8

The Controller Placement Problem in Software Defined Mobile Networks (SDMN)

Hakan Selvi,[1] Selcan Güner,[1] Gürkan Gür,[2] and Fatih Alagöz[1]
[1] *SATLAB, Department of Computer Engineering, Bogazici University, Istanbul, Turkey*
[2] *Provus—A MasterCard Company, Istanbul, Turkey*

8.1 Introduction

Traditional networks consist of a large variety of network nodes such as switches, routers, hubs, different network appliances, complicated protocols and interfaces, which are defined in detail through standardization. However, these systems provide limited ways to develop and adopt new network features and capabilities once deployed. Therefore, this semistatic architecture poses a challenge against adaptation to meet the requirements of today's network operators and end users. To facilitate required network evolution, the idea of programmable networks and software defined networking (SDN) has been proposed [1]. This approach is devised to simplify network management and enable innovation through network programmability. In the SDN architecture, the control and data planes are decoupled and the network intelligence is logically centralized in software-based controllers. An SDN controller provides a programmatic interface to the network, where applications can be written to perform management tasks and offer new functionalities. The control is centralized and applications are written as if the network is a unified system. While this simplifies policy enforcement and management tasks, the binding must be closely maintained between the control and the network forwarding elements [1]. For instance, an OpenFlow controller sets up OpenFlow devices in the network, maintains topology information, and monitors the network status. The controller performs all the control and management functions. The information of host locations and external paths are also managed by the controller.

Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture, First Edition.
Edited by Madhusanka Liyanage, Andrei Gurtov, and Mika Ylianttila.

Itsends configuration messages to all switches to set the entire path. The port for the flow to be forwarded to or other actions like dropping packets is defined by the OpenFlow controller [2].

SDN paradigm provides not only facilitation of network evolution via centralized control and simplified algorithms and programmability by enabling deployment of third-party applications but also elimination of middleboxes and rapid depreciation of network devices [3]. Since the underlying network infrastructure is isolated from the applications being executed via an open interface on the network devices, they are transformed into uncomplicated packet forwarding devices [1]. Therefore, the controller-related aspects of the software defined network are paramount for addressing the emerging intricacies of SDN-based systems.

Likewise the wired networks, current mobile networks[1] suffer from complex control plane protocols, difficulties in deployment of new technologies, vendor-specific configuration interfaces, and inflexible and expensive equipment [4]. Although the demand of smart wireless devices has experienced a quantum leap and the mobile data explosion is challenging the mobile networks, the mobile network infrastructure is not adapting to these conditions in a sufficient and flexible manner [5]. In that regard, the concept of SDMN is expected to be instrumental and alter the network architecture of the current LTE (3GPP) networks and, accordingly, of emerging mobile systems drastically [6]. Although SDMN paradigm will facilitate new degrees of freedom for traffic, resource, and mobility management, it will also bring forth profound issues such as security, system complexity, and scalability. The controller placement-related challenges also emerge as critical factors on the feasibility of SDMN.

In this chapter, we discuss important aspects of the Controller Placement Problem (CPP) in SDMN. First, we briefly introduce the SDN controller concept and describe the problem. Second, we discuss the characteristics of SMDN contrasted with wired networks and optimization parameters/metrics of CPP in SDMN. Then we present available solution methodologies and analyze relevant algorithms in terms of performance metrics. Finally, we conclude with some research directions and open problems for CPP in SDMN context.

8.2 SDN and Mobile Networks

The current 3GPP LTE standard defines cellular 4G networks and is updated regularly as releases with a perspective of 5G networks. The mobile network is separated into two strata with current LTE architecture: a packet-only data plane and a management plane to manage mobility, policies, and charging rules. Data plane consists of base stations (eNodeB), Serving Gateways (S-GW), and Packet Data Network Gateway (P-GW). Mobility Management Entity (MME), Policy Charging and Rules Function (PCRF), and Home Subscriber Server (HSS) constitute the management plane [7]. In LTE technology, the network mechanisms execute in a manner as shown in Figure 8.1. The S-GW serves as a local mobility anchor that enables seamless communication when the user moves from one base station to another. It tunnels traffic to the P-GW. It enforces quality of service (QoS) policies and monitors traffic to perform billing. The P-GW also connects to the Internet and other cellular data networks and acts as a firewall that blocks unwanted traffic. The policies at the P-GW can be very fine grained based on various parameters such as roaming status of the user, properties of the user equipment, usage caps in the service contract, and parental controls [4].

[1]In this work, we refer to infrastructure-based mobile networks when we use "mobile networks" term.

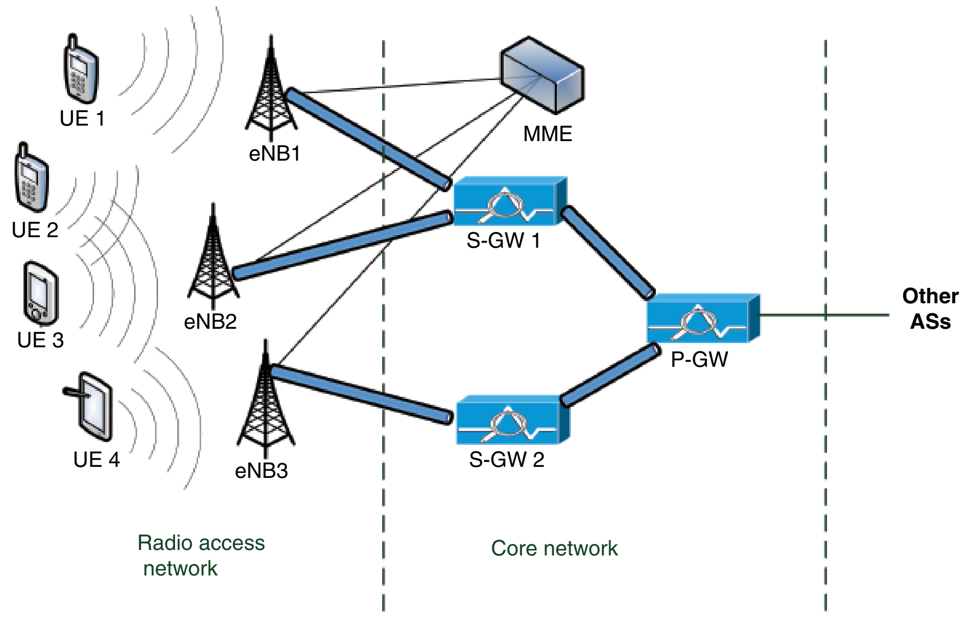

Figure 8.1 Existing LTE architecture (Adapted from Ref. 4).

Although this mobile communication architecture yields to easier management, it still has several limitations. Centralizing data plane functions such as monitoring and QoS functionality at P-GW node introduces scalability challenges due to pecuniary reasons. In that regard, applying SDN principle leads to a flattening of the data plane by simplifying its elements into pure forwarding elements and exporting the control plane intelligence to a remote controller node. This change makes possible to create cheaper equipments and reduces the scalability pressure on P-GW [4]. Additionally, SDN principles are expected to provide flexibility, openness, and programmability to the mobile networks. By means of this approach, mobile network operators can innovate inside their domain more easily with less dependence on UE vendors and service providers [6].

From the perspective of SDN and mobile networks, there are two possible paths for SDN integration to current mobile networks:

- Evolutionary: This is a more probable scenario since there is a huge installed base of mobile networks, and they are envisaged to evolve rather than being completely replaced to meet the requirements of 5G networks. Network virtualization and content-centric operation are expected to be more intrinsic for future networks. This trend is also rendering SDN principles more favorable. The SDN integration is going to be intertwined with these changes.
- Clean-slate approach: Clean-slate design and greenfield deployments provide more degrees of freedom since they are not subject to constraints posed by incumbent systems. However, they are more costly and difficult to implement. Although the design and specification of mobile networks according to SDN paradigm poses substantial issues such as security, scalability, and performance, it is more challenging in practical terms than those theoretical aspects.

The main enabler construct for realizing SMDN is a distributed layer of sensors and actuators embedded in mobile network tiers for enabling centralized control and intelligence migration to the controllers. Li et al. [4] describe a cellular SDN architecture and posit four main extensions to enable SDN principles in cellular networks, namely, *policy support*, *agent-based operation*, *flexible data plane functionality*, and *control of virtualized wireless resources*.

An SDN forwarding device contains one or more flow tables consisting of flow entries, each of which determines how a packet performs [1], and the controller updates these flow tables and instructs the switches as to what actions they should take via a programmatic interface called southbound interface [8]. Since the control is centralized and applications are written as if the network is a single system, policy enforcement and management tasks are simplified [9]. The outcome of the completed experiments in Ref. [10] shows that a single controller has capability to manage excessive number of new flow requests in an unexpected manner. However, in a large-scale mobile network deployment, the centralized approach has some limitations related to the interaction of control and forwarding elements, response time, scalability, infrastructure support, and availability. Typically, large amount of network flows originating from all infrastructure nodes cannot be handled by a single controller because of the limited resource capacity. Another work by Voellmy and Wang promotes this claim and shows that multiple controllers ensure high fault-tolerant networks with reduced latency [11]. Therefore, one must clarify four fundamental issues for SDMN context [12]:

1. How many controllers are needed?
2. Where in topology should they go?
3. How should they interact?
4. How will the mobile network evolution toward SDN be reflected in this problem? This effect can be on a multitude of aspects such as standardization, problem formulation, or solution methodologies.

The answers of these essential questions depend on the network topology among other user-imposed requirements. From the latency perspective, a single controller would be mostly adequate. On the other hand, fault tolerance and scalability concerns impel researchers to consider using multiple controllers in networks [12].

Moreover, the architecture of mobile networks and its provisioning is a challenging and complex task due to kaleidoscopic information about the network's topology [13]. Dynamically changing topology is an inherent characteristic of mobile networks. Thus, the deployment of multiple controllers requires being in harmony with related "on-the-fly" network units. Nevertheless, the controllers should perform synchronously to maintain a consistent view of the network [14]. If redundant controller(s) is integrated into the system, additional communication overhead will occur. Thus, the location of controllers should be optimal. Lastly, mobile network traffic can fluctuate over time; controller placement scheme should regard dynamic rearrangement of the number and the location of controllers [9].

8.3 Performance Objectives for SDMN Controller Placement

In this section, we will examine various controller requirements that affect the network state and algorithm efficiency. It is harder for fully distributed control planes to fail compared to centralized planes. However, for controller placement, there is a trade-off among performance objectives of the placement algorithm. Some algorithms could place controllers to maximize

Table 8.1 Performance objectives and their effects on networks

	Scalability	Reliability	Latency	Resilience
Fault tolerance		✓		✓
Service delay	✓		✓	
Utilization	✓	✓	✓	

fault tolerance, or some could minimize the propagation delay or distance to the *n*th closest controller [12]. The general performance objectives for control placement and their related effects on the network can be seen in Table 8.1. To minimize the delay of network-based services, scalability and latency are considered. For scalability, service delay may be traded, while latency minimization directly benefits service delay. For utilization of the network, the effects of the scalability, reliability, and latency are taken into account. Fault tolerance is directly affected by reliability and resilience objectives.

8.3.1 Scalability

For networks with more than one controller, a controller may become overloaded if the switches mapped to this controller have large number of flows. However, the remaining controllers may operate underutilized. It is instrumental to shift load across controllers over time depending on the temporal and spatial variation in traffic conditions. Static controller assignment can result in suboptimal performance since no switch will be mapped into a less loaded controller. The replacement of the controller can help to improve performance of overprovisioned controllers. Instead of using static mapping, elastic controller architecture can be used to map controller to balance the load as it reflects performance.

8.3.2 Reliability

According to IEEE, *reliability* is defined as “the probability that a system will perform its intended functions without failure, within design parameters, under specific operating conditions, and for a specific period of time” [15]. If the connection between controller and the forwarding planes is broken because of the network failures, some switches will be left without any controller and thus will be disabled in SDN-based networks. Network availability should be ensured to assure the reliability of SDN. Therefore, improving reliability is important to prevent disconnection between controller and the switch or between controllers. To reflect the reliability of the SDN controller and to find the most reliable controller placement for SDN, a metric can be defined as the expected percentage of the valid control paths when network failures happen [12]. A control path is defined as the route set between switches and their controllers and between controllers. Consistency of the network should also be ensured when multiple controllers are in the network.

Each control path uses existing connection between switches. If control path is represented as a logical link, SDN control network is responsible to enable a healthy communication between switches and their controllers, which is a requirement for control paths to be valid. The failure of control paths, which means the connection is broken between switch and its controller or among controllers, results in the case where control network will lose its functionality. If the number of the control paths is too large, forwarding service may fail, which causes severe problems. So to define a controller placement algorithm, reliability must be considered as a placement metric. To formulate reliability better, various statistical and empirical approached are possible [12].

The optimization target to define a reliable network is to minimize the expected percentage of control path loss. To maximize reliability of SDN, several placement algorithms are developed to automate controller placement decision that work in a reliability-aware manner [10]. These algorithms are described in Section 4.1.

8.3.3 Latency

Latency is simply "the delay between the time the data is sent from its origin and received at its destination" [16] and a critical QoS metric for communication networks. It is more important for multimedia communications since that kind of traffic is delay sensitive. For instance, one of the envisaged requirements of 5G networks is to have a delay smaller than 1 ms, which implies an order of reduction compared to 4G networks. For large-scale networks, single controller deployment is typically not sufficient to reach adequate performance due to various factors. However, when the network has several controllers, a new matter of contention emerges. Since several controllers maintain the control logic of the network, these controllers need to communicate with each other to maintain network consistency. The latency between controllers has to be considered, especially if controller communication traffic is frequent [17]. The latency in that setting comprises of processing, transmission, and propagation latencies.

Even though latency between controllers is considered during placement, the reaction of the remote controller and the time to pass for delivering the reaction to a switch bound the overall network performance. It can be called as *propagation latency*, which should be at reasonable levels for speed and stability. It is enough for propagation delay to become infeasible for real-time tasks or slow down unacceptably even for small delay. Adding some intelligence to switches can reduce these delays. However, this method adds complexity to the system, and it is against the idea that SDN uses a simple and dump switch model.

Controller placement algorithms are developed to minimize latencies or maximize some latency-based parameters, which are defined as:

- *Average-case latency*: If the network is simplified as a network graph, the connections between components represent edges. The weight of the edge represents propagation latency. The average propagation latency L_{avg} for a placement of controllers is average of minimum propagation latencies on each edge. $d(v, s)$ is the shortest path from node $v \in V$ to node $s \in V$:

$$L_{\text{avg}}(S') = \frac{1}{n} \sum_{v \in V} \min_{s \in S'} d(v, s)$$

- *Worst-case latency*: This value is defined as the maximum propagation delay from node to controller:

$$L_{\mathrm{wc}}(S') = \max_{(v \in V)} \min_{(s \in S')} d(v, s)$$

- *Nodes within a latency bound:* Instead of minimizing the average or worst case, it might be better to place controllers in a way that it maximizes the number of nodes within a latency bound. This approach is called as *maximum cover*. For most topologies, adding controllers yields slightly less than proportional reduction [12].

8.3.4 Resilience

A good controller placement should minimize latencies between nodes and controllers or among controllers. However, minimizing latency is not always sufficient. According to Ref. [17], the placement of the controller should also meet some resilience constraints. These constraints are defined in this section.

8.3.4.1 Controller Failures

Using more than one controller not only decreases latencies but also increases tolerance of the network to failures in case the controllers stop working. In a related work [18], it is assumed that a node is not able to route anymore and becomes practically off if it loses its connection to the controller. However, Hock et al. [17] suppose that in case a controller is out of order, all the switches assigned to the failed controller can be reassigned to the second closest controller by using a backup assignment or signaling-based shortest path routing. Until the last controller survives, all nodes are functional in this way. Although resilience is considered, this solution will probably increase the latency of the reassigned nodes and their new controller. The new controller may be much further away compared to the previous one. This situation would result in higher latency. The described failure scenario is an example of the worst case, since the last surviving controller is located furthest from the center of the network such that some of the nodes need to pass through the whole network to reach the controller. However, a placement algorithm to increase resilience should also consider this worst-case scenario during failure-free routing.

8.3.4.2 Network Disruption

In a network, not only controller failures occur. Network components, links, and nodes may also suffer damage, which is more important to consider because the topology itself is changed. Because of link failures, paths between some nodes are severed. This situation causes nodes to be assigned to some other controllers even though latencies may increase. Additionally, some parts of the network may be in danger because of these link failures, and many nodes cannot be assigned to any controller. Although these nodes may be working and are able to perform forwarding operations, they cannot get control messages from any controller.

Link failures prevent rerouting of nodes even if they are physically connected, since path will no longer be available.

8.3.4.3 Load Imbalance

If the nodes are assigned to the nearest controller using latency as a metric or shortest path distance between the node and controller, there may be situations when some controllers are overloaded due to excessive traffic flow. There can be an imbalance in the number of nodes per controller in the network. Typically, the higher the number of nodes attached to a controller, the greater the load on that controller. The increase in number of node-to-controller requests in the network induces additional delay due to queuing at the controller system. Resilience for controller placement requires nodes of different controllers to be well-balanced.

8.3.4.4 Intercontroller Latency

Since single controller is not sufficient to ensure resilience in a network, if any two controllers are far away from each other, the messages from one to the other need to pass through the entire network, which increases intercontroller latency.

8.4 CPP

The controller placement strategies affect every aspect of SDN, from node-to-controller latencies to network availability and from operational costs to performance. In addition, the peculiarities of SMDN such as mobility compared to wired networks complicate the problem structure. Optimizing every variable in this problem is NP hard, so it is very important to find an efficient controller placement algorithm [19]. Heller et al. concluded that finding optimal solution is computationally feasible but within failure-free scenarios in [12]. They took into consideration just latency requirements like *average-case latency* and *worst-case latency*, and reasonably, they presented that in most topologies, one single controller is enough to fulfill the existing latency requirements. If we broaden our viewpoint to CPP with various goals, which include reliability, network resilience, fault tolerance, or load balancing, many more controllers communicating with each other are necessary to meet these resilience requirements [17].

In Figure 8.2, we depict various factors or parameters in CPP setting. The fundamental factors determining the solution space are the location and number of controllers and their communication requirements. Moreover, the SDN controllers may be of different characteristics entailing processing capability, supported communication primitives, and intelligence. This condition implies nonidentical controllers leading to heterogeneity. The network characteristics also affect the problem structure drastically. For our focus in this chapter, this phenomenon is paramount since mobile networks exhibit peculiar inherent characteristics due to mobility, wireless transmission, and dynamic network structure. For any algorithm or scheme for CPP, there are practical constraints such as complexity. Although these factors are assumed to be operative for relatively asynchronous execution of controller, CPP algorithm may need to be more active and executed frequently when network function virtualization becomes more common.

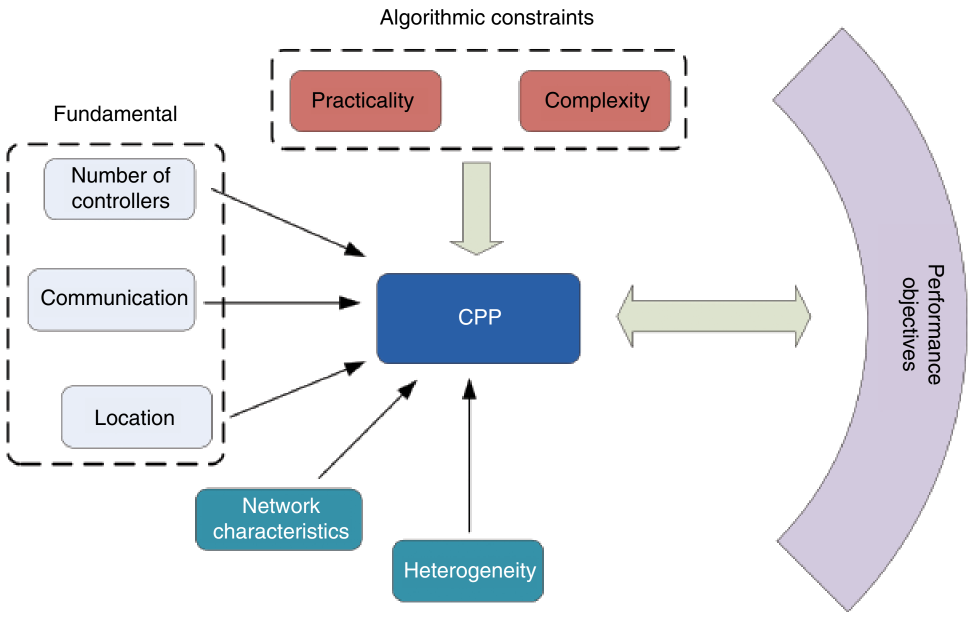

Figure 8.2 CPP parameters and performance objectives.

Since a single controller cannot handle large amount of flows within the network with desired performance capabilities due to capacity limitation, multiple controllers are employed for better network management. However, one must specify the number of controllers to use and the locations of them in the network architecture [19]. While trying to find these important questions' answers, the main objective should be not only to minimize the latencies between nodes and controller but also to maximize resilience by fulfilling certain constraints.

8.4.1 Placement of Controllers

The separation of forwarding and control planes allows forwarding plane to be simple with the controller plane entailing and managing the network intelligence. However, this separation can impair system performance, e.g. reduce reliability of the communication. Thus, in network design, placement of controller(s) should be considered in a way that it achieves both reliability and performance.

8.4.1.1 Single Controller Placement

In single controller placement (SCP), the aim is to place a single controller according to some predefined objective. A very common objective is to assure a high level of resilience 'i.e. to protect the controller from being disconnected from nodes. This is important due to the fact that the first requirement for successful operation of a controller is to keep it capable of communicating with its peers in the network.

In Algorithm 8.1 (optimal placement), all possible locations are scanned to find one node, which maximizes the resiliency of the network. Assume there is a switch A in the network. The switch is protected if and only if a switch B is not downstream of A and there is a link between A and B, which is not a part of the controller routing. For evaluating the protection status of switch *u*, the existence of the link, which satisfies the condition, is picked. The algorithm's final step is to find a location for controller, which minimizes the probability that when there is a failure in the network, a node is disconnected from a controller [20].

Algorithm 8.1 Controller Placement: Optimal Algorithm

```
procedure Optimal Placement (T)
        for each node $v \in V$ do
        $T$= controller routing tree rooted at $v$
        $\Gamma(T)$ = The weight of a routing tree to be the sum
  of the weights of all its unprotected nodes
             for each node $u \neq v$ do
                  $W = 0$
                if u is not protected then
                  $W$ = number of downstream nodes of $u$ in $T$
                end if
                $\Gamma(T) = \Gamma(T) + W$
             end for
             controller location = node $v$ with minimum $\Gamma(T)$
        end for
```

When the network size is large so that searching among all locations is not practical, a heuristic method is required. Algorithm 8.2 is a heuristic method that selects the node with the largest number of directly connected nodes. $D'(v)$ denotes the number of the protected neighbors of the node. The algorithm continues until it finds a node with the maximum number of protected neighbors [20].

Algorithm 8.2 Controller Placement: Greedy Algorithm

```
procedure Greedy Placement ()
     Sort nodes in V such that D(v(1)) ≥ D(v(2)) ≥
  D(v(n))
     controller location = v(1)
     for i = 1 to n do
          A=set of neighbors of node v(i)
          D′(v(i))=number of members of A that are
  connected to other
```

```
               members, either directly or through one hop
other
               than the controller.
           if D′(v(i)) > D′(controller location) then
               controller location = v(i)
           end if
           if (D′(v(i)) == D(v(i)) then
                break;
           end if
      end for
```

In previous two algorithms, no controller routing is considered. Any arbitrary routing tree can be chosen to maximize the protection of the network against component failures or optimizing performance. Since finding a routing that maximizes the protection of the network for any controller location is an NP-hard problem, algorithms can be used to find a suboptimal solution. Algorithm 8.3 is a resilience-improved routing scheme. The algorithm starts with a shortest path tree and modifies the tree to add to the resilience of the network. Iteration is continued until no further improvement is possible that increases the resiliency [20]. From all these three algorithms, Greedy Routing Tree algorithm performs better than the two other according to Ref. [20].

Algorithm 8.3 Greedy Routing Tree (GRT) Algorithm

```
procedure Routing Greedy (G, controller loc)
                          T = shortest-path tree
                          i = 1
                          repeat
                          for nodes v with d(v,
 controller) == i do
                                 if v is the only node
 with d(v,controller)==i then
                                         next;
                                 end if
                                 for every node u ∈ V \
 {downstream nodes of v} and
                               (v, u)∈ E and (v, u)∉ T
 do
                                        if d(u,
 controller) ≤ d(v, controller) then

 T′(u) = tree built by replacing (v,
                                   upstream node of v in
T) by (v,u)
                                        if (T′) < $\Gamma(T)$ then
                                             replace T by T′
                                                  end if
                                          end if
                                    end for
                            end for
                            i = i + 1
                            until all nodes checked
```

8.4.1.2 Multiple Controller Placement

As the size of the network increases, using a single controller reduces the reliability and degrades the performance of the network. Therefore, multiple controllers are employed for better network availability and management. The CPP for this setting is denoted as multiple controller placement (MCP). As in SCP, in case of deploying multiple controllers, consistency of the network state must be achieved in the communication between controllers. The answer to where to place controllers depends on the metric choices and network topology itself. To place the multiple controllers efficiently, the CPP should have a near-optimal solution once the optimal solution is unattainable.

Graph-Theoretic MCP Problem Formulation

The CPP is inherently suitable for graph-theoretic modeling since it addresses a node selection problem in network graph. If we define the network as a graph (V, E), V is the set of nodes and $E \subseteq V \times V$ is the set of links. Let n be the number of nodes $n = |V|$. Network nodes and links are typically assumed to fail independently:

- p is the failure probability for each physical component $l \in V \cup E$.
- $path_{st}$ is the shortest path from s to t, given that s and t are any two nodes.
- $V_c \subseteq V$ is the set of candidate location where controllers can be placed.
- $M_c \subseteq V$ denotes the set of controllers to be placed in the network.
- M and $P(M)$ denote the number and a possible topological placement of these controllers, respectively.

To reduce propagation delay, each switch is connected to its nearest controller using the shortest path algorithm. If several shortest paths exist, the most reliable one is picked. A good placement should maximize the existing connectivity among the switches. All controllers can be connected to all switches forming a mesh. However, this will increase the complexity and the deployment cost. It will decrease the scalability of the network since the network size grows as switches spread across the geographic locations. To maximize network resilience and connectivity, to increase scalability, and to decrease probability of failure, controllers should be placed accordingly, which is an optimization problem [18].

In the following part, we describe and discuss some MCP algorithms studied in literature.

Random Placement

Although this is usually not a practical algorithm, it is typically used as a baseline case for performance evaluation. In random placement algorithm, each candidate location may have a uniform probability of hosting a controller. In that case, RP algorithm randomly chooses k locations among all potential sites, where $k=1$ for single controller. Another option is to utilize a biased probability distribution, which reflects a preference among potential controller locations. This scheme is instrumental to concentrate controller deployments to specific network segments.

Greedy Algorithms

Greedy algorithms adopt the locally optimal solution at each stage of the algorithm's run. Although a greedy algorithm does not necessarily produce an optimal solution, it may yield locally optimal solutions that approximate a global optimal one in a reasonable amount of time.

Algorithm 8.4 l-w-Greedy Controller Placement Algorithm [19]

```
procedure l-w-greedy Controller Placement
    Sort potential location $V_c$ in descending order of node
  failure properties, the
    first $w|Vc|$elements of which is denoted as array $L_c$
    if $k \leq 1$ then
        Choose among all sets $M'$ from $L_c$ with $|M'|=k$ the set
  $M''$ with maximum $\partial$
        return set $M''$
    end if
    Set $M'$ to be the most reliable placement of size $l$
    while $|M'| \leq k$ do
        Among all set $X$ of 1 element in $M'$ and among all
  set $Y$ of $l+1$ elements
        in $L_c - M' + X$, choose sets $X$, $Y$ with maximum $\partial$
        $M' = M' + Y - X$
    end while
return set $M'$
```

The MCP problem naturally lends itself to greedy approaches since controllers can be placed one by one during the solution. Hu et al. [19] describe l-w-greedy algorithm (Algorithm 8.4) where controllers are placed iteratively in a greedy way. k controllers are needed to be replaced among $|V|$ potential locations. A list of potential locations is generated, which are then ranked increasingly according to failure probabilities of switches. One location at a time is selected from $w|V|(0<w\leq 1)$. For first iteration $l=0$, the algorithm computes the cost associated with each candidate location under the assumption that connections from all switches converge at that location. Location with the highest value is picked. In the second iteration, the algorithm

searches for a second controller with the highest cost from candidate locations. The algorithm is iterated until all k controllers have been chosen and placed.

For $l>0$, after l controllers have placed, the algorithm allows for l steps backtracking in each subsequent iteration. All possible combinations are checked of removing l of already placed controllers and replacing them with $l+1$ new controllers [21].

Metaheuristics

A metaheuristic is a higher-level heuristic designed to find, generate, or determine a lower-level heuristic that may provide a suboptimal, albeit sufficiently good, solution to an optimization problem. They are typically utilized when the optimization problem is too complex for the computational resources or incomplete or imperfect information is available. Some examples are tabu search, evolutionary computation, genetic algorithms, and particle swarm optimization.

Hu et al. [21] investigate various CPP algorithms including simulated annealing (SA) metaheuristic. SA is a probabilistic method for global minimum of a cost function that may possess several local minima [22]. Although SA is a known technique for global optimization problems, the key of effective usage is optimizing the configuration of the algorithm. It is important to reduce the search space and converge to the vicinity of optimal placement rapidly.

For the MCP problem, SA can be devised as follows:

1. *Initial state*: Place k controllers at the k most reliable locations.
2. *Initial temperature*: To make any neighbor solution to be acceptable, the initial temperature T_0 should be a large value. P_0 is acceptance probability in the first k iterations.
   Δ_0 is the cost difference between the best and the worst solutions obtained in Y executions of the random placement. T_0 can be computed by $-|\Delta_0|/\ln P_0$.
3. *Neighborhood structure*: $P(M)$ denotes a possible placement of k controllers. x_c controller location of $P(M)$. x_k location of $(V-P(M))$. The best exchange x_k in $(V-P(M))$ is defined such that $\Delta_{ck}=\min_j \in (V-P(M))\Delta_{cj}$ where Δ_{ij} is the reduction in the objective function that is obtained when $x_i \in (V-P(M))$. The cycle of the algorithm is completed when all $x_c s$ in $P(M)$ are examined.
4. *Temperature function*: The temperature decreases exponentially. That is, $T_{\mathrm{new}=}\alpha T_{\mathrm{old}}$ [21].

Brute Force

With a brute-force approach, all possible combinations of k controllers in every potential location are calculated. Then, the combination with the best cost is picked. This approach is exhaustive and optimal result is obtained after an extremely long execution time even for small networks. Feasible solution can be found by the brute-force algorithm, but it is infeasible to run a brute force to completion, which can take weeks to complete for large topologies [9].

Experimental Results

Figure 8.3 shows the cumulative distribution (CDF) of the relative performance of the algorithms on the Internet2 OS3E (Open Science, Scholarship and Services Exchange) topology according to Ref. [21]. The result of the algorithm comparison is 2-1-greedy and 1-1-greedy, and SA performs the best. SA performs better than 2-1-greedy, which finds better placement than 1-1-greedy. The random placement has the worst performance as expected.

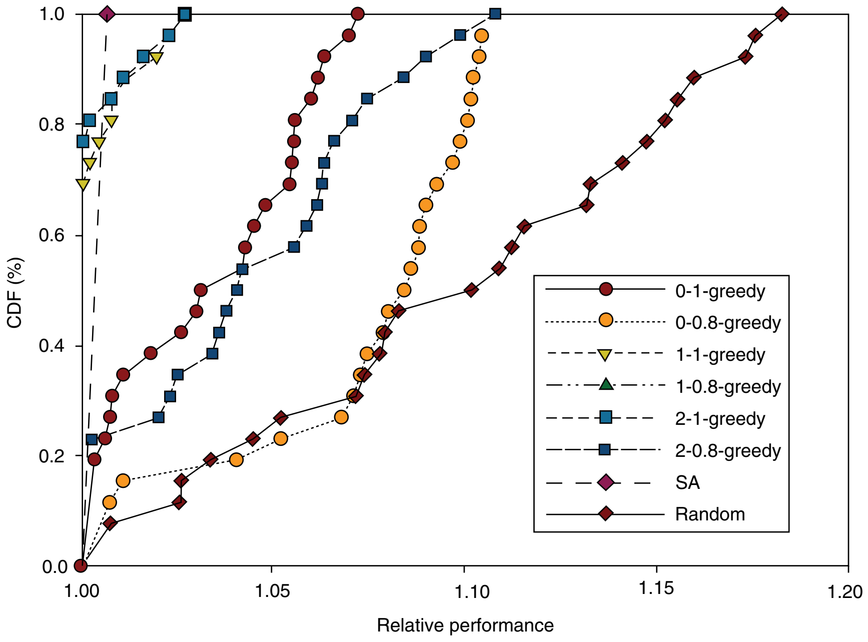

Figure 8.3 The CDF of relative performance of the placement algorithms on OS3E topology [21].

8.4.2 Number of Required Controllers

If the employed controllers are located efficiently but without a predetermined number, one should also find the answer of the following challenging question: how many controllers should we use in order to meet our objectives? Apparently, the answer is a variable according to trade-off considerations between related objectives/metrics. Heller et al. [12] studied to get a result from just latency point of view. Figure 8.4 shows that although the effect of controller numbers varies from average-case latency to worst-case latency, increasing controller numbers implies a proportionally reduction in both. Hu et al. approach the problem from a different viewpoint and focuses on reliability as the main concern [21]. The results of their experiments showed that the optimizations on different topologies provide very similar results. Using too few (even a single) controllers reduced reliability expectedly. However, the results also show that after a certain controller proportion in the network, additional controllers and expected path loss is inversely correlated because if large numbers of controllers are employed redundantly, too many controller paths between controllers cause low reliability.

According to Hock et al. [17], if one considers fulfilling more resilience constraints, which are addressed in Section 8.3.4, he must locate the controllers in a controller-failure and network-disruption tolerated way. Thus, there must be no controller-less node in the network, and "a node is considered controller-less if it is still working and part of a working subtopology (consisting of at least one more node), but cannot reach any controller. Nodes that are still working, but cut off without any working neighbors, are not considered to be controller-less." as defined in Ref. [17]. Therefore, number of controllers should be increased

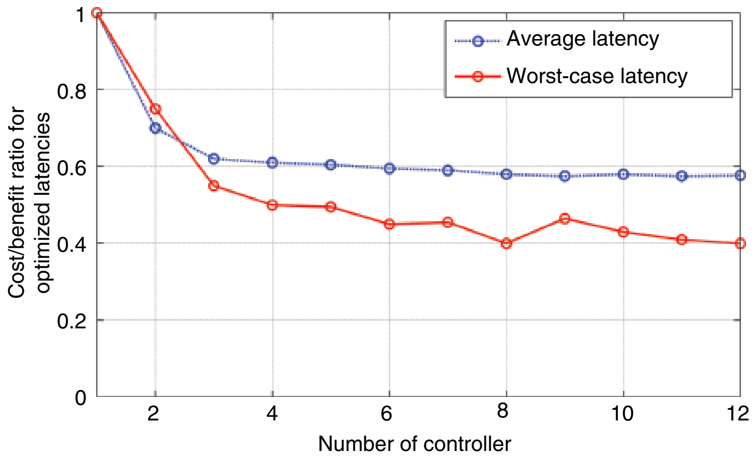

Figure 8.4 Cost/benefit ratio: a value of 1.0 indicates proportional reduction, where k controllers reduce latency to 1/*k* of the original single controller latency [12].

until there is no controller-less node. It can be deduced from the definition that if a node has at most two neighbors, one of them should be controller node to be not controller-less when both neighbors fail. (They limit defects to the case of two simultaneous failures in that work. Because if more than two arbitrary failures happen concurrently, the topology can be completely disrupted and no controller placement would help it anymore.) The experiment results show that on Internet2 OS3E topology, the number of controller-less nodes decreases with increasing number of controllers (*k*) and it is possible to eliminate all controller-less nodes in all one and two failure scenarios with a number of seven controllers.

So to calculate the number of required controllers, the network must be divided into virtual subtopologies consisting of at least two nodes that can be totally cut off from the remaining part of the entire network by at most two link/node failures, and one of the internal nodes must be controller node [17]. Then, we can find the number of necessary controllers in two phases:

1. Find all possible subtopologies of at least two nodes, which do not include any smaller subtopology in itself. Since all found subtopologies need a separate controller, the maximum count of necessary controllers is 8 in Figure 8.5.
2. Since it is aimed to use a minimum number of controllers covering all subtopologies, try to minimize the number, which is found in (1). There are three intersected subtopologies in the network as shown at the upper right-hand corner of Figure 8.5. Two controllers are sufficient to manage these three subnetworks. Thus, the minimum required controller number is 7.

There are 34 nodes in Figure 8.5, so there are (34/7) = 5.4 million possible placements with seven controllers, but there are two possible controller nodes for each subtopology, and three possibilities for intersected subtopologies reduce possible controller placements to $2^5 \times 3 = 96$. However, the best placement of these 96 possibilities in terms of maximum overall node-to-controller latencies is colored by red and the magnitude is 44.9% of diameter

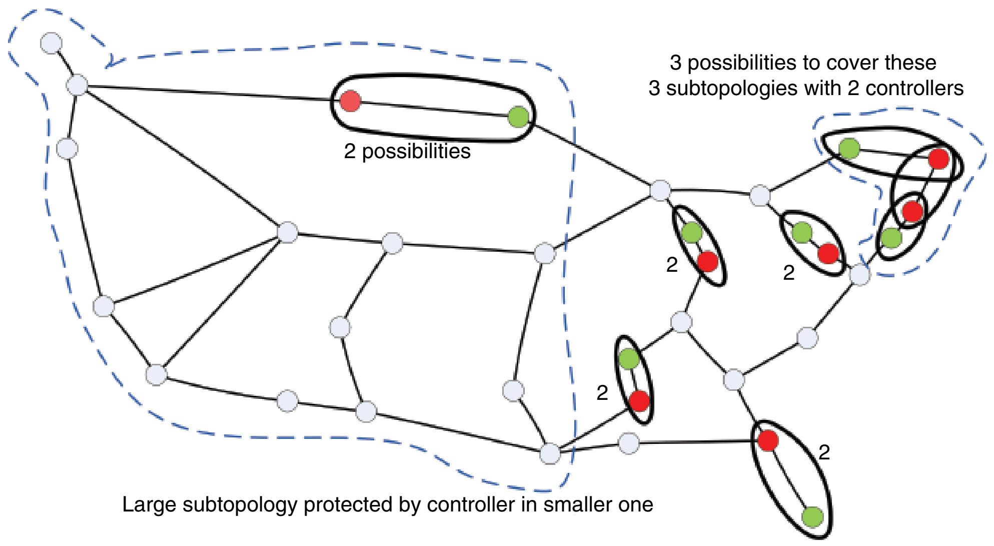
3 possibilities to cover these
3 subtopologies with 2 controllers
2 possibilities
2
2
2
2
Large subtopology protected by controller in smaller one

Figure 8.5 Subtopologies needing a controller to eliminate controller-less nodes [17]. There are $2^5 \times 3 = 96$ potential placements in total.

of the network, which is shown 22.5% in [12] regardless of any resilience constraint. That indicates that there is a trade-off between resilience constraints and latencies [21]. When optimizing for average latency in OS3E topology, the best placement for a single controller also provides the optimal reliability. However, if the network uses more controllers, optimizing for latencies reduces reliability by ~13.7%; in a similar vein, optimizing reliability increases the latencies, but it is possible to find an equilibrium point where it provides latency and reliability constraints [12].

8.4.3 CPP and Mobile Networks

The ramifications of mobile networks on CPP are depicted in Figure 8.6. For mobile networks, resilience is a critical issue considering the multitiered structure especially for heterogeneous wireless networks. Moreover, load variance is much larger, which affects the processing and response times of the controller. The load variance and characteristics of mobile networks are required to be integrated into these problem definitions. The practicality of the algorithm needs to be considered considering the information and protocol data exchange of mobile network specs.

The complexity factor has two aspects: *offline* corresponding to the complexity of the algorithm execution versus *online* corresponding to the complexity of the controller signaling and operation in the mobile network. Moreover, the complexity is higher due to the diversity of network nodes and mobile end devices. This is also reflected in the heterogeneity-related challenges caused by multiple network tiers and network function virtualization in the system.

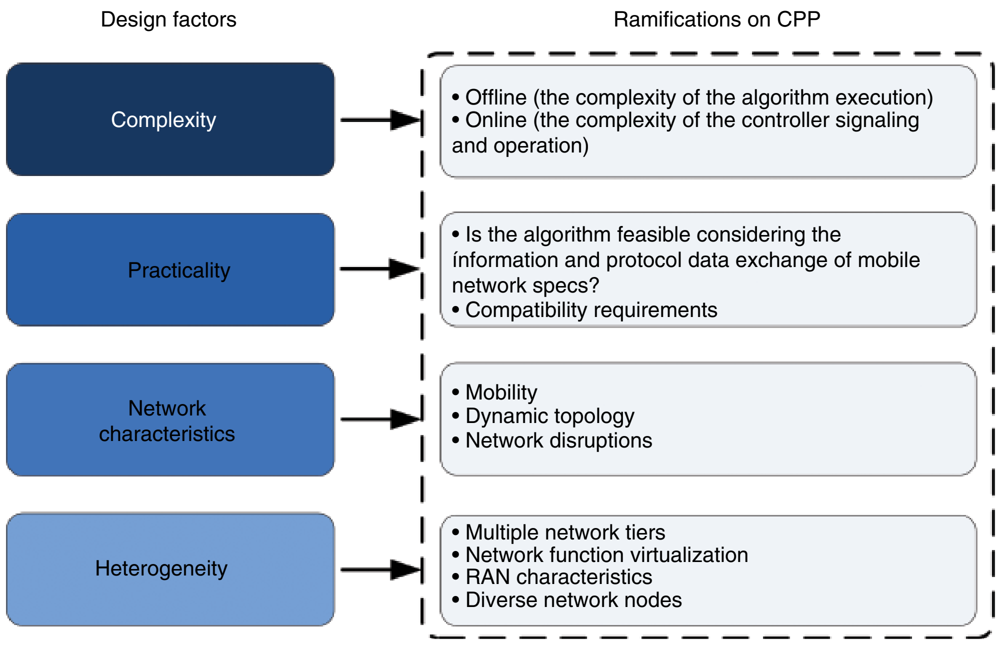
Design factors
Ramifications on CPP
Complexity
Practicality
Network characteristics
Heterogeneity
• Offline (the complexity of the algorithm execution)
• Online (the complexity of the controller signaling and operation)
• Is the algorithm feasible considering the information and protocol data exchange of mobile network specs?
• Compatibility requirements
• Mobility
• Dynamic topology
• Network disruptions
• Multiple network tiers
• Network function virtualization
• RAN characteristics
• Diverse network nodes

Figure 8.6 Ramifications on CPP due to mobile network domain.

The mobility attribute of mobile networks results in a dynamic topology and potential network disruptions in addition to load variance. This challenge needs to be addressed via adaptive and dynamic controller provisioning in the network. It complicates the problem structure with spatiotemporal changes in system parameters used in placement algorithms.

8.5 Conclusion

Current mobile networks suffer from complex control plane protocols, difficulties in deployment of new technologies, vendor-specific configuration interfaces, and expensive equipment. However, the wireless applications and services have become indispensable with the ever-increasing traffic volumes and bit rates. Therefore, mobile network infrastructure is supposed to adapt and evolve to address this stringent level of requirements in a sufficient and flexible manner. In that regard, the concept of SDMN is expected to be instrumental and emerge as an integral part of future mobile networks. Although SDN paradigm will facilitate new degrees of freedom in mobile networks, it will also bring forth profound issues related to mobile network characteristics. In that regard, the centralized controller and how to place it is a key issue for SDMN design and operation. Therefore, the controller placement-related challenges emerge as critical elements for the feasibility of SDMN. For practical SDMN, the controller placement algorithms have to be devised, which considers scalability, complexity, mobile network characteristics, and compatibility with generic SDN systems.

References

[1] Mendonca, M., Nunes, B. A. A., Nguyen, X., Obraczka, K., and Turletti, T. (2014) A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys and Tutorials, vol. 99, pp. 1–18.

[2] Fernandez, M.P. (2013) Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 1009–1016.

[3] Limoncelli, T. A. (2012) Openflow: A Radical New Idea in Networking. Communications of the ACM, vol. 55 no. 8: 42–4.

[4] Li, E., Mao, Z. M., and Rexford, J. (2012) Towards Software Defined Cellular Networks. Software Defined Networking (EWSDN), European Workshop 2012, Darmstadt, Germany.

[5] Cisco Visual Networking Index (VNI): Global Mobile Data Traffic Forecast, 2013–2018 Report. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html. Accessed January 21, 2015.

[6] Pentikousis, K., Wang, Y., and Hu, W. (2013) MobileFlow: Toward Software-Defined Mobile Networks. IEEE Communications Magazine, vol. 51, no. 7, 44–53.

[7] Mahmoodi, T. and Seetharaman, S. (2014) On Using a SDN-Based Control Plane in 5G Mobile Networks. Wireless World Research Forum, meeting 32, Marrakech, Morocco.

[8] Ashton, M. and Associates (2013). Ten Things to Look for in an SDN Controller. https://www.necam.com/Docs/?id=23865bd4-f10a-49f7-b6be-a17c61ad6fff. Accessed January 21, 2015.

[9] Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R., and Boutaba, R. (2013) Dynamic Controller Provisioning in Software Defined Networks, Network and Service Management (CNSM), 2013 9th International Conference on, 18–25, Zürich,Switzerland.

[10] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012) On Controller Performance in Software-Defined Networks. In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot–ICE), vol. 54.

[11] Voellmy, A. and Wang, J. (2012) Scalable Software Defined Network Controllers. Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM'12, pp. 289–290, New York, NY, USA.

[12] Heller, B., Sherwood, R., and McKeown, N. (2012) The Controller Placement Problem. ACM HotSDN 2012, pp. 7–12.

[13] Mülec, G., Vasiu, R., and Frigura-Iliasa, F. (2013) Distributed Flow Controller for Mobile Ad-Hoc Networks. 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, pp. 143–146. Timisoara, Romania.

[14] Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A. (2012) Logically Centralized?: State Distribution Trade-Offs in Software Defined Networks. ACM HotSDN 2012, pp. 1–6.

[15] IEEE. (1999) IEEE standard for communication-based train control (CBTC) performance and functional requirements. IEEE Std 1474.1-1999, New York, USA.

[16] IEEE. (2005) IEEE standard communication delivery time performance requirements for electric power substation automation. IEEE Std 1646-2004, New York, USA.

[17] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., and Tran-Gia, Phuoc (2013) Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks. Proceeding of the 25th Int. Teletraffic Congress (ITC), Shangai, China.

[18] Zhang, Y., Beheshti, N., and Tatipamula, M. (2011) On Resilience of Split-Architecture Networks. IEEE GLOBECOM 2011, pp. 1–6.

[19] Hu, Y., Wendong, W., Gong, X., Que, X., and Siduan, C. (2012) On the Placement of Controllers in Software-Defined Networks, The Journal of China Universities of Posts and Telecommunications, vol. 19, no. 2, pp. 92–97.

[20] Behesti, N. and Zhang, Y. (2012) Fast Failover for Control Traffic in Software-Defined Networks. Next Generation Networking and Internet Symposium. IEEE GLOBECOM 2012, Anaheim, CA, USA, pp. 2665–2670.

[21] Hu, Y., Wendong, W., Gong, X., Que, X., and Shiduan, C. (2013) Reliability-aware Controller Placement for Software-Defined Networks. IFIP/IEEE International Symposium on Integrated Network Management (IM2013), Ghent, Belgium.

[22] Bertimas, D. and Tsitsiklis, J. (1993) Simulated Annealing. Statistical Science, vol. 8, no 1, pp. 10–15.