],3], the softmax activation function is used to calculate the attention weights of length same as j . Instead of further using this value for calculating the context vector, it is directly used as pointers to the input sequence. W_1, W_2 and v are the parameters that the neural network optimizes during training.

$$\begin{aligned} u_j^i &= v^T \tanh(W_1 h_j + W_2 s_i) \quad \text{where } j \in (1, \dots, n) \\ p(C_i | C_1, \dots, C_{i-1}, P) &= \text{softmax}(u_i) \end{aligned} \quad (3)$$

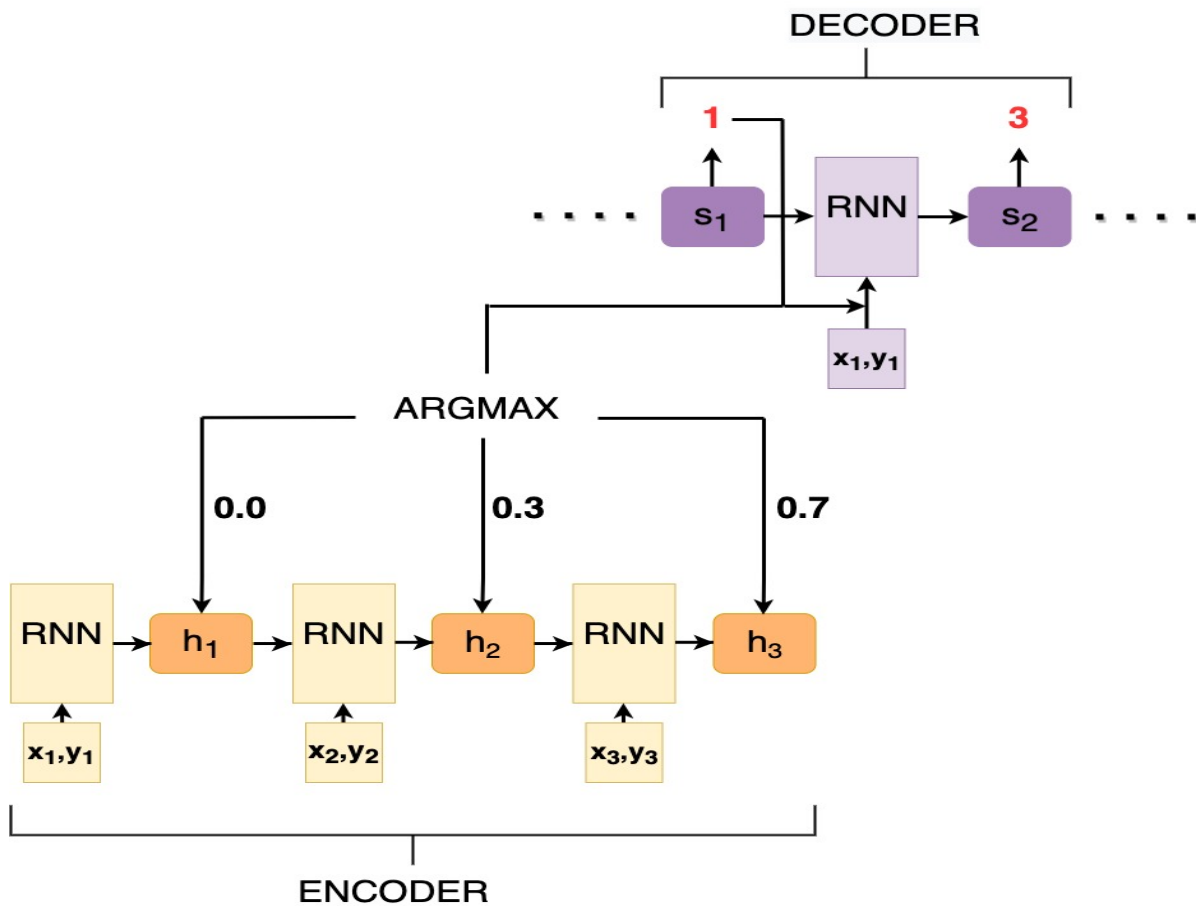


Abbildung 6.1: Pointer Attention Mechanism

7 Models

Mitigating SPP using Pointer Networks is the primary goal of the thesis.

Figure 5.1 is an example of the pointer networks where (x_1, y_1) (x_2, y_2) and (x_3, y_3) are the coordinates of the input. Here, the inputs to the neural network encoder are the x and y coordinates and the output produced are 1, 3, 2 respectively, where 1 indicates the first input, 3 as the third input and 2 as the second input. Thus for a common problem like Travelling Salesperson Problem (TSP), the output could mean the order in which the given cities are visited.

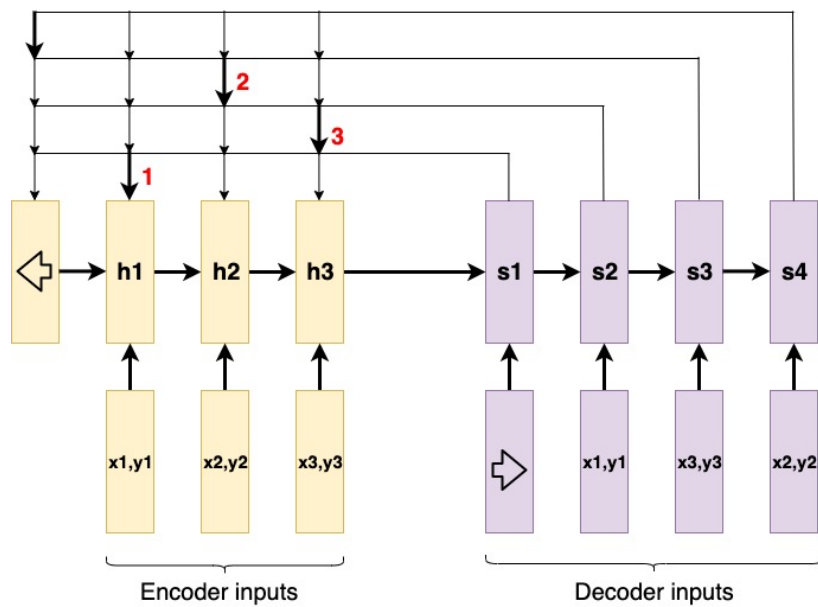


Abbildung 7.1: Pointer network implementation

8 Implementation

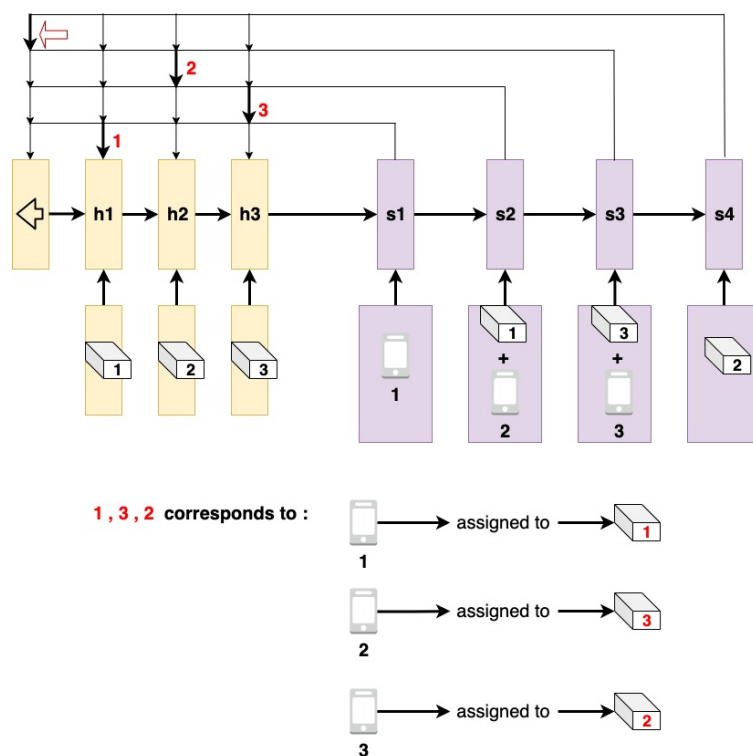


Abbildung 8.1: Pointer network implementation

The implementation in Figure 5.1 uses the MEC node details as input to the pointer network. The application service details are served as inputs to the decoder, concatenated with the previous decoder output values. The output, thus, produced is the MEC nodes to which the application services are migrated to.

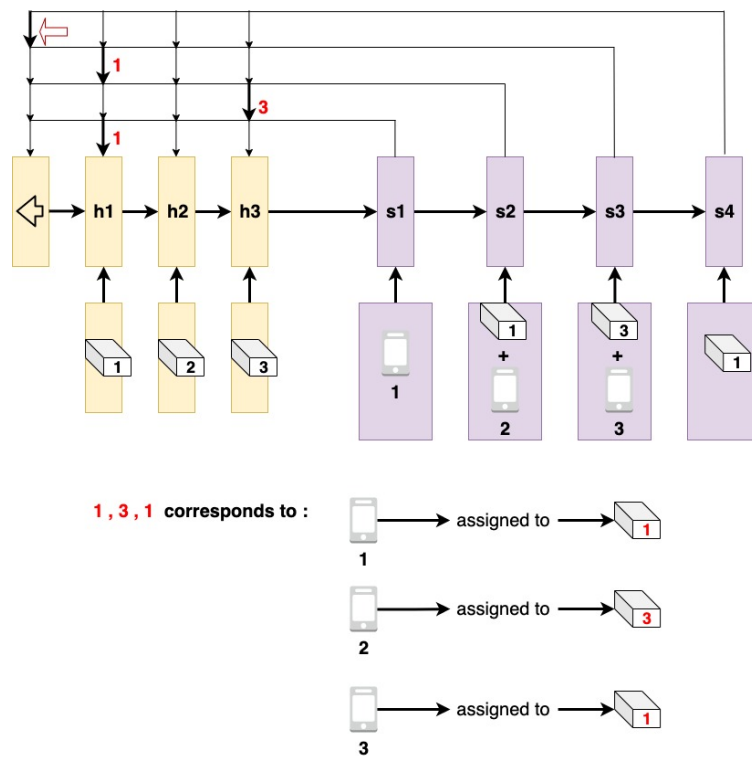


Abbildung 8.2: Pointer network implementation with multiple users on a single MEC node

The same implementation takes into account whether a MEC node shares multiple users that it can accommodate considering its resource constraints. It also ensures that no MEC node sits idle letting an application service to starve.

9 Related Work

SPP in an edge computing environment is a matter of research ever since the advent of the Mobile edge computing era. [4] gives a brief introduction to the Fog computing environment and how SPP can be addressed using various placement possibilities and optimization strategies. A heuristic approach to solve SPP in a scalable in-network environment considering network, as well as research characteristics is introduced in [5]. A framework targeting the same problem with a single agent Reinforcement Learning strategy in a distributed mobile edge cloud environment is discussed in [3]. Pointer network architecture and its working is introduced in [9] which shows how pointer networks can outperform the other existing architecture for combinatorial optimization problems. Belo and Pham [2] proved that the combination of Reinforcement Learning and Neural network using Pointer network gives the best solution to combinatorial optimization problems.

10 Evaluation

The implementation plans to use a simulated network architecture with n users and m MEC nodes. The MEC nodes host k services at a time. Network constraints like latency, and resource constraints like CPU power, RAM, etc. for the user application will be taken into consideration. In order to reduce the latency, the distance of the users from the edge nodes will be taken into account, ensuring that the respective edge node can accommodate the application's resource constraints. The objective of the Reinforcement Learning implementation would be to maximize the rewards, which is the appropriate MEC node the application is migrated to.

11 Execution plan

The thesis implementation is planned to be executed as follows:

WORKING	PERIOD
Literature review to be able to understand the objective thoroughly	2 week
Implementing Pointer network to understand its working and differences that make this network architecture different from other existing architectures	3 weeks
Trying to implement SPP using a supervised learning model in Pointer networks with a simple simulation	3 weeks
Using Reinforcement learning for the same basic implementation.	4 weeks
Modifying the working using a dynamic scenario.	5 weeks
Performing trial and error to ensure if changes in implementations is a good choice	4 weeks
The report work would go hand in hand keeping a note of results obtained from various implementation strategies	5 weeks
TOTAL	26 weeks

Literaturverzeichnis

- [1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [2] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.
- [3] Florian Brandherm, Lin Wang, and Max Mühlhäuser. A learning-based framework for optimizing service migration in mobile edge clouds. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking, EdgeSys@EuroSys 2019, Dresden, Germany, March 25, 2019*, pages 12–17, 2019.
- [4] Adrien Lebre Farah Ait Salaht, Frédéric Desprez. An overview of service placement problem in fog and edge computing. *ACM Comput. Surv.*, page 43, 2019.
- [5] Julien Gedeon, Michael Stein, Lin Wang, and Max Mühlhäuser. On scalable in-network operator placement for edge computing. In *27th International Conference on Computer Communication and Networks, ICCCN 2018, Hangzhou, China, July 30 - August 2, 2018*, 2018.
- [6] Sean D. Holcomb, William K. Porter, Shaun V. Ault, Guifen Mao, and Jin Wang. Overview on deepmind and its alphago zero AI. In *Proceedings of the 2018 International Conference on Big Data and Education, ICBDE 2018, Honolulu, HI, USA, March 09-11, 2018*, 2018.
- [7] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.
- [8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
- [9] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700, 2015.