13

Load Balancing in Software Defined Mobile Networks

Ijaz Ahmad,[1] Suneth Namal Karunarathna,[1] Mika Ylianttila,[1] and Andrei Gurtov[2]
[1] *Center for Wireless Communications (CWC), University of Oulu, Oulu, Finland*
[2] *Department of Computer Science, Aalto University, Espoo, Finland*

13.1 Introduction

Load balancing is a composition of methods to distribute workload among multiple networks or network components such as links, processing units, storage devices, and users to achieve optimality in respect to resource utilization, maximum throughput, and minimum response time. It also helps to avoid overload and provide quality of service (QoS). In a situation where multiple resources are available for a particular functionality, load balancing can be used to maximize network efficiency and increase fairness in network resource usage while keeping a balance between QoS and resource usage.

Network load balancing started in the form of load balancing hardware, which were application neutral and resided outside of the application servers. These network-based appliances could load balance using simple networking techniques. For example, virtual servers were used to forward connections to the real server deploying bidirectional network address translation (NAT) to load balance among multiple servers. A simple load balancing scenario is shown in Figure 13.1, where virtual servers balance the load among multiple real servers to ensure high availability and QoS.

Today, load balancing technologies are used mostly in the IP layer and application layer having layer-specific load balancing and distribution mechanisms. In this chapter, the basics of load balancing are introduced with the commonly used load balancing technologies in legacy wireless networks and state their problems and challenges. Moving forward to SDN, how SDMN-based load balancing technologies can solve the challenges existing in the

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

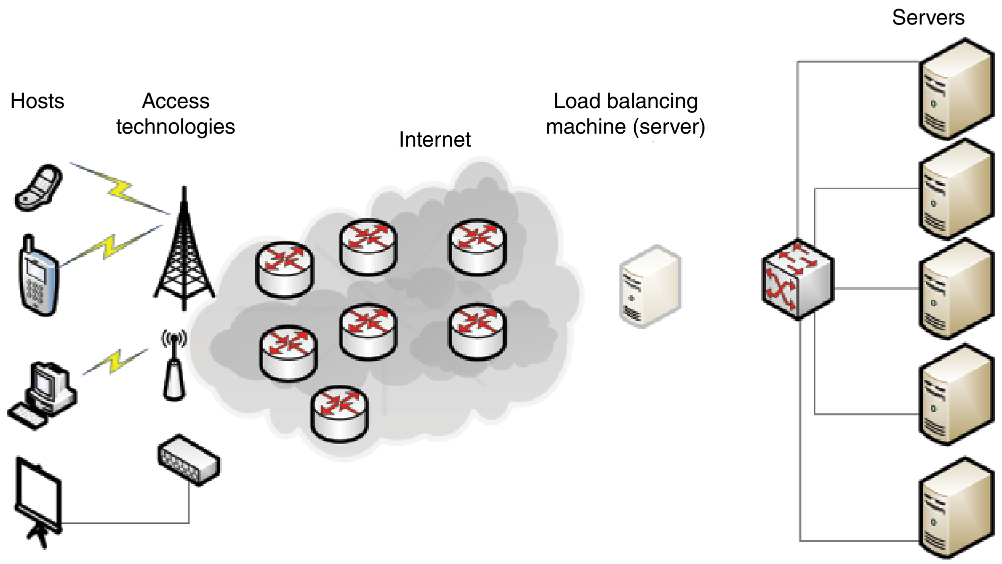


Figure 13.1 Load balancing among multiple servers.

currently used load balancing technologies in legacy wireless networks is elaborated. Toward the end of the chapter, future directions and research areas in load balancing in SDMN are discussed.

13.1.1 Load Balancing in Wireless Networks

In wireless networks, load balancing mechanisms are used to distribute traffic evenly among cells, nodes, and frequency bands to utilize resources of the network more efficiently. Highly loaded cells can be off-loaded to less heavily loaded neighboring cells, traffic on various backhaul or core network nodes can be shared among multiple nodes, and bandwidth can be dynamically shared to ensure QoS to subscribers. Since the aim of next-generation wireless networks is to provide high data rate services to mobile users in large coverage areas, bandwidth is a major consideration for operators to provide efficient services in dense and congested areas.

In order to use the available radio spectrum efficiently and effectively with high QoS, operators install small cells that significantly improve coverage and network capacity. These cells may use different technologies, such as cellular, WLAN, CDMA, or E-Band, to maintain the required QoS and quality of experience (QoE) through novel load balancing mechanisms. Technology convergence is an interesting approach to achieve high availability, fulfill QoS requirements, offer differentiated services, and provide network redundancy for network resilience. These goals of technology convergence can be achieved with the help of novel load balancing mechanisms for traffic and workload balancing.

Load balancing is a matured research topic that has been investigated for more than a decade in the context of mobile communications. However, due to vastly differing network architectures, intertechnology load balancing is mostly limited to researches because of the complexity in interoperability.

13.1.2 Mobility Load Balancing

In wireless networks, a user has the privilege to move around and still use network services. A mobile device can start or terminate connections randomly while traversing various cells or networks. Therefore, it is highly probable that a cell gets load of traffic that is beyond its capability with respect to its resources. Hence, mobility load balancing (MLB) is very important in cellular networks in particular and other wireless networks in general. MLB balances the load among available cells in certain geographical locations by means of controlling mobility parameters and configurations including UE measurement thresholds. MLB modifies handover (HO) regions to redistribute load between neighboring cells. The principle of MLB is adjusting the HO regions by biasing HO measurements, causing users in a cell edge to migrate from highly loaded cells to less heavily loaded neighboring cells to improve efficiency of resource utilization. Since the load redistribution is carried out automatically between neighboring cells, this MLB is an important feature of Self-Organizing Networks (SON).

13.1.3 Traffic Steering

Traffic steering is the capability of a network to control and direct voice and data traffic to the best suitable cell or radio technology within a network. It can be deployed in multiple layers such as frequency layers or hierarchical layers of cells (macro-, pico-, or femtocells) to provide resources to an end user in a certain geographical area. Traffic steering could optimize the network capacity and user experience through efficient utilization of the available pool of resources from a multitude of coexisting networking technologies in the core and edge. Traffic steering can be used to help MLB in a network. It can also be used to off-load macrocells toward low-power cells, HeNB, or Wi-Fi to accommodate large part of the traffic demand and minimize eNB power consumption. The primary challenge for traffic steering is coordinating mobility configurations in multiple overlaid cells.

13.1.4 Load Balancing in Heterogeneous Networks

Today, a typical smartphone can connect to the Internet via several different radio access technologies including 3GPP-standardized and non-3GPP technologies such as Wi-Fi (802.11×). Cellular base stations are getting diverse to satisfy user experience. Macrocells are shrunken to microcells, and picocells, distributed antennas, and femtocells are added continuously in cellular networks. Since the currently deployed networks are already dense in terms of nodes or base station installations, cell splitting is not a viable solution due to high intercell interference and costly capital expenditures (CAPEXs). Hence, the solution inclines toward overlaid structures where different varying architectures are overlaid to cowork and cooperate. These heterogeneous architectures would essentially use separate spectrum and different network architectures and topologies. With the introduction of heterogeneity from many directions in wireless networks, load balancing is crucial for the end user experience and overall system performance.

13.1.5 Shortcomings in Current Load Balancing Technologies

Load balancing is a critical requirement in large commercial networks, traditionally achieved with load balancers, which are expensive and independent entities in most of the cases. Generally, commercial load balancers sit on the path of incoming requests and then spread requests over several other servers. Current load balancing algorithms assume that the requests are entering to the network through a single gate where the load balancer is placed though there can be several such choke points in a large network. On the other hand, servers and data centers may dynamically move across the network by means of virtualization introduced with programmability. Furthermore, different network sections may need totally different load balancing or optimization techniques to achieve the expected results.

It is clear that the traditional load technologies are not capable of meeting the requirements in today's large commercial networks. Thus, a different approach of load balancing is needed where the functions could come out of the box and deploy on top of traditional network elements to enable load balancing based on network and server congestion in an intelligent manner. Therefore, next-generation load balancers must have the following characteristics: (i) load balancing as a property of a network over traditional network elements, such as switches and routers; (ii) flexibility in application and service level (load balancing in application and service level that enhance ability to experiment new algorithms); (iii) dynamicity in terms of the ability to adapt to the changing conditions of the network where server congestion and route remapping are required; and (iv) dynamic configuration management to automatically adapt and scale with changes in network capacity, such as virtual machine (VM) mobility and data center mobility. Thus, load balancing in modern and future networks needs dynamism, which can be brought about by SDN through global visibility of the network state and open interfaces for programmability.

Current load balancing methods make a number of assumptions about the services that are not valid in the current requirements of higher data rates, need of seamless mobility, high availability, and expected and offered QoS. These assumptions [1] are as follows:

- Requests enter the network through a single point where load balancing devices can be placed at a choke point through which all the traffic must pass. This condition might not work for all networks, and hence, operators end up with congestion while using these expensive devices. In enterprise networks, there can be many choke points such as egress connections to the WAN, campus backbones, and remote servers.
- The network and servers are static, which can be true for a data center but not for wireless networks and enterprise networks. For example, in wireless networks, a base station can get congested at any time due to user movements, changes in channel conditions, etc. Similarly, operators of data centers move VMs of virtualized data centers to efficiently use their servers. With these changes, load balancers need to track changes in the network and movements in data centers to direct requests to the right places.
- Congestion is at the servers but not in the network, which might be true for data center hosting only one service, whereas, in cloud data centers, the network may be congested differently at different places.
- The network load is static, and hence, the load balancers spread traffic using static schemes such as equal-cost multipath (ECMP) routing. Such load balancing is suboptimal since some parts of the network might be heavily loaded and hence can be congested.
- All services require same load balancing algorithms, meaning that HTTP and video request can be served with same load balancing schemes. This is not feasible due to varying nature of requirements of different services such as bandwidth, mobility, and link capacity requirements. It is also difficult to provide each service its own type of load balancing, since in virtualized data centers, more and more services will be deployed by different users and they will be moving around.

Since the current networking technologies have no centralized control and lack global visibility, MLB is yet a challenge. In cellular networks, handoff is initiated by eNB with the help of measurements from the UE. These eNBs are weakly coordinated in terms of loose centralized control and visibility of near-cell traffic load or resource usage. Similarly, intratechnology mobility is not yet in practice, and hence, load balancing in heterogeneous networks (HetNet) cannot be materialized for better user satisfaction and efficient resource usage.

13.2 Load Balancing in SDMN

SDMN drives the motivation toward load balancing with the logically centralized intelligence or network operating system (NOS), which is capable of interoperability between different systems or network. SDMN enablers such as OpenFlow [2] introduce common programmable interfaces over which various network entities can talk regardless of the underlying technology. Added to that, replacement of network entities with software applications can substantially reduce the network cost and improve flexibility. In SDMN, load balancing mechanisms would enable to harvest the benefits of low-cost heterogeneous networking technologies to work in parallel with cellular networks. Even though spectrum scarcity is a major issue faced by the cellular network operators, cellular networks are still not capable to utilize the locally available wireless networks due to lack of efficient load balancing technologies.

The common centralized control plane in SDMN would enable redirecting network traffic through lower load middleboxes, links, and nodes. A common control plane in SDMN would be like the one shown in Figure 13.2. All the logical control plane entities such as Mobility

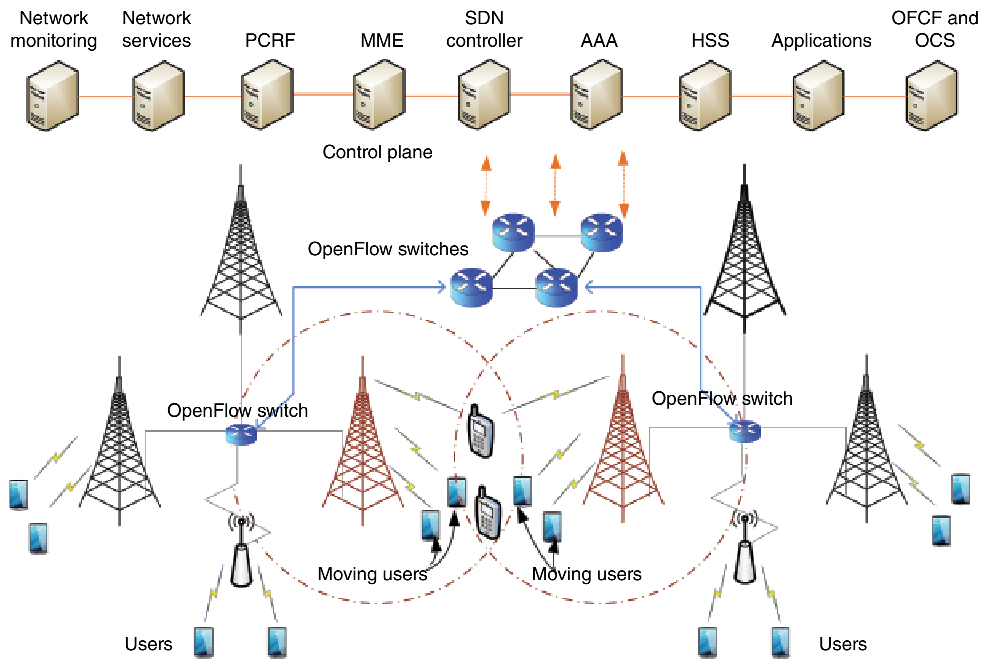


Figure 13.2 Software defined mobile network.

Management Entity (MME), AAA, PCRF, HSS, etc. are logically centralized and placed in high-end servers. These entities would (re)direct the data plane in real time. SDMN enables load balancing algorithms to be installed in the application server as a load balancing application in the control plane. The SDMN controller will fetch network load statistics from the data path and provide these statistics such as packet and byte counter values to the load balancing application. Similarly, UE mobility reports from MME would be provided to the load balancing application. Hence, centralized load balancing decisions based on the global view of the real network load would be taken.

OpenFlow enables flow-based routing and network virtualization with its extensions. The legacy network elements could be programmed with OpenFlow to enable flexible forwarding and management of commercial networks. This novel packet forwarding mechanism could be utilized for load distribution among technologically different systems as far as they are managed by a single controller or internetworked set of distributed controllers. The ability to move forwarding intelligence out of legacy network elements to a logically centralized control plane with an efficient forwarding approach is the significance behind OpenFlow. On the other hand, efficient load balancing is a counterpart of intelligent HO between the stations that are managed by the same NOS though they can be in technologically isolated domains.

13.2.1 The Need of Load Balancing in SDMN

Software defined networks are about a centralized control plane controlling and manipulating the forwarding behavior of the data plane from a logically centralized network view. Hence, load balancing is very important in SDMN to maintain a fair trade-off between multiple control plane devices.

13.2.1.1 Server Load Balancing

SDN enables applications to interact with and manipulate the behavior of network devices through the control layer. Applications benefit from the visibility of resources and therefore can request the states, availability, and visibility of the resources in specific ways. Network operators and service providers desire to control, manipulate, manage, and set policies by using applications for various network control, configurations, and manipulation options. These applications are deployed in operator's clouds implemented on high-end servers, which must be available to the increasing number of users and applications or services. Hence, server load balancing is required to ensure high availability to client requests and scalability by distributing application load across multiple servers. Besides that, server load balancing is of particular interest in SDMN since the control plane functionalities could be deployed on custom-belt logically centralized servers.

13.2.1.2 Control Plane Scalability

In SDNs, control functions, such as load balancing algorithms, require writing control logic on top of the control platform to implement the distribution mechanisms and deploy them on the forwarding elements such as switches and routers as shown in Figure 13.3. The control

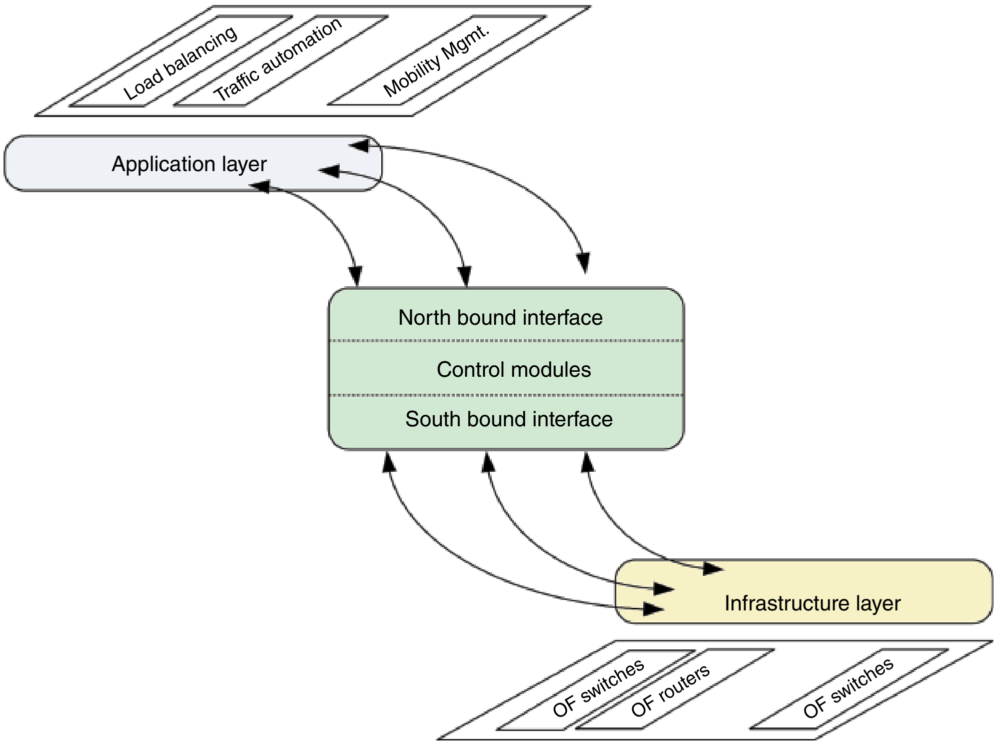


Figure 13.3 SDN-abstracted control plane.

platform of SDN can be either distributed or centralized, and the entity that implements the control plane functionalities is referred to as the SDN controller. The SDN controller is responsible for managing and controlling the whole network through an NOS from a central vantage point having global view of all the network resources.

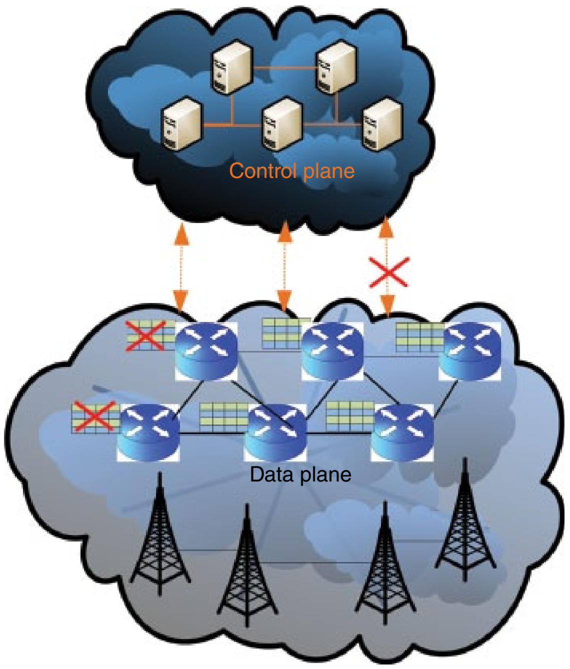
However, centralization of the control logic of networks opens up its own type of challenges. Control plane scalability is one such challenge, which can be solved through efficient load balancing technologies. In SDN, most of the complexity is pushed toward the controller where forwarding decisions are taken in a logically centralized manner. A challenge for the currently available controller implementations is specifying the number of forwarding devices to be managed by a single controller to cope with the delay constraints. If the number of flows on the controller increases, there is a high probability that the sojourn time will increase, which is deeply dependent on the processing power of the controller. Today's controller implementations are not capable to handle the huge number of new flows when using OpenFlow in high-speed networks with 10 Gbps links [3]. This also makes the controller a favorite choice for denial of service (DoS) and distributed DoS attacks by targeting its scalability limitation. Therefore, controller efficiency has been the focus of many researches to enhance its performance through control platform distribution, devolving and delegating controller responsibilities, increasing memory and processing power of the controller, and architecture-specific controller designs. Proper load balancing mechanisms will enable the control plane platform to cost-effectively handle situations where scalability failure could be detrimental to the whole network performance. Simply increasing the number of controllers will not help mitigate the risk of single point of failure as shown in Ref. [4] where the load of the failed controller is distributed among other controllers. The load must be put on the controller having the least original load so that all the controllers share the workload and increase overall system efficiency. Such distribution of load requires efficient load balancing methodologies.

13.2.1.3 Data Plane Scalability

Data plane enables data transfer to and from users, handling multiple conversations across multiple protocols, and manages conversations to/from remote peers. SDN enables remote control of data plane, making it easy to deploy load balancing mechanisms in the data plane via a remote procedure call (RPC).

OpenFlow abstracts each data plane switch as a flow table (shown in Fig. 13.4b), which contains the control plane decisions for various flows. The switch flow table is manipulated by the OpenFlow controller using the OpenFlow protocol. One challenge for SDN is that how

(a)



(b)

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

Figure 13.4 (a) The SDN data plane. (b) OpenFlow switch flow table.

efficiently the forwarding policies can be set from a logically centralized control plane on the forwarding devices. A scenario is shown in Figure 13.4a, where the SDN switches acquire flow rules from the controller. If the controller–switch path has higher delay, resources in the switches can be exhausted. For example, a switch has limited memory to buffer TCP/UDP packets for flow initiation until the controller issues the flow rules. Similarly, if a link to the controller is congested or the controller is slow in installing flow rules due to any reason (e.g., fault), the switch resources might be already occupied to entertain new flows. Besides that, recovering from link failure can take longer than the required time due to the centralized control plane. These challenges necessitate using OpenFlow switches according to their capacities through novel load balancing technologies.

13.2.2 SDN-Enabled Load Balancing

13.2.2.1 The Basis of Load Balancing in OpenFlow

As we know that SDN separates the control and data planes in a network, the OpenFlow variant of SDN defines an application programming interface (API) on the data path to enable the control plane interact with the underlying data path. The controller uses packet header fields such as MAC addresses, IP address, and TCP/UDP port numbers to install flow rules and perform actions on the matching packets. The action set comprises of, for example, forward to a port, drop, rewrite, or send to the controller. Flow rules can be set for either microflow that matches on all fields or wildcard rules that have empty (don't care bits) fields. A typical switch can support larger number of microflow rules than wildcard rules since wildcard rules often rely on expensive TCAM memory, while microflow rules use the SRAM, which is larger than TCAM. The rules are installed either with a fixed timeout that triggers the switch to delete (called the hard timeout) or with specified time of inactivity after which they are deleted (called the soft timeout). The switch also counts the number of bytes and packets for each rule, and the controller can fetch these counter values as shown in Figure 13.5.

The most basic mechanisms of load balancing in OpenFlow can use these counter values from the switches to determine how much load a switch is handling. Thus, the traffic load on various switches can be easily seen in the control plane, which can enable the controllers to load balance among the switches by using various coordination mechanisms. Load balancing in OpenFlow can also use the choice of using either the wildcard rules matching mechanism or microflows matching. Microflows matching would require the controller to be involved in small flows rather than aggregated flows and hence use more resources of the control plane. There can be a trade-off among the wildcard matching and microflows matching based on the controller load and availability. If matching on microflows is necessary, other mechanisms such as distributed control plane architectures can be used. That would require load balancing mechanisms among the controllers.

In the OpenFlow standard of SDN, a controller installs separate rule for each client connection, also called "microflow," leading to installation of a huge number of flows in the switches and a heavy load on the controller. Therefore, various approaches are suggested to minimize the load on the controller. These include using the wildcard support in the OpenFlow switches so that the controller directs an aggregate of client requests to server replicas. The wildcard mechanisms exploit the switch support for wildcard rules to achieve higher scalability

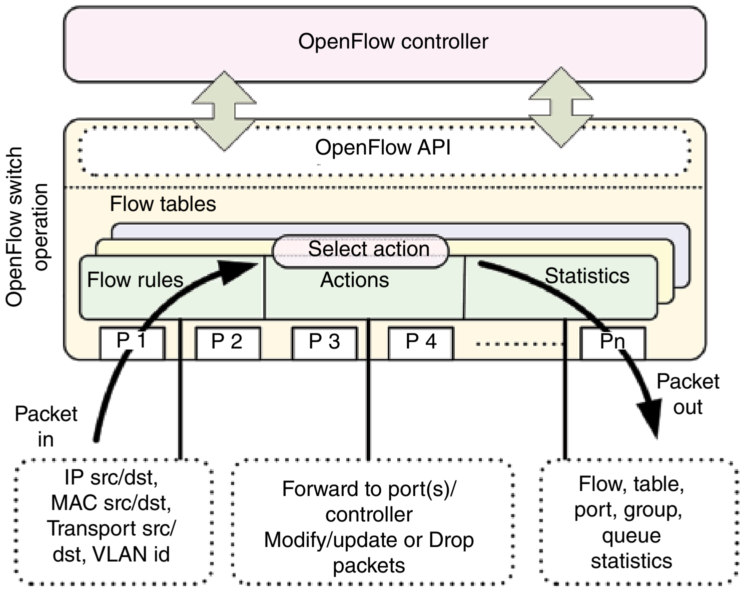

OpenFlow controller
OpenFlow API
OpenFlow switch operation
Flow tables
Select action
Flow rules
Actions
Statistics
P 1
P 2
P 3
P 4
Pn
Packet in
Packet out
IP src/dst, MAC src/dst, Transport src/ dst, VLAN id
Forward to port(s)/ controller Modify/update or Drop packets
Flow, table, port, group, queue statistics

Figure 13.5 Architecture of OpenFlow switch.

besides maintaining a balanced load on the controller. These techniques use algorithms that compute concise wildcard rules that achieve target distribution of the traffic and automatically adjust to changes in load balancing policies without disturbing existing connections.

13.2.2.2 Server Load Balancing

Data centers host a huge variety of online services on their servers. These services can also be offered to other operators for CAPEX and OPEX cost savings. Since the magnitude of services in normal data centers is huge, these data centers use front-end load balancing technologies to direct each client request to a particular server replica. However, dedicated load balancers are expensive and can easily become a single point of failure and congestion. The currently used variant of SDN, that is, the OpenFlow [2] standard, provides an alternative solution where network switches divide traffic among the servers. An OpenFlow controller installs the packet handling rules in OpenFlow switches on run-time and can change these rules immediately if changes are required.

OpenFlow-based server load balancing is proposed in Ref. [5] for Content-Centric Networks (CCNs). One of the important functionalities in server load balancing is imposing policies that balance client requests in CCNs. Server load balancing in Ref. [5] proposes three load balancing policies to balance client request on servers. The first policy uses per client request scheme that maps every new request to a fixed content server. This client-based policy forwards the Address Resolution Protocol (ARP) reply of the least loaded server to a new client that initiates the ARP request. In case the client is not new, then the same server will reply to the user.

The second policy balances the load based on the OpenFlow switch statistics, which are checked periodically by the OpenFlow controller, and the load is estimated. The controller in this case finds the statistics of the amount of data sent through the existing flows and estimates the load of traffic handled from each server. Hence, the traffic is distributed among the available content servers using this load-based policy. Whenever an overloaded server is detected, the most demanding content requests are switched to another less congested server leading to efficient distribution of traffic among all the servers. The third policy is proximity based in which clients are assigned to the servers having the most quick response using first-come, first-served technique. However, this technique is useful in low network traffic having negligible traffic delays [5].

Use Case: Live VM Migration
Live VM migration provides an efficient way for data centers to perform load balancing by migrating VMs from overloaded servers to less heavily loaded servers. Administrators can dynamically reallocate VMs through live VM migration techniques without significant service interruptions. However, live VM migration in legacy networks is still limited due to two main reasons. First, live VM migration is limited to LAN since the IP does not support mobility without session breakups. Second, network state is unpredictable and hard to control in the current network architectures.

SDN enables live VM migration since the control plane is centralized having global visibility of the network and is independent of the layered IP stacks. Since the SDN controller has information about the underlying network topologies, SDN-based VM migration would diminish the chances of migration breakups due to topological complexities existing in legacy networks. For example, to migrate a VM, the new end-to-end forwarding paths can be easily established without interrupting the service by pushing new forwarding rules in the switch flow tables. Modifying the existing flow rules in OpenFlow switches would hardly require temporary storage of the existing flow packets compared to session breakups in the current networking environments.

SDN had made it possible to migrate a whole system comprising of VMs, the network, and the management system to a different set of physical resources. For example, the LIve Migration Ensemble (LIME) [6] design leverages from the control–data plane separation logic of SDN to migrate an ensemble of VMs, the network, and the network management. LIME clones the data plane state to a new set of OpenFlow switches and then incrementally migrates the traffic sources. OpenFlow-based interdomain VM migration is illustrated in Ref. [7] where it is shown that OpenFlow data center can be configured on the fly regardless of the complexity of its topology.

13.2.2.3 Load Balancing as SDN Applications

Online services, network function applications, and management plane functionalities are implemented in the application plane in SDNs. These application plane functionalities are implemented on high-end servers. To properly load balance among multiple servers, front-end load balancing mechanisms could be used, which typically directs various requests to the right servers and their replicas.

Most of the load balancing mechanisms in SDN reside in the SDN application plane working on top of the control plane. For example, Aster*x [1] is a NOX application that uses the OpenFlow architecture to measure the state of the network and directly control the paths taken by flows. As shown in Figure 13.6 [1], the Aster*x load balancer relies on three functional units:

- *Flow Manager*: This module manages the routes of flows based on the chosen load balancing algorithm.

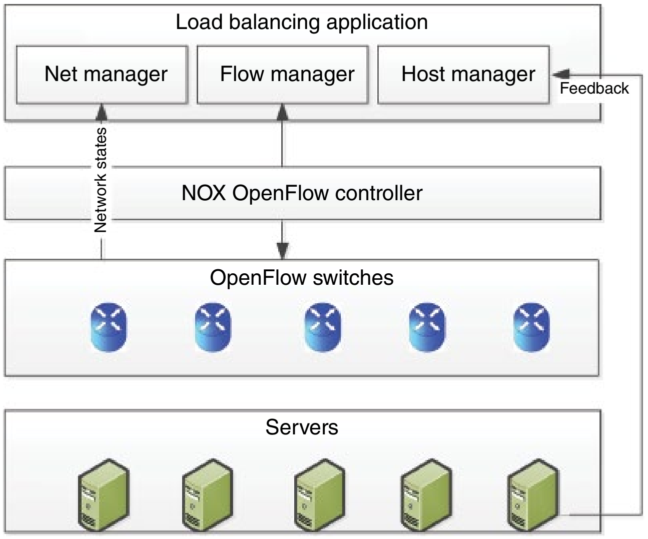


Figure 13.6 Aster*x load balancing architecture and its functional units.

- *Net Manager*: This module is responsible for keeping track of the network topology and its utilization level.
- *Host Manager*: This module keeps track of the servers and monitors their state and load.

Aster*x enables service providers to load balance their network based on different types of applications. The options that applications have include proactive versus reactive load balancing, load balancing on individual versus aggregated flow requests, and static versus dynamic load balancing. These choices make Aster*x a scalable distributable load balancing architecture.

13.2.2.4 Control Plane Load Balancing

In SDN, the controller implementing the control plane functionalities installs flow rules in the data path. Since a controller can set up a limited number of flows in the forwarding devices, it is suggested to use multiple controllers working in a logically centralized fashion. Hence, the latest versions of OpenFlow support multiple controllers in a single network domain where switches can have simultaneous connections to multiple controllers. Therefore, distributed OpenFlow controller architectures such as HyperFlow [8] and Onix [9] are proposed to implement multiple controllers to manage large networks. Load balancing among such distributed controllers plays a vital role to maintain a fair workload distribution of the control plane and ensure quick response. Load balancing in such scenarios will also enable maximum aggregate controller utilization and mitigate the risks of controller being a single point of failure or bottleneck.

Distributed Control Plane

BalanceFlow [10] is a controller load balancing architecture for wide-area OpenFlow networks. BalanceFlow works at the granularity of flows and partitions control traffic load among multiple controller instances in a large network. All the controllers in this architecture maintain

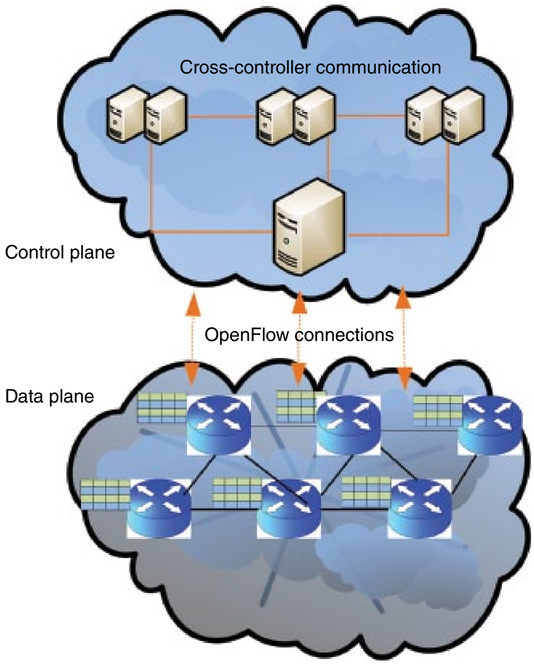


Figure 13.7 Architecture of the BalanceFlow controller.

their own load information, which is published periodically with other controllers. The controller architecture is hierarchical where one controller acts as a supercontroller to keep a balance of load on the rest of the controllers in the domain. When the traffic conditions change, the supercontroller partitions the traffic and allocates controllers to different flow setups to maintain a balance of workload among the working sets of controllers. This architecture also minimizes the flow setup delay since the nearest controller to the switch is allocated for the flow setup in the switch. Figure 13.7 shows the BalanceFlow load balancing architecture.

The BalanceFlow architecture has two requirements, that is, simultaneous multiple controller connections and controller X actions extension in the OpenFlow switches. The controller X extension in the switches allows sending flow requests to particular controllers. A controller, let's say controller k, maintains an $N \times N$ matrix M_k, where N is the number of switches in the network. Elements in the ith row and jth column denote the average number of flow requests from switch i to switch j. When a flow request packet is received, the controller first learns the switch from which the packet has arrived. After checking the destination address of the packet, the controller locates the corresponding egress switch for that flow, and the relevant element in the matrix is updated periodically. The average number of flow requests from switch i to switch j is calculated using the following formula:

$$R_{\text{avg}}(i,j) = (1-w)R_{\text{avg}}(i,j) + wT(i,j) \tag{13.1}$$

where w is the weighted coefficient and $T(i, j)$ is the number of flow requests from switch i to switch j in a certain period of time. The supercontroller collects the flow request matrixes from all the controllers and calculates the average number of flow requests handled by each controller. After calculating the total number of flow requests in the whole network, the supercontroller reallocates different flow setups to different controllers.

Control–Data Plane Load Distribution

Another approach for load balancing in SDN is to devolve some of the control plane responsibilities back to the data plane. Devolved OpenFlow or DevoFlow [11] is one such example. The main idea behind developing such architectures is the implementation costs of involving the control plane too frequently. For example, in OpenFlow, the controller might be required to install flow rules and gather switch statistics (byte and packet counters) in very quick successions. Hence, the control plane working at such granularity would hinder the deployment of SDN architectures in large-scale deployment.

Two mechanisms are introduced to devolve the control from controller to a switch. The first is rule cloning and the second one is localizing some of the control functions in the switch. For rule cloning, the action part of wildcard rules in OpenFlow packets is augmented with a Boolean CLOONE flag. If the flag is clear, the switch follows the normal wildcard mechanisms; otherwise, the switch will locally clone the wildcard rule. The cloning will create a new rule by replacing all the wildcard fields with values matching the microflow and inheriting other aspects of the original rule. Hence, subsequent packets for the microflow will match the microflow-specific rule and thus contribute to microflow-specific counters. This new rule will be stored in the exact-match lookup table to minimize the TCAM power cost. In the local action set of DevoFlow, local routing actions are performed by the switch instead of involving the controller. The set of local actions include multipath support in the switch and rapid rerouting in the switch. DevoFlow enables multipath routing by allowing the clonable wildcard rule to select an output port for a microflow according to some probability distribution. The rapid rerouting would enable a switch to use one or more fallback paths if the designated output port goes down.

Load Balancing in Case of Controller Failure

In SDN, it is highly probable that the network fails due to a controller being the single point of failure. To avoid single point of failures, the use of multiple controllers is suggested. However, proper load balancing is required to redistribute the traffic of the failed controller among other controllers. Otherwise, if the load is evenly distributed, a controller already loaded to its capacity will fail, and such process can lead to cascading failures of the controllers [4]. Therefore, the optimal strategies for handling the controller failure in multicontroller environment must satisfy the following requirements:

- The whole network must have enough capacity to tolerate the load of a failed controller.
- The initial load must be balanced with respect to the capacity of controllers.
- After failure of a controller, the load redistribution must not cause overload to another controller having being already working to its full capacity. Rather, proper load balancing algorithms must be used to deploy less extra load on heavily loaded controllers and vice versa.

13.2.2.5 Data Plane Load Balancing

Open Application Delivery Networking (OpenADN) [12] enables application-specific flow processing. It requires packets to be classified into application flow classes using cross-layer communication techniques. The cross-layer design allows application traffic flows' information

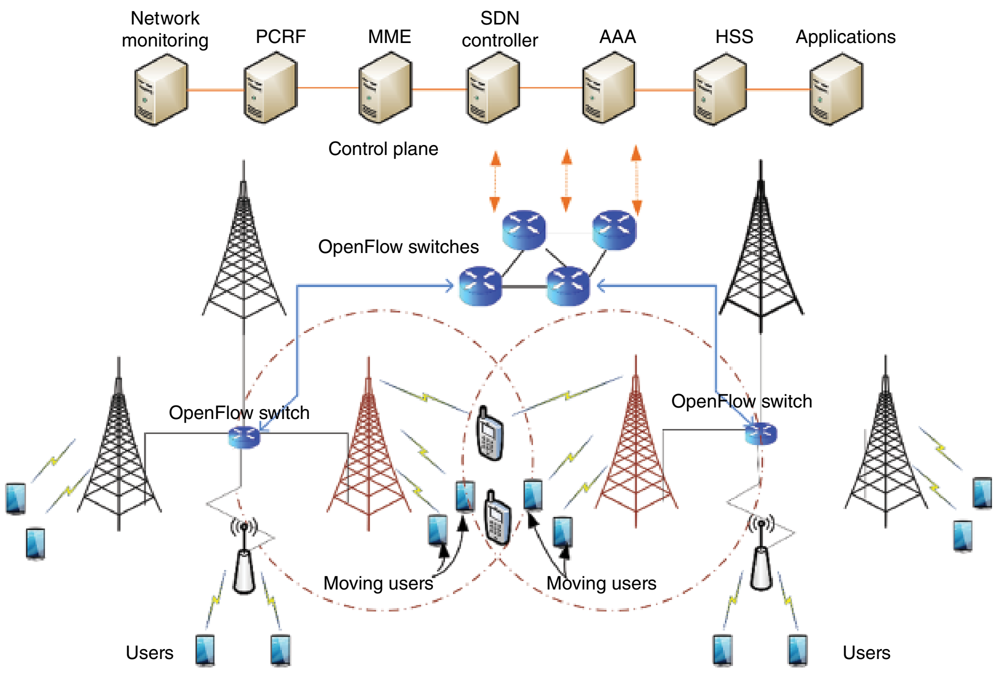


Figure 13.8 Software defined mobility load balancing among cells.

to be placed in the form of a label between network and transport layers. This Application Label Switching (APLS) layer forms layer 3.5 that is handled by the OpenADN switches enabling application traffic to be handled at the packet layer. Hence, it is possible to enable flow-based load balancing in the OpenFlow switches.

13.2.2.6 MLB

SDN provides a common control protocol such as OpenFlow that works across different wireless technologies with minimal changes. This capability of OpenFlow has made the integration of SDN into the current wireless networks straightforward where the data path remains the same, but the network control and logical elements such as MME, PCRF, and control part of SGW/PGW are abstracted in the control plane as shown in Figure 13.8. This makes it easy to use the available standardized mobility mechanisms in the radio part with the SDN-featured control plane having novel MLB algorithms implemented on top of the control plane. MLB in legacy networks followed by SDMN-enhanced features is described in the following text.

MLB in Cells

Load balancing in the cells of cellular is carried out with the help of eNBs. The aim of load balancing through eNBs is to keep a balance of load among the neighboring cells in order to improve the overall system capacity. Hence, load information is shared among the eNBs for maintaining a fair distribution of workload among the pool of eNBs. The information is shared via the X2 interface directly between eNBs since there is no central radio resource management (RRM) system in LTE. Generally, the exchange of load information falls into two categories depending on the purpose it serves. First, the exchange of load information is used to perform load balancing on the X2 interface where the frequency of exchanging the information is rather low. Second, the exchange of information is used to optimize the RRM processes where the frequency of sharing the load information is high.

The load imbalance is detected by comparing the load of cells and then exchanging that information among the eNBs. The exchanged cell load information comprise of radio measurements corresponding to the usage of physical resource blocks (PRBs) and nonradio related such as processing or hardware resources usage. Normally, a server–client method is used for sharing the information among eNBs using the Resource Status Response and Update messages. This information is reported via X2 interface between the requesting eNB (client) and the eNBs that have subscribed to these requests (servers). This load reporting is periodic according to the periodicity expressed in the Resource Status Response and Update messages that triggers the procedure. A separate load indication procedure is used for sharing load information related to interference management. This information is also shared via X2 and has direct influence on some RRM process in real time.

The objectives of MLB can be achieved by adjusting the HO parameters between the overloaded cells and its neighboring cells. By adjusting the HO parameters, some UEs in the overloaded cells can be handed off to less loaded neighboring cells. The number of users and utilization of PRBs in a cell can be used to indicate the load and usage of the physical resources in LTE. Each base station (or eNB in LTE) measures its serving cell load. The distributed solution of MLB in 3GPP requires an eNB to cooperate with its neighboring eNBs via the X2 interface. Overloaded eNBs obtain its neighboring cell loads and adjust the HO parameters via X2 to force some of UEs to hand off from the current cell to the neighboring cells. In LTE, the HO decision is generally triggered by the event A3 simplified as

$$M_{\mathrm{n}} > M_{\mathrm{s}} + \mathrm{HO}_{\mathrm{margine}}, \tag{13.2}$$

where M_{n} is reference signal received power (RSRP) in dBm or reference signal received quality (RSRQ) in dB for a neighboring cell, M_{s} is RSRP or RSRQ of the serving cell, and $\mathrm{HO}_{\mathrm{margine}}$ is a margin between M_{n} and M_{s} in DB. Each cell can have its own value of $\mathrm{HO}_{\mathrm{margine}}$. HO decision is based on formula (13.2) by measuring these parameters in UEs. When an eNB detects that its serving cell is overloaded, the $\mathrm{HO}_{\mathrm{margine}}$ to its neighboring cells will be adjusted to trigger handoff of UEs from the current cell to the neighboring cell. For efficient and precise MLB, the measurement reports from UEs can be used to predict the cell loads after adjusting the HOmargine.

Since these measurement reports from UEs can contain M_{s} and M_{n} in the formula (13.2), the eNB can collect information of M_{s} and M_{n} of each UE located at cell edge between the serving cell and its neighboring cells. Hence, the eNB can measure PRB utilization of UEs at the serving cell. Therefore, the PRB utilization at the neighboring cell can be estimated in accordance with the user throughput and modulation and coding scheme on the neighboring cell. This would enable to allocate users to the neighboring cells while not congesting the cell and effectively balance load among neighboring cells. However, in such mechanisms and techniques, the eNBs itself adjust the $\mathrm{HO}_{\mathrm{margine}}$ of the serving cell cooperating with its neighboring eNBs. The eNBs are required to cooperate and exchange information that makes the network rather complex and difficult to scale and maintain. SDN on the other hand centralizes all the control plane functionalities where centralized servers collect the network information and direct individual entities such as eNB to set HO parameters and perform HOs when required. An SDMN architecture having mobile users in neighboring cells is shown in Figure 13.8. Due to mobility, the PRB usage of resources will either increase or decrease. The centralized control plane in SDMN will collect information regarding usage of PRBs in neighboring cells and hence be able to easily compare the load in the two cells. Since implementing new functionalities in SDN requires writing software logic on top of the control plane, the mobility management algorithms can be implemented on top of the control plane, which will utilize global visibility of the network physical resources. For example, the MLB algorithm can be implemented as an SDN application that has real visibility of the physical resources of the neighboring cells and will be in better state to adjust the HO margins between neighboring cells. Another advantage of such centralized MLB would be to distribute the unbalanced cell load when several cells close to each other are overloaded.

MLB in MME

In cellular networks, a UE is associated with one particular MME for all its communications where the MME creates a context for that UE. The MME is selected by the Nonaccess Stratum (NAS) Node Selection Function (NNSF) in the first eNB from which the UE connected to the network. When a UE becomes active with an eNB, the MME provides the UE context to that eNB using the Initial Context Setup Request message. With the transition back to idle mode, a UE Context Release Command message is sent to the eNB from the MME to erase the UE context, which then remains only in the MME.

For mobility within the same LTE system or inter-eNB HO, the X2 HO procedure is normally used. However, when there is no X2 interface between the two eNBs or if the source eNB is configured to initiate HO toward a particular eNB, the S1 interface is used. In HO process through the X2 interface, the MME is notified only after completion of the HO process. The process of HO and control flow during HO on the S1 interface is shown in Figure 13.9. The HO process comprises of a preparation phase where resources at the core network are prepared for HO, followed by an execution phase and a completion phase. Since MME is actively involved in HOs and context maintaining, it is very important to perform load balancing among the MMEs in a cellular network.

The aim of MME load balancing is to distribute traffic among the MMEs according to their respective capacities. S1 interface is used to perform load balancing among MMEs in a pool of MMEs in cellular networks. MME carries out three types of load management procedures over the S1 interface. These include a normal load balancing procedure to distribute the traffic, an overload procedure to overcome a sudden rise in load, and a load rebalancing procedure to either partially or fully off-load an MME. The MME load balancing depends on the NNSF present in each eNB, which contains weight factors corresponding to the capacity of each MME node. A weighted NNSF carried out at each eNB in the network achieves statistically balanced distribution of load among MMEs. However, there are some specific scenarios that require specific load balancing actions. First, if a new MME is introduced, the weight factor corresponding to the capacity of this node may be increased until it reaches an adequate level of load. Similarly, if an MME is supposed to be removed, the weight factor of this MME should be gradually decreased so that it catches minimum traffic and its traffic must be

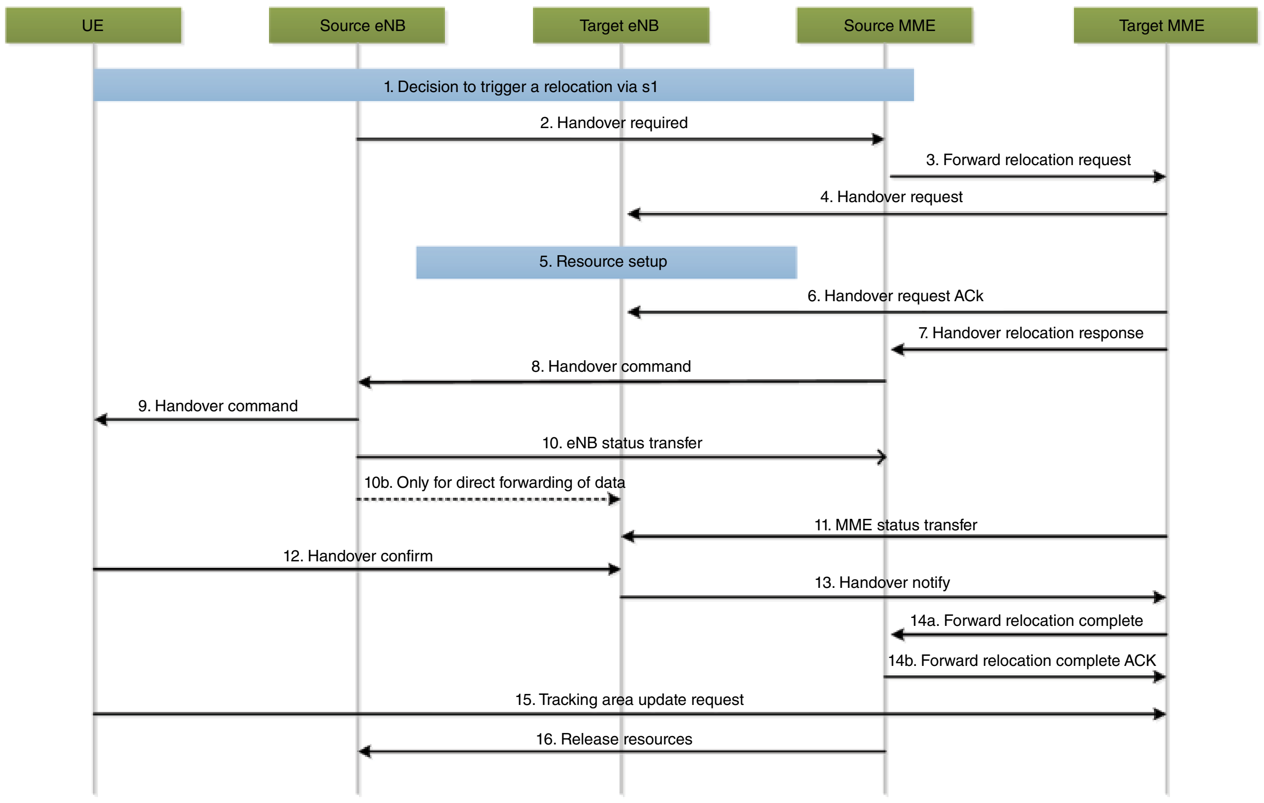


Figure 13.9 S1 handover control flow.

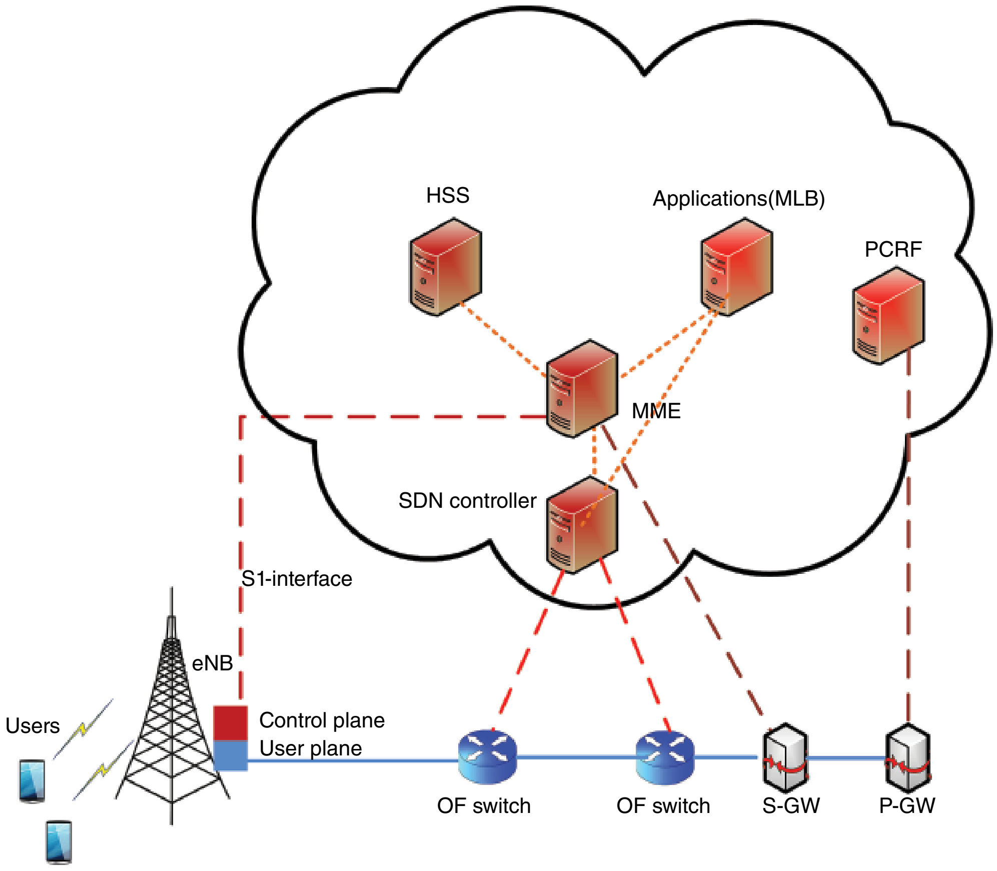


Figure 13.10 MME load balancing in SDMN.

distributed among the remaining MME nodes. Second, if there is an unexpected peak in the load, an overload message can be sent over the S1 interface to eNBs to temporarily restrict certain types of traffic to that particular MME. The MME can also adjust the number of eNBs and restrict the types of traffic it needs to avoid. Third, if an MME wants to rapidly remove the UEs, it will use the rebalance function to force UEs to reattach to other MMEs using a specific cause value in the UE Release Command S1 message [13].

In SDN, MME becomes part of the control plane and interacts with eNBs through the S1 interface. Along with MME, the control planes of the SGW and PGW are also abstracted to the control plane. Since MME is now a logical entity in the SDMN-centralized control plane, checking its load and maintaining a fair load on MME would be easy. There can be a separate application for load balancing among MMEs, or the MME load balancing algorithms can be part of the overall load balancing application. Since the current MME load balancing is dependent on the load measurement values of NNSF in the eNB, these values represent only the load of the MMEs, which are attached to that particular eNB. Thus, each eNB has limited visibility of those MMEs that are not listed in its NNSF. This makes the current MME load balancing rather inefficient. In SDMN, the load of all the MMEs in a certain geographic location would be gathered to the MME load balancing application. This could be done either through getting the load measurement values directly from the MMEs or through the S1 interface by fetching the measurement values (weight factors and current load) from the NNSF of all the eNBs. Hence, the MME load balancing in SDMN would be carried out in the presence of load values of all the MMEs. Such a load balancing scenario is shown in Figure 13.10.

13.3 Future Directions and Challenges for Load Balancing Technologies

Wireless networks are constrained from system capacity and user QoE. Therefore, various types of wireless networking technologies are proposed and used that comprise of varying cell sizes, differing architectures, and heterogeneous infrastructures. Each type of networking technology has its own limitations, and hence, a trade-off is always desired, which should be part of future load balancing technologies. Small cells such as femtocells, picocells, and Wi-Fi AP provide better data rates, but these data rates will most likely be backhaul constrained since the wired backhaul has a fixed capacity. Therefore, intelligence must be imbedded into the network to off-load macrocells to smaller cells while taking the backhaul constraint into account so that the cells are not loaded beyond a certain threshold.

Seamless mobility based on seamless HOs between various networks in a HetNet is very important to maintain fair load distribution among cells and provide the best possible services to end users. However, HOs involve signaling overhead costs with complicated procedures from network management point of views. Added to that, wireless networks are prone to instantaneous saturation and varying interference that might force the UE to hand over again, thus introducing a ping-pong behavior. Therefore, it may be preferable from a system-level view to temporarily tolerate a suboptimal base station than a ping-pong behavior. Power constraints cause another challenge for interoperable networks where small cell base stations have less power compared to macrocells. UEs can transmit at the same power level in uplink regardless of the base station type where the need of strong coordination among the femto-, pico-, micro-, and macrocell base station is very important for load balancing. Such asymmetries require a centralized control mechanism where load balancing mechanisms direct the base station and maintain a balanced load among the cells irrespective of their transmit power capabilities. 3GPP has initiated work items on device-to-device (D2D) communication allowing direct communication between cellular users. Such direct communication opens up new directions for load balancing technologies motivated by off-loading the load from cellular networks.

Although network security is an integral part of network management, it has been rarely researched in parallel to network load balancing. It is extremely important in SDMN to develop load balancing architectures that work according to network security policies. For example, security lapses of the controller can introduce delay in setting flow rules in the switches leading to congestion in switches with unsolicited traffic flows. Therefore, it is necessary to consider network security while designing and deploying traffic load balancing technologies in SDMN.

References

[1] Handigol, Nikhil, Mario Flajslik, Srini Seetharaman, Ramesh Johari, and Nick McKeown. "Aster*x: Load-balancing as a network primitive." In ninth GENI Engineering Conference (Plenary), Washington, DC, 2010.

[2] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: Enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review, vol. 38, no. 2 (2008): 69–74.

[3] Jarschel, Michael, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, and Phuoc Tran-Gia. "Modeling and performance evaluation of an OpenFlow architecture." In Proceedings of the 23rd International Teletraffic Congress, San Francisco, CA, USA, pp. 1–7, 2011.

[4] Yao, Guang, Jun Bi, and Luyi Guo. "On the cascading failures of multi-controllers in software defined networks." 21st IEEE International Conference on Network Protocols (ICNP), pp. 1–2, October 7–10, 2013. DOI:10.1109/ICNP.2013.6733624.

[5] Choumas, Kostas, Nikos Makris, Thanasis Korakis, Leandros Tassiulas, and Max Ott. "Exploiting OpenFlow resources towards a cContent-cCentric LAN." In Second European Workshop on Software Defined Networks (EWSDN), pp. 93–98. IEEE, October 10th–11th, 2013, Berlin, Germany.

[6] Keller, Eric, Soudeh Ghorbani, Matt Caesar, and Jennifer Rexford. "Live migration of an entire network (and its hosts)." In Proceedings of the 11th ACM Workshop on Hot Topics in Networks, pp. 109–114. ACM, Redmond, WA, 2012.

[7] Boughzala, Bochra, Racha Ben Ali, Mathieu Lemay, Yves Lemieux, and Omar Cherkaoui. "OpenFlow supporting inter-domain virtual machine migration." In Eighth International Conference on Wireless and Optical Communications Networks, pp. 1–7. IEEE, Paris, 2011.

[8] Tootoonchian, Amin and Yashar Ganjali. "HyperFlow: A distributed control plane for OpenFlow." In Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, pp. 3–3. USENIX Association, San Jose, CA, 2010.

[9] Koponen, Teemu, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, Scott Shenker. "Onix: A distributed control platform for large-scale production networks." Ninth USENIX Conference on Operating Systems Design and Implementation, vol. 10, Vancouver, BC, Canada, pp. 1–6, 2010.

[10] Hu, Yannan, Wendong Wang, Xiangyang Gong, Xirong Que, and Shiduan Cheng. "BalanceFlow: Controller load balancing for OpenFlow networks." In IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), vol. 2, pp. 780–785. IEEE, Hangzhou, 2012.

[11] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. "DevoFlow: scaling flow management for high-performance networks." In Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11), pp. 254–265. ACM, New York, 2011. DOI:10.1145/2018436.2018466.

[12] Paul, Subharthi and Raj Jain. "OpenADN: Mobile apps on global clouds using OpenFlow and software defined networking." In Globecom Workshops (GC Wkshps), 2012 IEEE, Palo Alto, CA, USA, pp. 719–723, 2012.

[13] Alcatel-Lucent, "The LTE network architecture: strategic white paper." Available at: http://www.cse.unt.edu/~rdantu/FALL_2013_WIRELESS_NETWORKS/LTE_Alcatel_White_Paper.pdf (accessed on February 19, 2015), 2013.