Load balancing of elastic data streams in cellular networks

Kaiyuan Wu

January 25, 2005

Contents

Chapter 1

Introduction

This Master's thesis focuses on minimizing the delay of data flows in the packet-switched cellular networks, including General Packet Radios Service (GRPS), Enhanced Data rates of Global Evolution (EDGE), Universal Mobile Telecommunications Service (UMTS) network and Wireless LAN (WLAN).

1.1 Background

Conventional voice call services and Short Message Service (SMS) are prevailing in people's daily life. And new data services like Multimedia Messaging Service (MMS) and video streaming are improving users' experience in the new generation of communication networks. For these boosting data services, the packet-switched network such as GPRS and EDGE are widely deployed nowadays to provide coverage in the urban and suburb areas to fulfill such service requests. Another active technology to provide such services is the WLAN. It offers much higher data access, rather increased flexibility.

In the long run, the development of communication network proceeds towards all-IP networks, where all the services are delivered via packet-switched networks [7]. Therefore, from the network's point of view, the data requests including pictures, video clips, files are the byte streams. These streams are also called data flows, meaning a stream of data packets that are delivered from a source to a certain destination.

Above Internet Protocol (IP), TCP is the main workhorse that provides a reliable, sequenced service on an unreliable network [17]. Two most potential problems that adversely affect the services performance are network capacity and receiver capacity. To avoid congestion from these capacity problems, a few mechanisms are defined to alleviate such fluctuations of the capacity by varying the transmission rate [17]. The fluctuation in data rate will cause the data flows to experience different rates during their service periods. This is why flows or TCP connections are elastic in terms of their tolerance in transmission rates.

TCP is originally designed to work over fixed networks, though in theory TCP should be quite independent of the technology that used by TCP's underlying protocol. However, deploying TCP over wireless networks brings some challenges: narrower bandwidth, longer delay and larger delay variations than in a fixed network [7]. Thus, more endevours are required to alleviate these drawbacks and deliver appealing data services to the users with reasonable performance.

1.2 Statement of the problem

The network in the thesis is assumed to be a packet-switched cellular network. In such networks, relatively small units of data containing the address of destination or packets, are transmitted. The advantage of breaking data into small units is that the same data path can be shared among all the packets that may belong to different data services. In the network's perspective, this mechanism is notably efficient compared with the circuit-switched network. In its counterpart, the connection is dedicated to an individual user after establishing the connection even though the user most probably cannot utilize the entire resources.

To simply model such networks, only two adjacent cells are considered in the thesis. Furthermore, the congestion control mechanism in TCP is assumed to function well enough so that network resources are statistically shared by the flows in the network. In a whole, we assume the network works in a Processor Sharing (PS) manner [3].

Nowadays, load balancing (LB) has been widely used to improve the system performance in the computing systems. Sometimes, jobs rush into the system, especially to a specific server and force it work quite heavily. In the meantime, some different servers might be rather idle even though there are still jobs waiting for the services. The motivation of the load balancing is to address such problems. The basic idea of LB is to distribute the jobs to several servers according to a certain scheme to make the entire system work more efficiently. Some works have been explored to apply this technique into the WLAN and adopting LB in cellular networks [13, 18] brings great improvement.

The main objective of the thesis is to propose an load balancing scheme for the data flows in the overlapping area of two adjacent cells in cellular networks. The aim of such scheme is to minimize the mean delay at the flow level. A reasonable LB scheme should achieve a considerably low system delay and is not difficult to derive. Applying such scheme can absolutely increase the network efficiency to improve the users' satisfaction.

1.3 Research method

The research problem is modeled after a literature review in the related fields. Queueing theory, especially $M/M/1 - PS$ queue along with the process optimization tool like Markov Decision Process (MDP) plays a very important role in the analysis of the model. Policy iteration in MDP is the main tool to continuously refine the process generated from the research problem. It is from these repeated operations that we seek a load balancing scheme to achieve a great progress in the flow delay performance.

Quite a few numerical experiments are implemented in Mathematica to support the previous analytic results. They also report the difference among various schemes obtained through heterogeneous algorithms.

Roughly speaking, this work can be divided into two parts. First, analytic results are examined after modeling the research problem. Secondly, numerical experiments are implemented to verify the analytic results and also to further study of various schemes in a systematic way.

1.4 Structure of work

The remainder of the thesis is organized like follows. Fundamental knowledge of different cellular networks are covered including both mobile networks and WLAN in Chapter 2. Chapter 3 presents the basic ideas of flow level traffic modeling and Chapter 4 introduces MDP, and an example of implementing MDP is demonstrated as a case study. In Chapter 5, the research problem is formulated along with some observations of the related model. The research problem is studied in the symmetric case in Chapter 6 and the asymmetric case is covered in Chapter 7. Furthermore, some numerical experiments for both cases are presented in Chapter 8 to provide supporting evidences for the analytic results and insight into various LB algorithms. Conclusions are drawn and future works for this research problem are also proposed in Chapter 9.

Chapter 2

Data traffic in cellular networks

The past few years have witnessed a rapid and significant development in the cellular networks: various new technologies have been developed and standardized, new components in networks have been widely deployed globally and different data services boom to improve the users' experience beyond imagination. For a long time, users expect to enjoy the services through mobile network as if they are connected to the wired network. Nowadays, this demand is manifested by the exponential growth of data services in wireless network, typically like receiving and sending mails and downloading data files. To meet the increasing needs, GPRS has been introduced by May 2001, which enables packet data rates of up to 20 kbps per time slot over the existing GSM network [12]. Ever-increasing demand in data traffic further contributes to the development of Enhanced GPRS (EGPRS), which is part of Enhanced Data rates for Global Evolution (EDGE). In theory, it triples the bit rate of a GPRS network because of adopting a new modulation method [12]. Some basic features of Universal Mobile Telecommunication Services (UMTS) network will be introduced briefly. Another prevailing and promising technology in communication area is the wireless LAN (WLAN). It has already provided higher speed data access and it is believed to be a complementary part of cellular network in 3G telecommunication system. At last, load balancing is explored along with a case study.

2.1 GPRS

GPRS is known as one of the cautious steps to the 3G mobile network, which features the packet-based communication services for mobile devices. Mainly because of its enhanced connection speed, GPRS is regarded to as a 2.5G technology upgrade, highlighting that such system bridges the gap between

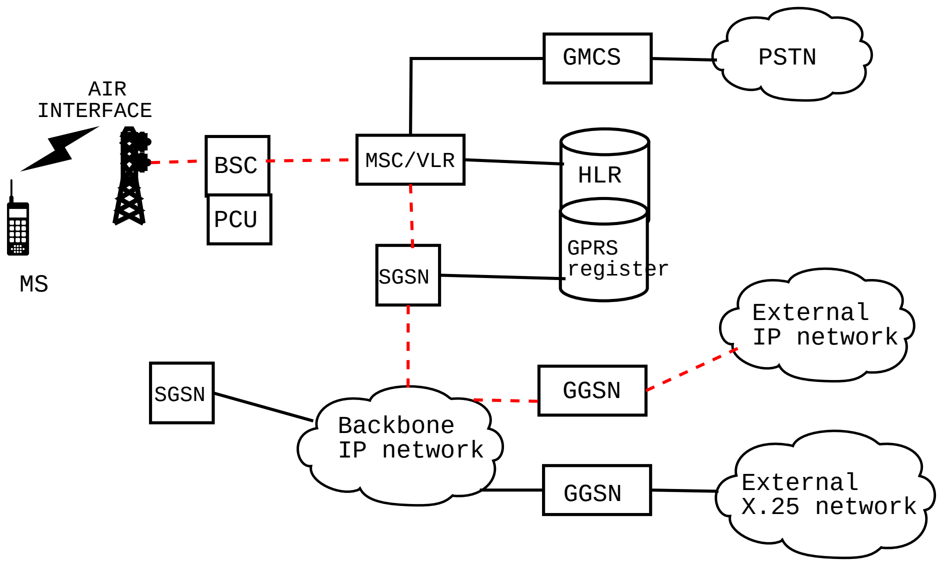


Figure 2.1: GPRS network architecture, the dashed line indicates the connection from MS to the Internet

2G and 3G system.

Circuit-switched connection is mostly suitable for real-time services such as conventional voice service because they are really sensitive to the delay. During such service, the connection is dedicated to an individual user even though most probably one fails to utilize one hundred percent of the resources. In this sense, circuit-switched connection is not ideal for the non real-time services when taking account of the network efficiency because such applications have rather loose delay requirements. Packet-switched connection is more efficient because a data path can be absolutely exploited by all the packets that may belong to different data services.

There are quite a few key advantages of GPRS over GSM:

1. Speed
   In theory, the maximum data rate of GPRS is 171.2 kbps combined with standard GSM time slots. In real use, the user can expect a speed rate of 20-50 kbps, which is a lot more better than the unsatisfactory 9.6 kpbs or 14.4 kbps in GSM.

2. Always-on connectivity
   Unlike in GSM, one does not need to dial up to access the data service.

3. More applications
   With the increase of the connection speed, GPRS enables most of the

Internet applications. And surely the experience of the usage is more pleasant than that in GSM.

4. Billing
   The user is charged according to the data volume that is used rather than the time when connection is established, most of which is waiting.

GPRS is an overlay network over GSM, meaning that GPRS keeps almost all the existing GSM hardware. In fact, many of the infrastructures in GPRS are shared with those in the GSM network, which implies that the operator can avoid deploying the GPRS network from scratch. This reuse of the GSM network can save considerable cost and makes it possible to rapidly and widely implement such packet-switched update.

Some modifications are required in software, hardware or even both in the Base Station Controller (BSC) , Mobile Switching Center (MSC) and some new components are required to support the new data services. Part of the changes can be seen in Figure 2.1. It shows the general architecture of GPRS network and actually the components of the GSM are still included in the graph, where they coexist and cooperate together to provide satisfying services to Mobile Stations (MS). The radio tower in the figure stands for the Base Station (BS) which transmits and receives the radio power to cover a geographical area and meanwhile provides various services to the MSs that are within its coverage via the air interface. Base Station Controller (BSC) takes care of a number of BSs to manage the radio resource allocation to these several BSs and collects necessary information about all the BSs under its control periodically.

The new components in GPRS are the Packet Control Unit (PCU), Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN). A brief review of these units is given below:

- PCU functions to differentiate the circuit switched data for the GSM network and packet switched data that is destined for the GPRS network.

- SGSN is in charge of the routing and IP address assignment. One SGSN controls a couple of Base Station Controllers (BSC). When the handset is moving through different cells, SGSN figures out which BSC the user belongs to and how to route the packets to the MS. Sometimes user may move to the cells that are in the charge of different SGSN. In such a circumstance, a hand-over will be performed to the new SGSN which is transparent to the MS.

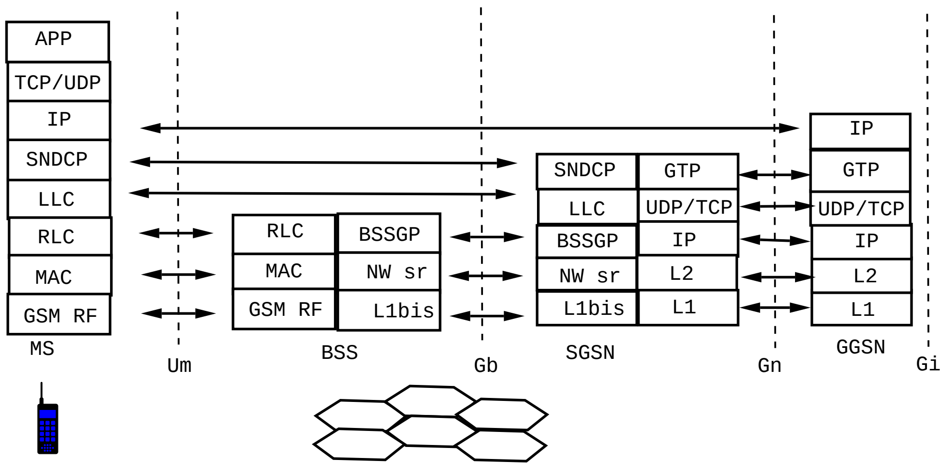


Figure 2.2: EGPRS protocol stack from MS to GGSN

- GGSN is the last part to connect to an Internet Service Provider (ISP) or company's network. Basically, it is a gateway, router and firewall combination.

In practice, former components of GSM network still handle the voice service while the GPRS network focuses on the data services. This coexistence is based on the fact there is no dedicated channel for the new GPRS sector.

The dashed line in Figure 2.1 indicates the connection from MS to the IP network, which illuminates that all the contents that the Internet offers is also obtainable via the GPRS network in theory. Actually, the main driving force behind GPRS technology is to access Internet or Intranet through wireless devices.

The GPRS consists of layered protocols that provide quite a few control mechanisms, such as error correction and retransmission. Figure 2.2 shows the protocols between MS and SGSN. A brief introduction of some protocols is given below:

- Internet Protocol (IP) provides a best-effort way to transport datagrams from source to destination

- Transmission Control Protocol (TCP) is designed specifically to provide a reliable end-to-end connection over an unreliable internetwork. Congestion control is implemented in TCP to effectively avoid congestion caused by heavy load.

- Subnetwork Dependent Convergence Protocol (SNDCP) maps the network level packet data units onto the underlying logical link control layer. It also provides optional compression functionality of both the TCP/IP header and the data content.

- Logical Link Control (LLC) provides a reliable ciphered link between SGSN and MS.
- Radio Link Control (RLC) provides reliable transmission of data according to optional Automatic Repeat-reQuest (ARQ) functionality.
- Medium Access Control (MAC) controls MS access to the air interface and provides scheduling for associated signaling.
- Base Station System GPRS Protocol (BSSGP) manages the routing and Qos information for the BSS
- GPRS Tunneling Protocol (GTP) tunnels the protocol data units through the IP backbone

All these protocols cooperate together to avoid traffic congestion and guarantee a rather reliable services to the MS.

In brief, network efficiency in GPRS is improved because channels can be shared simultaneously by both voice and data packets. When a specific user does not transmit data on the channel, the share that he possesses then can be used by any other customers, which promises that the channels can always be utilized at a high efficiency.

2.2 EGPRS

Generally, EDGE was developed and standardized in 1999 to further increase the data speed in the existing cellular network. When applying into the GPRS cellular network, the new combination is called Enhanced GPRS (EGPRS). The kernel of the EDGE technique is the application of the 8-PSK modulation scheme which provides a higher data transmission rate per radio time slot than that is achieved with Gaussian Minimum-Shift Keying (GMSK) modulation in GSM. An 8-PSK signal can carry three bits per symbol over the radio path rather than one in GMSK case. In EGPRS, even around 60 kbps per time slot can be reached in ideal environment. It also features nine modulation and coding schemes (MCS), by which network can instantaneously switch over these MCSs in response to a change of radio interface parameters. When the EGPRS is on operation, some time slots are reserved for data traffic. The number of such slots can be adjusted dynamically depending on the network situation and data traffic demand. Thus, it is possible to always choose a proper MCS or number of time slots to reach the best throughput when the radio parameters vary. The benefit of this flexibility is undoubtedly immense, when taking account of the highly fluctuating radio environment where MSs tend to roam frequently and may suffer from different signal fading influenced by the MS movements or geographical environment.

The deployment of EGPRS needs modifications and updates in the BTS, BSC and MS as well in both hardware and software. However, the entire change is rather small when considering the whole infrastructure. Basically, EGPRS can achieve a rather high data rate so that quite a few 3G services thus can also be carried out in such a cellular network.

2.3 UMTS

Universal Mobile Telecommunications Service (UMTS) network is the European version of third generation (3G) mobile communication system. It is a next-generation cellular technology standard proposed by international standards organizations that include the Third Generation Partnership Project (3GPP) and International Telecommunication Union (ITU). The radio access technology that UMTS uses is called Wideband Code Division Multiple Access (WCDMA), which increases the voice capacity and greatly enhances data rates. There are other proposals for the air access technologies such as CDMA 2000 and TD-SCDMA [7], which are not covered here.

To well understand the differences between the second generation and the third generation systems, we may look at the new requirements of the third generation systems which are listed below:

1. Bit rates up to 2 Mbps
2. Variable bit rate to offer bandwidth on demand
3. Multiplexing of service with different quality requirements on a single connection, for example, speech, video and packet data
4. Delay requirement from delay-sensitive real time traffic to flexible best-effort packet data
5. Quality requirements from 10 % frame error rate to 10^{-6} bit error rate
6. Co-existence of second and third generation systems and inter-system handovers for coverage enhancements and load balancing

	WCDMA	GSM
Carrier spacing	5 Mhz	200 kHz
Frequency reuse factor	1	1-18
Power control frequency	1500 Hz	2 Hz or lower
Quality control	Radio resource management algorithms	Networking(frequency) planning
Frequency diversity	5 Mhz bandwidth gives multi-path diversity with Rake receiver	Frequency hopping
Packet data	Load-based packet scheduling	Time slot based scheduling with GPRS
Downlink transmit diversity	Support for improving downlink capacity	Not supported by the standard, but can be applied

Table 2.1: GSM vs. WCDMA in air interface

7. Support of asymmetric uplink and downlink traffic, e.g. web browsing
8. High spectrum efficiency
9. Co-existence of FDD and TDD modes

Table 2.1 lists the main differences between WCDMA and GSM in terms of air interface, which presents the new requirements and new features of WCDMA technology.

UMTS uses the same fundamental architecture for voice and data services as GPRS/EDGE. UMTS is divided into two main sectors: the UMTS Terrestrial Radio Access Network (UTRAN) and the UMTS core network. The biggest difference between GPRS/EDGE networks and UMTS networks is the deployment of WCDMA in the radio access network [12].

2.4 WLAN

WLAN is a high-bit rate solution for local areas with rather limited mobility support, however it is still flexible in terms of the exemption of the troublesome planning and deployment of the wires, especially in the terrains where wire solution is inconvenient or even impossible. Within the coverage area, user can freely roam his WLAN enabled terminal or station and get access to high data speed. First of all, the dominant standard that it came up with is named 802.11. Actually, it has been developed to a series of standards by the IEEE 802 working group. The most popular 802.11 family members in the series are

- IEEE 802.11a 54 Mbps, 5 GHz standard
- IEEE 802.11b Enhancements to 802.11 to support 5.5 and 11 Mbps using 2.4 ISM (Industrial, Scientific and Medical) band allocation.
- IEEE 802.11g 54 Mbps 2.4 GHz standard (backwards compatible with 802.11b) in ISM band

In addition to these basic versions, there are quite a few standards available for the service enhancement, extension or even corrections to previous specifications.

WLAN consists of four basic components: Distribution System (DS), Access Points (AP), Wireless medium and Mobile Station (MS), which are depicted in Figure 2.3. A brief introduction is given below.

- Distribution System: DS is a backbone network that connects several access points or Basic Service Sets (BSSs). It could be either wireless or wired.
- Access Point: AP is basically a station in the BSS that is connected to the DS. The dominant function of AP is to perform the bridging to the backbone. The AP can be analogous to a base station in cellular network, however, the former one's coverage radius might be much smaller.
- Wireless Medium: It is the bridge that connects the MS and AP and several physical layer technologies are well defined in this part such as Frequency Hopping, Direct Sequence Spread Spectrum and Infra-Red.
- Mobile Station: The WLAN-enabled terminal such as laptop, handset, etc.

Generally, IEEE 802.11 network can have two topologies to provide different scales:

1. Basic Service Set (BSS): It is composed of a group of stations under the direct control of a single coordination function. All stations in a BSS can communicate with any other stations directly.
2. Extensive Service Set (ESS): Several BSSs are involved in this case and they are connected by a DS. ESS can provide larger network coverage,

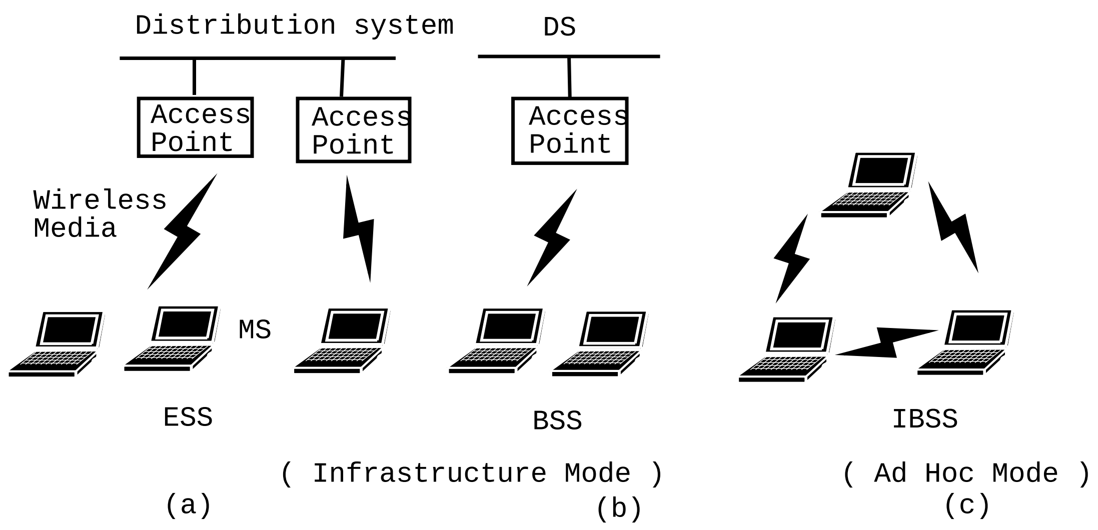


Figure 2.3: WLAN: working modes and main components (a) Infrastructure mode ESS (b) Infrastructure mode BSS (c) Ad hoc mode IBSS

which is also illustrated in Figure 2.3.

As to the flexibility, IEEE 802.11 standard is able to work in two different working modes, with or without AP. They are Infrastructure Mode and Ad Hoc Mode illustrated in Figure 2.3. In Infrastructure Mode, Access Point is required to bridge the wireless clients access to the wired network. In such mode, networks offer the advantage of scalability, centralized security management and improved accessibility. In Ad Hoc Mode, stations can even communicate directly with each other without a WLAN coverage. This case is also regarded as peer-to-peer or an independent BSS (IBSS) mode. Compared with the infrastructure mode, this mode is quicker to set up where wireless infrastructure does not exist. A simple example is several people sitting together in a room without WLAN coverage and having their WLAN enabled terminals connected to each other directly.

Nowadays, WLAN is widely deployed in airports, hotels, universities and even upscale coffee shops and is boosting a revolution in the access to the Internet. This application further strengthens the mobility of the laptop so that within the AP coverage, users can enjoy multiple data service without the limitation of the wire.

2.5 Load balancing

Load balancing (LB) can be applied in many fields, but mostly it plays an important role in the parallel and distributed computing systems. Assume we have a finite number of processors that handle the coming jobs in a system. In some circumstances, new arriving assignments may rush to a specific processor, which make it work quite heavily while the rest might be rather idle on the other hand. The total performance of the system is likely to degrade in this case and the utilization of the computing resource keeps relatively low. The motivation behind LB is to address such a problem: In essence, LB is designed to equally partition the work loads to several computing units such as processors in the manner to maximize the system utilization and performance.

In a cellular network, it is analogous to look BS in mobile network and AP in WLAN as the processors in computing system. Table 2.1 implies that load balancing has been standardized in UMTS and the packet data will be balanced according to the cell load [12]. And a case study of LB will be introduced in a WLAN system in the end of the section.

In order to achieve a high utilization of the resource, the load balancing should be 'fair' in some sense. One option, for example, is to distribute the total work to several optional computing units to minimize the work load difference between the busiest and idlest units. The practical implementation might be transferring the jobs from heavily loaded servers to rather idle serves so that jobs can be handled more efficiently and new assignments may enter the system without long waiting outside.

In general, we can broadly categorize load balancing algorithms into static or dynamic. For a static algorithm, the information for making the decision is collected before the assignment enters the system. And the state of the system will not affect the decision when the assignment is scheduled to a certain computing unit, in other words, the decision-making procedure is state independent. On the other hand, a dynamic allocation mechanism maps the assignment to the resources as they arrive in the system and states of the system have a great impact on its decision. In this way, decisions are made 'on the fly' and involve the working information in other computing units. In a word, dynamic load balancing is more complex to implement because it needs to make the decision dynamically, depending on the information in other computing units. On the other hand, static load balancing mechanism prescribes the decisions even before the assignments enter the system and totally independent to the states.

In terms of the implementation, load balancing can also be broadly categorized into two classes: centralized or distributed algorithm. In a centralized load-balancing algorithms, the global load information is collected into a scheduling processor such as a centralized access server which makes all the load balancing decisions for individual node according to such load messages. In a distributed algorithm, each node broadcasts its load information to the other nodes so that load balance tables are maintained locally and load of each node changes according to such table to achieve a better performance of the system. Normally, a centralized algorithm imposes less overheads but operates with rather low reliability because the collapse of the scheduling processor really means a disaster to the entire system. The information of the loads thus cannot be maintained properly or at least be messed up so that the realization of the load balancing is inaccurate or even impossible to approach. In contrast, a distributed algorithm is easier to implement despite of its rather low ability [21].

Some key components in load balancing are load metric, load measurement and load balancing operation. Load metric is the quantity to evaluate whether a system is balanced or not. Frequently used metrics are number of calls, blocking probability, packet loss, throughput, delay and so forth. It is also the goal for designing a load balancing scheme to optimize. Load measurement is the realization to calculate the load metric because always the load metric and other system information are quantifiable. There are options for either global or local servers being involved in the load balancing process. In the global situation, jobs migrate from the heavy loaded servers to any of the servers within the entire system if necessary while in the local case only servers in the scope of predefined area can accept the transferred jobs. Then based on load measurements, proper server is selected among all the candidates to redistribute the workload to realize the load balancing globally or locally according to the predefinition.

Case Study

Hector Velayos et al. [18] tried to avoid overloaded and underloaded cell to increase the total throughput performance in WLAN. We now take it as an example for studying load balancing in real WLAN scenario.

The model in [18] contains a few APs that are connected to the same Ethernet backbone. As claimed, AP in WLAN can be viewed as a server of computing units. Load Balancing Agent (LBA) is installed in each AP, by which load information of corresponding AP is broadcasted periodically to the common backbone. Based on these messages, an individual LBA figures out whether the load is balanced or not among the neighboring APs.

The load metric in this case is the throughput in each AP in bytes per second. Basically, the load situation of an AP can be broadly categorized into three segments: overloaded, balanced and underloaded, respectively. The throughput at APi is realized by the balance index β defined as,

$$\beta = \frac{(\sum B_i)^2}{(n \sum B_i^2)}, \tag{2.1}$$

where n is the number of neighboring APs over which the load is distributed and B_i is the throughput of APi. The balance index is 1 when all APs have the same throughput and tends to $1/n$ when the throughput is severely unbalanced. For classifying the load, an average load L is first defined as

$$L = \frac{\sum B_i}{n}, \tag{2.2}$$

along with a relatively small δ. The APs with load between L and $L + \delta$ are defined as balanced. Greater than $L + \delta$ or smaller than L is assumed to be overloaded or underloaded. The setting of δ is tricky: by adjusting δ, the range of balanced area is changing so that the number of unbalanced (overloaded and underloaded) varies correspondingly. In this sense, different amount of job load migrations are needed to eventually fix the system in a rather balanced condition. However, it is a tradeoff between the unbalance tolerance and job migrations workload.

The load balance implementation location in this case is global because all the APs are possibly involved when a workload migration occurs. The policy of load transfer deployed in [18] is a so-called *best candidate* scheme, which generates much less load transfers compared with the simple *random selection* policy. The best candidate scheme picks the stations whose throughput is the closest to the difference between the throughput of overloaded AP and the average load L.

The numerical experiment part verifies that this balancing scheme increases the total network throughput and decreases the cell delay considerably.

Chapter 3

Flow level traffic modeling

In this chapter, an overview of flow level modeling is explored from some observations of literatures. Some basic ideas of $M/M/1 - PS$ queue are then followed to introduce the service discipline that is associated with some simple but powerful results. Statistical bandwidth sharing is a relatively young field [16], but there are already a few papers in this area. This study is concerned about the behavior of the flow level traffic in data network, in which these flows are required to adjust their transmission rates to fully exploit of their bandwidth. With this statistical sharing, some simple and explicit performance expressions are derived. Under the perfect fair sharing assumption, the powerful results for processor sharing (PS) are accessible [14]. A general study is presented in [3], in which behaviors of Poisson arrival data traffic are examined in the flow level. In the cellular networks, Borst [4] showed that in certain cases, the flow-level performance may be evaluated by means of a multi-class PS model, where the total service rate varies with the total number of the user in the cellular network.

3.1 Data traffic modeling

Traffic in the packet-switched network is composed of data flows, which is normally embodied by individual transactions of packets stream corresponding to certain digital documents in the data network such as web pages, emails or data files. In [14, 3], the distribution of document size is modeled as one of the heavy-tailed distribution, the Pareto distribution:

$$Pr(size > x) \sim (\frac{k}{x})^{\alpha}, \tag{3.1}$$

where the exponent α satisfies $1 < \alpha \leq 2$ ($\alpha \leq 1$ results in a distribution with infinite mean). This distribution implies that the majority of flows are very small while only few ones are considerably large. The flow is mainly realized by one or several TCP connection, by which it is transmitted at any rate up to the limit imposed on the link and the system capacity. This flexibility in the transmitting rate explains why such flows are called 'elastic'.

Characteristics of IP traffic at packet level is notoriously complicated, the great variability in which can be manifested by the asymptotic self-similarity during the time scale smaller than the round trip time (RTT) [3]. It makes packet-level modeling quite complex to be analyzed. In contrast, flow modeling captures the behaviors of the dynamic users' demand. Thus, when examining the throughput of the system, it more naturally calls for the characterization at the flow level, where bandwidth sharing can be achieved by TCP congestion control algorithm. It is basically a resource allocation problem involving many flows, links and complicated global dynamics. In this algorithm, there are two windows to maintain the transmission work properly. One is the window that is approved by the receiver, the other is called congestion window defined by the sender. However, they both stand for the number of bytes the sender may transmit from either entity's point of view. Besides, there is a third parameters called threshold. The possible number of sending the contents is the minimum of the two windows. The basic idea is that each sender needs to maintain these two windows and ensures that the transmission rate is not beyond either window. And when congestion window is below the threshold, this window grows exponentially, otherwise, it increases linearly.

The congestion control algorithms in TCP are implemented to exploit the available capacity while adjusting the sending rate of competing transfers [17]. In this sense, it allocates the bandwidth between different flows, so the quality of service of elastic flow is rather relaxed. It has also been proved the performance of the flow is quite dependent on the number of flows in the link when a specific flow enters, which varies as new flows arrive and existing flows complete their jobs. This fluctuation in flow number can be viewed as a stochastic process. In [3], the arrival process is assumed to be Poisson process and quite a few simulations are conducted in an isolated link to gain insight into how TCP functions with all the flows and its resulting performance. Experiments illuminate that if there is no restriction on the receiver window and each involving flow has the same RTT, the behaviors of the entire flows are comparable to the simulation of the considered fair sharing system, where bandwidth tends to be equally shared among all the flows in progress, at least for the large flows. It is understandable when realizing that all the individual flows tend to experience different transmission rates which are quite determined by the extremely fluctuating flow number in the system upon their arrival. On the other hand, the receiving rate for big flows is more stable statistically.

Let $\pi(n)$ be the probability of n flows in the link at any instance and $R(s)$ be the expected response time of a flow of size s. Let $\rho = \lambda\delta/C$ denote link load, where λ is the Poisson arrival rate, δ is mean of job size and C is the link capacity. In the fair sharing, it is observed that the number of flows in progress is comparable to the number of customers in a $M/G/1 - PS$ queue, which will be introduced in the next section. Since the behaviors of the flows in fair sharing discipline can resemble the flow behaviors achieved by TCP, therefore, the performance of TCP connections can be modeled as an $M/G/1 - PS$ queue to some extent.

The fair sharing in bandwidth allows the powerful results from $M/G/1 - PS$ queue. However, unfair sharing might be more realistic in practice because various factors adversely affect the 'fairness'. Differentiated services mechanism is commonly implemented because geographical environments vary a lot in cellular network and flows tend to have different RTT. Sharing with bias always happens deliberately or not. Size-dependent scheme, for example, is observed to achieve better performance by scheduling flow transmissions according to their document size [14]. However, this mechanism preserves the performance for the majority of smaller flow and may systematically discriminate the larger documents. Borst [4] proposed a multi-class Processor-Sharing model when examining the flow level performance, which again reveals that user population is interrelated to the throughput characteristics and parameter setting.

To have an insight into the flow level traffic, one needs to understand bandwidth sharing among all the links of a network. Let the network be a set of links $l \in L$ and denote the capacity of each link l be C_l. A route r is then a subset of links in this network. Let flows on route r arrive at rate λ with mean size δ_r. Thus, the load of an individual link would be

$$\rho_l = \sum_{r \ni l} \frac{\lambda_r \delta_r}{C_l},$$

where $\rho_l < 1$. A notable conclusion is drawn in [3] that the bandwidth shared by small flows might vary widely because of the significant fluctuation because the number of flows in the system changes largely and frequently. It partially explains why it does not make more sense when the operator provides very precise promises in terms of the average throughput. However, for long flows, generally throughput is loosely constrained by the expected available capacity $C_l(1 - \rho_l)$ [3].

Based on such a capacity constraint, we assume that the flow traffic is perfectly fluid implying that the network bandwidth sharing is realized instantaneously as the number of the concurrent flows in the system changes.

In this way, flow control protocol realizes the ideal fairness and then the expected flow throughput performance is a simple function of the document size and capacity of each link [16]. In fact, an individual flow is likely to experience other flows coming and departing during its stay in the system. Thus, its obtained capacity from its TCP connection in a certain time scale is approximated to be C_l/n_l, where C_l is the capacity of the link it shares and n_l concurrent flows in it. Furthermore, the arrival process of flows is supposed to be a Poisson process and the document size can be arbitrary distributed.

This is really a model in an ideal situation because in the real world many factors affect the bandwidth sharing. In general, due to the highly unreliable wireless transmission, the throughput performance of cellular network suffers more complexity compared with the wired data network [17]. The capacity of cellular network may vary greatly because of the fluctuation of the transmit power and the increase of interference. Also, different RTTs for individual flows may make such balanced fairness even impossible to realize in the cellular network.

However, the ideal and simple $M/G/1 - PS$ queue still can model the wireless network to some extent and it provides some explicit and powerful formulae to describe the flow level performance.

3.2 Processor sharing queue

Many systems can be roughly generalized as a contention for limited resources by its users. Sometimes, conflicts arise when users request resources simultaneously especially when the workloads are considerably large compared with the resources. To avoid plunging system into a messy situation and keep it function properly as it is designed to be, a scheduling algorithm then is needed to allocate the resources to users properly. Basically, it is a set of rules to decide which customer is served and for how long. From now on, when talking about user we really specify the flows in the packet-switched cellular network to match the contents in the previous section though it is very straightforward to generalize the user definition to other applications. So, the period when a flow keeps in service is referred to as its quantum. A variety of service disciplines are developed to address the conflicts which adversely affect the system performance. Common disciplines are listed as following: FCFC (First Come, First Served), LCFS (Last Come, First Served), RR (Round-Robin with a fixed quantum), PS (Processor Sharing), SPT (Shortest Processing Time first) and SRPT (Shortest Remaining Process Time first).

When employing RR discipline, for example, flows take turns to get served for a pre-fixed time period, or quantum. If the quantum is much smaller than the total service time of a specific flow, the RR discipline evolves into the PS case, where flows are served simultaneously as soon as they enter the system. Denote the capacity of a server by C and the number of flows in the system by i. Using the PS discipline, the capacity of the server is equally shared by these i flows. The advantage of this discipline is that upon the arrival of a specific flow, it can be served by a certain portion of C without any waiting. The most appealing property about PS queue is that its average properties are insensitive to the distribution of the service demands of the customers, which makes it possible to apply it in more general situations. Partially because of its fairness and insensitivity, this PS discipline is widely adopted in modeling the bandwidth sharing system.

Assume that the arrival process of flows is Poisson with intensity λ and total service intensity is with parameter μ. At an instance, assume i flows in system, the overall departing probability per time unit is μ as long as $i > 0$.

The queue length distribution of the PS queue is the same as the ordinary M/M/1-FIFO queue [19].

$$E[N] = \frac{\rho}{1-\rho} \qquad E[T] = \frac{1/\mu}{1-\rho}, \tag{3.2}$$

where, $E[N]$ is the average number of customers in the queue and $E[T]$ is the average delay for this queue.

As said, these notable results are insensitive to the distribution the of job size, in other words, they still hold for the $M/G/1 - PS$ queue.

Chapter 4

Markov decision processes

In this chapter, we highlight the Markov Decision Process (MDP), which is the main optimization tool for refining the system. First, continuous-time Markov chains is brushed up. Then the fundamental ideas of MDP are introduced along with a case study. People make decisions every day. After that, it produces an immediate cost and generates possibilities to transfer to some other situations. As time evolves, quite a few consequences are collected to form the whole performance. In many ways, decisions are not isolated: a reasonable immediate result does not promise an excellent overall performance. The objective of Markov Decision Process (MDP) is to optimize the performance for a Markovian model in terms of some specific metrics.

4.1 Markov chains

In this section, continuous-time Markov chains (CTMC), a special type of Markov process, will be reviewed.

4.1.1 Continuous-time Markov chains

In CTMC, the states $s \in S$ are discrete and a state transition occurs at any random time and after that, system stays at that specific state for an exponentially distributed time.

A stochastic process is regarded as a Markov process if the next state (random variable) solely depends on the preceding state. In other words, the whole information that influences the future of the process has been fully summarized in the current state. Thus,

$$P[X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1]$$

$$= P[X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n], \tag{4.1}$$

which is referred to as the Markov property.

A CTMC can be described by a transition rate matrix, in which the basic elements are defined as

$$q_{i,j} = \lim_{\Delta t \to 0} \frac{P\{X_{t+\Delta t} = j \mid X_t = i\}}{\Delta t}, \quad i \neq j, \tag{4.2}$$

which is the probability per time unit that the system transits from state i to state j.

The diagonal entries in transition rate matrix are defined as $q_{i,i} = -q_i$, where q_i denote the total transition rate out of state i, $q_i = \sum_{j \neq i} q_{i,j}$. The time that system spends in some state is exponentially distributed and the mean is inversely proportional to q_i as $1/q_i$.

Due to the exponential distribution characteristics, we have

$$P\{X_{t+\Delta t} = i \mid X_t = i\} = 1 - q_i \Delta t + o(\Delta t), \tag{4.3}$$

where $o(\Delta t)$ is negligible compared with $\Delta t \to 0$. Hence, $q_i \Delta t + o(\Delta t)$ corresponds to the probability that the system transfers to another state in time unit Δt. Also, we write the probability from state i to certain state j,

$$P\{X_{t+\Delta t} = j \mid X_t = i\} = q_{i,j} \Delta t + o(\Delta t), \quad j \neq i. \tag{4.4}$$

So far, a basic idea of *embedded discrete time Markov Chain* is introduced, in which concentration is more at the transition instances. We ignore the exact time for an individual stay in some state and more focus on the probability by which one state transfers to another. The transition probability of such embedded Markov chain is given as

$$p_{i,j} = \lim_{\Delta t \to 0} P\{X_{t+\Delta t} = j \mid X_{t+\Delta t} \neq i, X_t = i\} \tag{4.5}$$

Combined with conditional probability, (4.3) and (4.4), rewrite $p_{i,j}$ as

$$\begin{aligned} p_{i,j} &= \lim_{\Delta t \to 0} \frac{P\{X_{t+\Delta t} = j, X_{t+\Delta t} \neq i \mid X_t = i\}}{P\{X_{t+\Delta t} \neq i \mid X_t = i\}} \\ &= \begin{cases} \frac{q_{i,j}}{q_i} & i \neq j \\ 0 & i = j \end{cases} \end{aligned} \tag{4.6}$$

We also define the state probability in balance

$$\pi_j = \lim_{t \to \infty} P\{X_t = j \mid X_0 = i\},\ j \in S$$

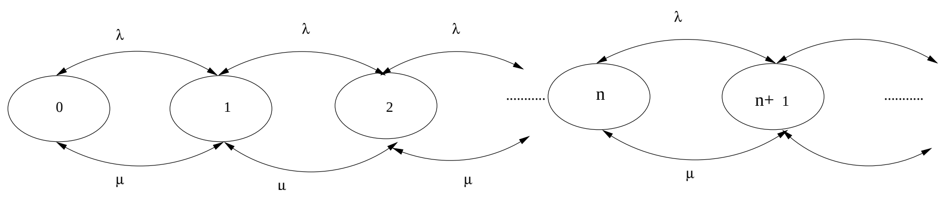


Figure 4.1: $M/M/1$, a typical birth-death process

This probability of process in state j is independent of state i because, as time passes by, the information of the initial state will be "washed out". In the stable states, the transition frequency from state i to state j equals the one from state j to state i, where $i \neq j$. Thus,

$$\sum_{i \neq j} q_{j,i}\pi_j \quad = \quad \sum_{i \neq j} \pi_i q_{i,j}, \tag{4.7}$$

with the normalization condition

$$\sum_j \pi_j = 1.$$

Combined with $\pi \cdot Q = 0$, a solution of the stationary probabilities is derived. The results are easier for computer processing especially when state set S is large

$$\pi = e \cdot (Q + E)^{-1}, \tag{4.8}$$

where E is an $n \times n$ index matrix with elements 1 and e is the vector with elements 1.

4.1.2 Birth-death process

Birth-death process is a special case of Markov chains, in which a state transition only occurs between neighboring states. Its name comes from the fact that such process is appropriate for modeling the changes in the size of population in Figure 4.1. In state i, the population size is i. Moreover, a transition from state i to state $i - 1$ stands for a "death" whereas a transition from state i to state $i + 1$ signifies a "birth" in the population.

Case study

A typical example is the $M/M/1$ queue depicted in Figure 4.1 in which customers arrive according to a Poisson process with rate λ and the service times are independent and identically distributed exponential random variables with mean $1/\mu$. Let $p_{i,i-1}$ and $p_{i,i-1}$ denote the transition probabilities from current state to neighboring states, respectively ($i \geq 1$).

$$p_{i,i-1} = \frac{\mu}{\lambda + \mu} \qquad p_{i,i+1} = \frac{\lambda}{\lambda + \mu},$$

which corresponds to the results in (4.6).

4.1.3 Quasi-birth-death process

A quasi-birth-death (QBD) process has two dimensions for possible transitions. In Figure 4.2, the y-axis stands for the different levels and x-axis for the various phases inside particular level. When the process is in level i_1 and phase i_2, we define that it is in state (i_1, i_2). The transition between states can only occur within neighboring levels by the definition of the QBD process (transitions between level 0, 1 and 2 in Figure 4.2).

A simple model, for example, is studied in [1], in which packets arrive in the system according to a Poisson process with intensity λ and capacities of two servers are μ_1 and μ_2, respectively. Let i_1 denote the number of packets in server 1 and i_2 for server 2, correspondingly. If we consider the granularity of the packets, i_1 and i_2 thus only increment by rate 1. If we let x-axis represent i_1 and y-axis for i_2, it results in a QBD. In this case, even in the same level of the QBD, the transition can only occur in the neighbouring state (transitions between level $i-1$, i and $i+1$ in Figure 4.2).

In brief, in this specific case it has a more stringent requirement, where state (i_1, i_2) can only transfer to the next state $(\tilde{i}_1, \tilde{i}_2)$, where $\mid \tilde{i}_k - i_k \mid \leq 1, k = 1.2$.

4.2 Markov decision process

This section explores the fundamental ideas of Markov Decision Process (MDP), which is widely used in the inventory management, hardware maintenance, communication models, etc. , as an effective systematic optimization tool. First, a few key components in MDP are introduced briefly. Secondly, the important concept called relative cost will be explained with a case study. The last section features an important tool: policy iteration to optimize a MDP. Practical implementation will also be examined with a case study. In the remainder of the section, the main focus is on the semi-Markov decision model, where the consecutive decision instances are not identical but random.

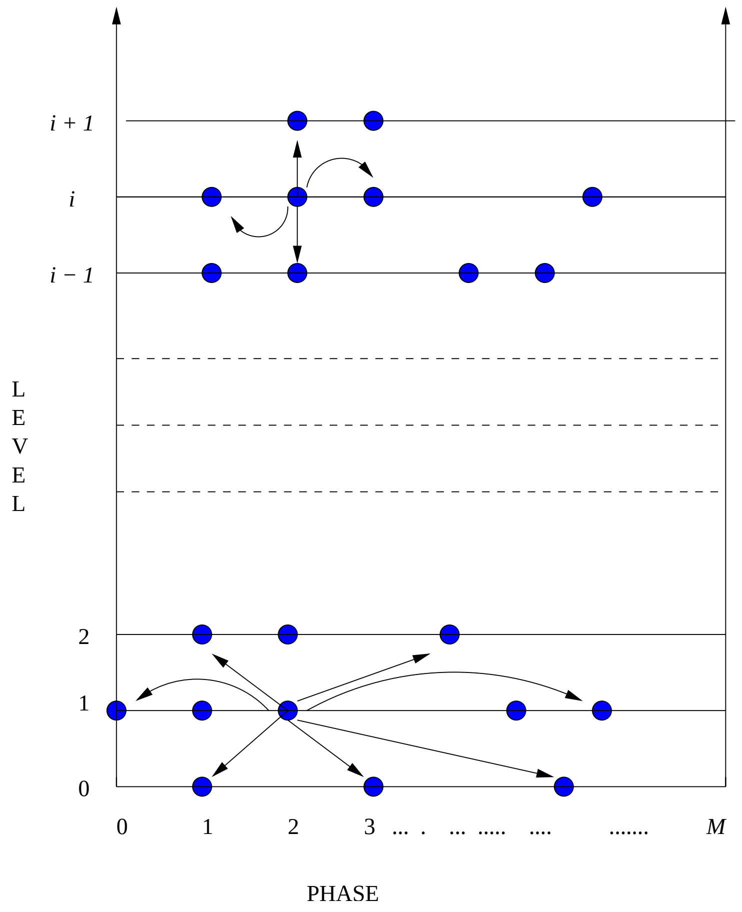


Figure 4.2: A quasi-birth-death process

4.2.1 Components of MDP

A symbolical model of decision making is given in Figure 4.3. First we go through a decision-making scenario, which features some important components in MDP to be explained later. Suppose we observe a process at time epoch $t = t_0, t_1, t_2 \cdots$, at which the process is in state $i \in I$. These states are discrete and can be enumerated by a nonnegative integers $i = 0, 1, 2, \ldots$. When process is in a specific state, multiple actions $a \in A$ are available and one of them must be chosen. Note that the actions for each individual state are possibly different. However, in the scope of this thesis, we suppose the available actions $a \in A$ for each state are the same.

In state i, if the process chooses an action a, two major consequences occur:

1. an immediate cost $C_i(a)$ is generated as the product of the cost rate $R_i(a)$ and the average accumulating time $1/q_i(a)$

2. process evolves to a new state j according to a transition probability $p_{i,j}(a)$.

Basically, $R_i(a)$, $1/q_i$ and $p_{i,j}(a)$ are the functions of the current state i and the chosen action a. In practice, we assume that before taking any actions, the process has the necessary information to select a specific $a \in A$ including the immediate cost $C_i(a)$ and $p_{i,j}(a)$. In order to choose actions in all states for the process, a *policy* α is needed. In this way, a policy is a rule to select the action in each state. In the scope of the thesis, the focus is on an important subset of policies called *stationary* policies, where a specific action is determined when a process is in a specific state i. A notable fact is that if a stationary policy α is implemented, the state sequence X_t forms a Markov chain with the transition probabilities $p_{i,j}(\alpha_i)$ that is determined by such a policy. We assume the process is time-homogeneous meaning that as long as the process selects the action a in state i, $C_i(a)$ and $p_{i,j}(a)$ keep unchanged even if the decision times vary.

In brief, several components have been introduced in action taking scenario:

1. policy for choosing actions $\qquad \alpha$

2. a set of system states $\qquad i \in I$

3. a set of available actions $\qquad a \in A$

4. a set of immediate cost rate $R_i(a)$ and average accumulating time $1/q_i$ dependent on the state and the action

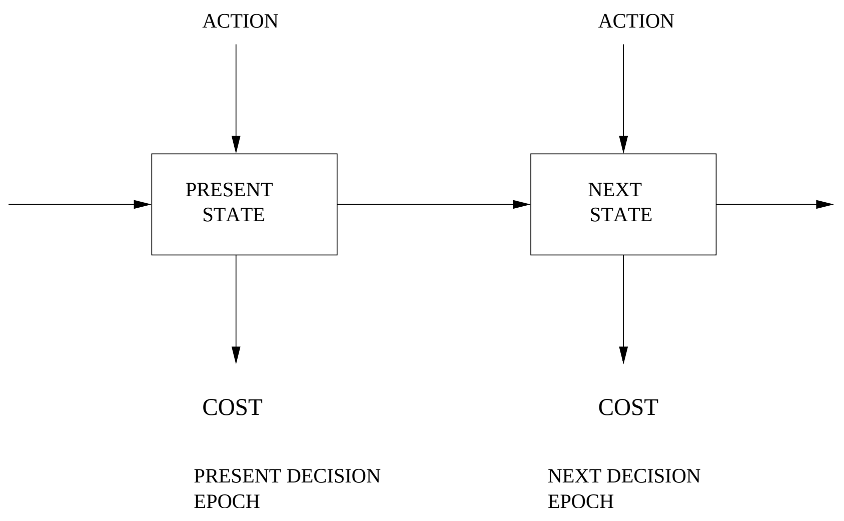


Figure 4.3: Diagram of a Markov decision problem

5. a set of transition probabilities dependent on state and action $p_{i,j}(a)$

As mentioned, once a stationary policy α is settled, an action in each state i is fixed correspondingly based on the given policy. Thus, policy α can be regarded as a mapping from the state set I to the action set A. Let α_i denote the action taken in state i under policy α.

As time evolves, one can collect a series of costs from each occupied states. The problem to minimize the overall cost in this context can be regarded as a Markov decision problem.

4.2.2 Howard's equation

In this section, *relative cost* will be introduced along with a case study. The transition rates $q_{i,j} = p_{i,j}q_i$ of a Markov decision problem can be fixed if a stationary policy α is applied. Afterwards, π_i can be easily derived from $q_{i,j}$ with the fact that $\sum_i \pi_i = 1$ in a similar way introduced in the section 4.1 .

An average cost rate of a given policy α can be viewed as the expectation of the immediate cost rate like $\bar{R}(\alpha) = \sum_i \pi_i(\alpha)R_i(\alpha_i)$, where $\pi_i(\alpha)$ is the stationary probability when system occupies state i and $R_i(\alpha_i)$ is the immediate cost rate when process takes action α_i in a specific state under policy α.

We may first introduce relative value and Howard's equation, which facilitates the later discussion. Denote $V_n(i, \alpha)$ as the expected cumulative cost over n steps meaning the sum of the consequential costs over n steps when the process starts from state i and always takes decisions by policy α. Figure

4.4 illustrates that the cost increments tend to converge to the average cost as decision epochs evolves in time [19]. In fact, under a given α, starting from different state i usually generates the same $V_n(i, \alpha)$ when $n \to \infty$. However, in order to evaluate the difference among these different consequences caused by various starting states, a comparison is made between $V_n(i, \alpha)$ and the average culmulative cost. This is the motivation of the relative value v_i and it reports the evaluation among different starting points under a given α:

$$v_i(\alpha) = \lim_{n\to\infty}(V_n(i, \alpha) - \tau_n(i, \alpha) \cdot \bar{R}(\alpha)),$$

where $\tau_n(i, \alpha)$ is the culmulative time from state i under policy α.

Generally, relative value $v_i(\alpha)$ of state i tells us under policy α how much greater cost is expected if the process begins from state i rather than from equilibrium.

Numerically, it might be difficult to realize $n \to \infty$. Howard's equation describes the relative cost in a slightly different way. The five key components listed in the previous section are included in such equation and their relevance is revealed somehow.

$$v_i(\alpha) = \frac{1}{q_i} \cdot R_i(\alpha_i) - \frac{1}{q_i} \cdot \bar{R}(\alpha) + \sum_{j \neq i} p_{i,j}(\alpha_i)v_j(\alpha) \qquad \forall i \in I, \tag{4.9}$$

where $p_{i,j} = q_{i,j}/q_i$. The term $\frac{1}{q_i} \cdot R_i(a) - \frac{1}{q_i} \cdot \bar{R}(\alpha)$ explicitly explains under policy α compared with the average cost, how much greater cost is consumed if process transfers to state i when we only take account of the immediate cost of state i. The latter term $\sum_{j\neq i} p_{i,j}(a)v_j(\alpha)$ stands for the weighted sum of the relative costs for each possibly states j as the next state for the process.

This notable equation paves the way for optimizing a process numerically because actions in each state thus can be intentionally chosen among all the options by examining the expected cost for each possible states.

Case study

Before discussing the powerful optimization tool: policy iteration, we first look into a case study of a typical MDP.

Let's first study the $M/M/1$ queue depicted in Figure 4.1. The parameters are the same as those in the case study in section 4.1. First, we clarify the key components to establish the Howard's equation. Let i, a nonnegative integer, denote the number of customers in this $M/M/1$ queue. As to the immediate cost $C_i(a)$, we refer to the cumulative time in the queue. Intuitively,

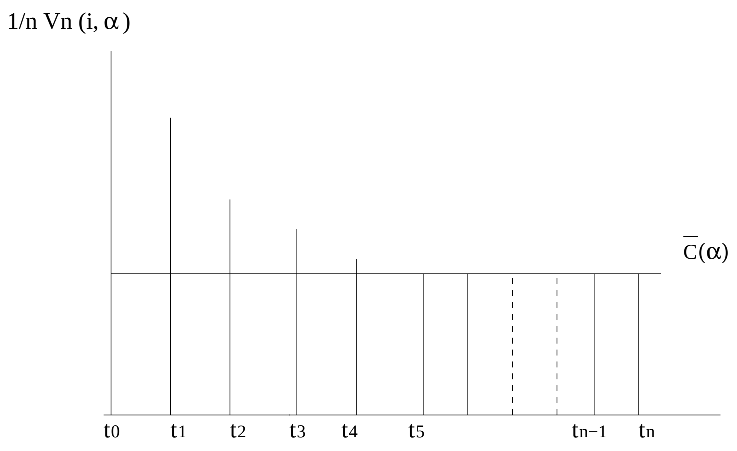


Figure 4.4: Terms in cumulative cost compared with the average cost

the larger user number in the queue, the higher rate the time is accumulated. So, basically the immediate cost rate is the number of customers in the queue. Another factor that influence the cost is the accumulating time. It means the average time, during which the rate is collected in a specific state. The average accumulating time is $1/(\lambda+\mu)$ except in state 0 because in state 0 the cost only accumulates after the first arrival, which results in an average collecting time to be $1/\lambda$.

From the knowledge about the $M/M/1$ queue, when in equilibrium, the mean queue length or the cost rate is

$$g = E[N] = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda},$$

where $\rho = \lambda/\mu$ is the load of the system.

When the process is in a given state i, there are always two options for actions, which is simply transferring to either new state $i+1$ or $i-1$ in this situation. As explained in the case study in section 4.1, the probabilities for such transitions are

$$p_{i,i-1} = \frac{\mu}{\lambda+\mu} \qquad p_{i,i+1} = \frac{\lambda}{\lambda+\mu},$$

for $i \geq 1$. In state 0, $p_{0,1} = 1$.

Thus, Howard's equation for state 0 is

$$v_0 = -\frac{g}{\lambda} + 1 \cdot v_1. \tag{4.10}$$

Similarly, for state i where $i > 0$, Howard's equations are

$$v_i = \frac{i}{\lambda+\mu} - \frac{g}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu}v_{i+1} + \frac{\mu}{\lambda+\mu}v_{i-1} \quad \forall i > 0. \tag{4.11}$$

Now, a follow-up study is carried out to derive the relative cost for an additional user in an $M/M/1$ queue. In (4.11), multiplying by $\lambda+\mu$ in both sides of the equal sign results in

$$i - \frac{\rho}{1-\rho} + \lambda(v_{i+1} - v_i) + \mu(v_{i-1} - v_i) = 0.$$

Let $u_n = \mu v_n$ and rewrite the above equation as

$$i - \frac{\rho}{1-\rho} + \rho(u_{i+1} - u_i) + (u_{i-1} - u_i) = 0$$

$$\rho(u_{i+1} - u_i - \frac{i+1}{1-\rho}) = u_i - u_{i-1} - \frac{i}{1-\rho}.$$

Thus,

$$u_{i+1} - u_i = \frac{i+1}{1-\rho}, \tag{4.12}$$

which implies that

$$v_{i+1} - v_i = \frac{i+1}{\mu-\lambda} \quad \forall i > 0. \tag{4.13}$$

Furthermore, for state 0, (4.10) implies

$$v_1 - v_0 = \frac{\lambda}{\mu-\lambda} \cdot \frac{1}{\lambda},$$

which also satisfies (4.13).

Finally, we come to

$$v_{i+1} - v_i = \frac{i+1}{\mu-\lambda} \quad \forall i \geq 0, \tag{4.14}$$

the marginal cost of an additional customer. By definition of relative cost, $v_i - v_j$ tells how much more it costs if the process starts from state i rather than j. Accordingly, marginal cost informs us the cost resulting from the arrival of an additional customer, which is manifested by the difference between v_{i+1} and v_i.

4.2.3 Policy iteration

A few tools are proposed to optimize the process performance according to the MDP, they are:

- Policy iteration,
- Value iteration,
- Linear programming.

In the remainder of the section, after a brief introduction of policy iteration, we examine how to implement it in a case study.

The key element of the policy iteration is a basic policy α, on which refinement is carried out through iteration. Howard's equations are established in each state that the studied process can experience. By setting the relative cost for a reference state be zero, we can solve the relative cost of each state $v_i(\alpha)$ and the average cost rate $\bar{R}(\alpha)$ under the given policy α.

As claimed in the previous section, when the process is in state i, various actions a can be taken and probably cause distinct immediate costs $C_i(a)$ and probabilities $p_{i,j}(a)$ to next possible states. The basic idea of policy iteration is to test any possible action a one by one and choose the action with the least expected cost. During this process, we employ the average cost rate $\bar{R}(\alpha)$ and the relative costs for next possible states $v_j(\alpha)$ directly from the previous policy. The expression of such expected cost is given below. Note that all the relative costs and the average cost in evaluating the expected cost are calculated under the previous policy α. Such 'best action' test shall be implemented in each state i, thus an iterated policy $\alpha^{'}$ is composed by all these newly selected actions with the least expected costs in each state.

In brief, policy iteration consists of the following three steps:

1. Choose a basic policy α as a starting-point.
2. For the given policy α, solve a set of relative value equations $v_i(\alpha) = \frac{1}{q_i}R_i(a) - \frac{1}{q_i}\bar{R}(\alpha) + \sum_{j\neq i} p_{i,j}(a)v_j(\alpha)$ $i \in I$ along with $\bar{R}(\alpha)$ by setting the relative cost of a reference state i be zero.
3. For each state $i \in I$, find an action a among ones available in the state i to fulfill the least expected value while keeping the previous $\bar{R}(\alpha)$ and $v_i(\alpha)$ unchanged. In mathematical form,

$$\alpha_i^{'} = \arg\min_{a\in A}\{\frac{1}{q_i}R_i(a) - \frac{1}{q_i}\bar{R}(\alpha) + \sum_{j\neq i} p_{i,j}(a)v_j(\alpha)\}, \tag{4.15}$$

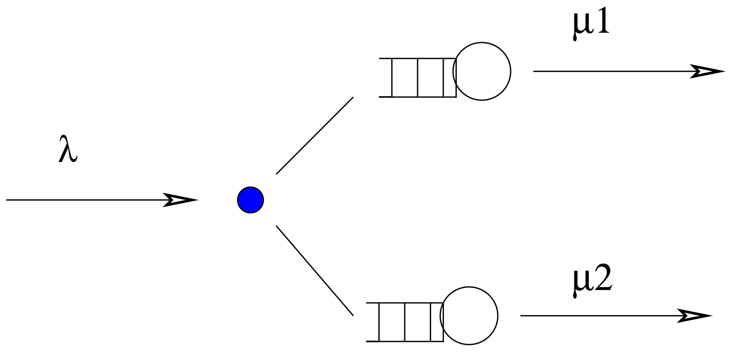


Figure 4.5: Basic routing problem

where $\alpha_i^{'}$ stands for the best action that is selected in state i based on the given policy α. In theory, $\alpha_i^{'}$ for the new policy $\alpha^{'}$ can never be worse than the α_i for the given basic policy α because the latter policy consists of all the 'best actions' that are selected intentionally based on the previous policy. By choosing the optimal action in each state we obtain a policy that is more likely to minimize the cost. Iteration goes on until the average revenue of the improved policy equals the one of the previous policy. And the final improved policy is regarded as *optimal policy* α^*.

Case study

Now, we apply the policy iteration in the problem depicted in Figure 4.5 as a case study. In fact, it is discussed in [1] with deliberation.

Now in Figure 4.5, packets arrive according to a Poisson process with intensity λ (packets/s) and they can either be routed to node 1 or node 2 whose capacities are denoted by μ_1 and μ_2 respectively, where $\mu_1 > \mu_2$. Let α_0 denote the static randomized policy, in which λ is routed to node 1 with probability p and to node 2 with probability $1-p$. This also implies that the whole system can be modeled by two separate $M/M/1$ queues with arrival intensities $\lambda_1 = p\lambda$, $\lambda_2 = (1-p)\lambda$, respectively. Now, one can denote the action set $A = \{1, 2\}$, meaning routing packet to node 1 or 2 and try to derive a stationary policy based on α_0.

Let i_n denote the user number in node n, $n = 1, 2$. If i_1 stands for 'phase' and i_2 for 'level' in QBD on page 27, the user number in the nodes can thus be regarded as a QBD in section 4.1.3.

Let $\hat{i}$ denote (i_1, i_2). When the process is in each state $\hat{i}$ and upon packets arriving, there are only two available actions that can be taken: route packets either to node 1 or 2. Now we are trying to implement the policy iteration to form a better policy $\alpha^{'}$ based on α_0. When process is in state $\hat{i}$ the immediate cost $C_{\hat{i}}(a)$ is the accumulating rate multiple the average staying time in this state. The average cost $\bar{C}_{\hat{i}}(\alpha)$ is derived as the weighted sum of the two separate average costs from the policy α_0.

In this specific case, $C_{\hat{i}}(a)$ actually does not even necessarily involve the action that probably varies from time to time. It solely depends on the state $\hat{i}$ itself, in other words, $C_{\hat{i}}(a)$ remains the same to both actions that belong to the same state. Thus, the first two terms in (4.15) are the same for either action and the only difference lies in the relative cost.

We now attempt both two actions $a \in \{1, 2\}$ in turn and find the very one with a less expected cost. Suppose under a static policy, the process is in state (i_1, i_2), the relative value of this state is $v(i_1, i_2) = v_1(i_1) + v_2(i - 2)$ because in static policy, we can model the two stations separately. Therefore, the relative costs for each action are $v_1(i_1 + 1) + v_2(i_2)$ for routing the flow to BS1 and $v_1(i_1) + v_2(i_2 + 1)$ for BS2. Because the immediate cost and the average cost are fixed for a specific state in (4.15), now the only element that affects the expected value is the relative value of a possible next state, namely, $(i_1 + 1, i_2)$ or $(i_1, i_2 + 1)$. A comparison of the expected values like $v_1(i_1+1)+v_2(i_2)-[v_1(i_1)+v_2(i_2+1)]$ evolves to $v_1(i_1+1)-v_1(i_1)-[v_2(i_2+1) - v_2(i_2)]$. Thus, a comparison between two marginal marginal costs is sufficient to finally decide the 'best action'.

According to (4.14), one can write the marginal cost for either node i with arriving intensity λ_i. A comparison is conducted between these two marginal costs as below:

$$t(i_1, i_2) = \frac{i_1 + 1}{\mu_1 - \lambda_1} - \frac{i_2 + 1}{\mu_2 - \lambda_2}, \tag{4.16}$$

where the former term is the marginal cost for node 1 and latter for node 2. If $t(i_1, i_2)$ is negative, it shows that the expected cost of routing a packet to node 1 in state (i_1, i_2) is smaller than routing to node 2. However, $t(i_1, i_2)$ being zero means in state (i_1, i_2), routing a packet to either node results in the same expected cost. This is actually how 'best action' $a^{'}$ is selected. When applying it in each state of the process, one can find a curve at which the expected costs are the same for both nodes. We thus construct the new policy $\alpha^{'}$, which can be completely characterized by such a curve

$$l(i_1) = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} i_1 + \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} - 1. \tag{4.17}$$

Thus, when the process is in the states that are above the curve, it is better to route packets to node 1. On the other hand, it costs less in the long run to route packets to node 2 when the process is in the states below the curve.

Chapter 5

Problem formulation

In this chapter, the research problem is formulated in detail, and it involves almost all the knowledge that has been covered in the previous chapters. Assumptions and notations that are used in the study will also be introduced generally. Some papers have already been addressed the related problems so far. In [1], an optimal routing problem in a connectionless packet-switched network is studied. The metric of optimization is the mean packet delay, which is also the mean length of queue according to Little's result.

5.1 Assumptions and notations

In the thesis, the focus is on the packet-switched cellular networks, especially the air access subsystem. A few hypotheses and notations are here claimed to facilitate the problem formulation. For simplicity, only two adjacent cells are considered in the scope of the thesis. Between these two neighboring base stations, there is an overlapping area in between, which is depicted in Figure 5.1.a. We can simply split the entire area into three parts. Let region 3 denote the overlapping area and 1 abd 2 stand for the left or the right part, respectively.

When Mobile Stations (MS) are in region$n$, $n = 1$, 2, the data flows will be directly routed to the BSn, where $n = 1$, 2. In this case, these flows are called *dedicated* flows. Sometimes MSs may move to region 3 thus the flows generated within this region could be routed either to BS1 or BS2, meaning that data services might be provided by either of the BSs. Consequently, the flows are regarded as *flexible* flows.

Assume the flow arrival processes to be Poisson processes of rate λ_i for the dedicated streams in BSn, $n = 1, 2$, and of rate ν for the flexible stream. Without loss of generality, we assume that $\lambda_1 \geq \lambda_2$. We also suppose the

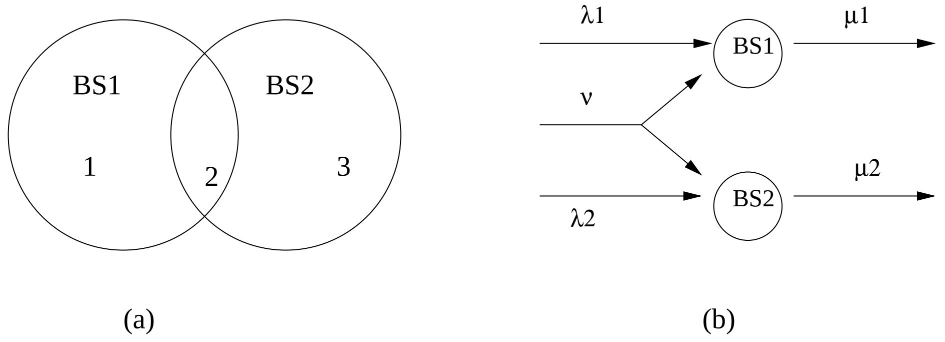


Figure 5.1: (a) two base stations with overlapping area (b) traffic allocation model

service times to be exponentially distributed with the parameter μ_i, where $i = 1, 2$. Here, we specify that service time refers to the time span to transfer the demanded bytes with the full BS capability. We also assume that each BS works in a processor sharing (PS) manner so that all the flows in process can share the BS capacity equally. We denote the number of flows in each base station by i_1 and i_2, which can be regarded as a stochastic processes. When there are i_1 flows in BS1 and i_2 flows in BS2, we define that the process is in state (i_2, i_2). Figure 5.1.b depicts the flow traffic that is described above.

Based on this information, we route flexible flows into appropriate BS to decrease the average flow delay in the system on the arrival of a new flexible flow. However, decisions can be made based on different levels of information of this system, which most likely results in different policies. Both static and dynamic policies are considered in the thesis. Static policies are determined based on time independent information of the system and decisions could be randomized according to the probability p. Dynamic policies depend on the time dependent knowledge and in this case, it should be stationary. So for a dynamic scheme, the action set for each state is totally the same: either route the flexible flow to BS1 or BS2, denoted by $a \in A = \{1, 2\}$.

The model in Figure 5.1.b is very similar as we discussed in Figure 4.5 and the main difference is that each node or BS now has a dedicated flow to deal with, which leads to the change in the arrival rate. However, it is still QBD as discussed in section 4.1.3 if we think i_1 as phase and i_2 as level on page 26 so that we can enumerate states by the flow number in either BS like state (i_1, i_2). Thus, most states have four states as the next possible destination to transition, all of which are bounded to the same level or phase. State (i_1, i_2) denoted by a circle in Figure 5.2, for example, can possibly transit to the states that are restricted to $(\tilde{i}_1, \tilde{i}_2)$, where $| \tilde{i}_n - i_n | \leq 1, n = 1, 2$. However, states (i_1, i_2), where $i_1 = 0$ or $i_2 = 0$, differ a little bit from the interior states

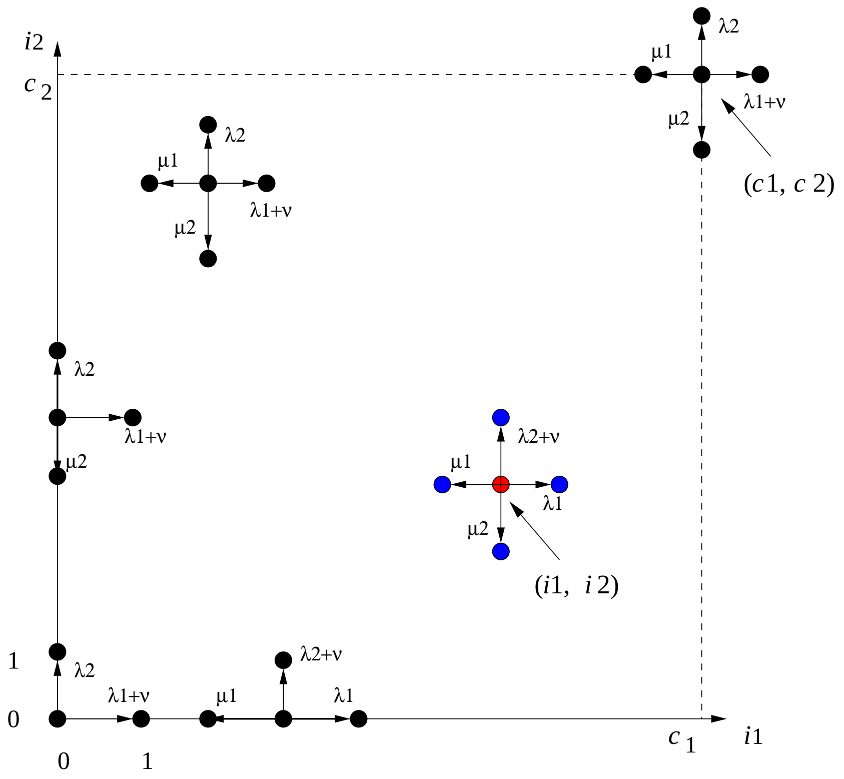


Figure 5.2: The flow diagram

in the transition degree situation. Inevitably, no departure could ever happen when there is no flow in a certain BS, which makes only three possible states to transfer for these kinds of states. Two typical examples are illustrated in Figure 5.2. One represents a state where $i_1 = 0$ and the other one is for a state in which $i_2 = 0$. Note that the actions in each state depicted in the Figure 5.2 is assumed to have been already selected.

In theory, since the bandwidth can be shared by all the flows statistically, a BS can hold almost an infinite number of flows if the network combats interference effectively and there is no lower bound of the transmission rate. Nevertheless, to study the model numerically we need to truncate the flow number to approximate the infinite situation. The chosen number taken into account for truncation will be discussed in chapter 8. Now, we just denote the number for truncation to be c_n for BSn, where $n = 1, 2$ so that the state space is given as $S = \{(i_1, i_2) \mid i_1 \leq c_1, i_2 \leq c_2\}$. As discussed in the previous chapter, as long as a stationary policy α is fixed, the sequence of states X_t forms a Markov chain with probabilities $p_{i,j}$, see section 4.2.1 on page 28. Because we denote the state in the problem with two arguments, we denote

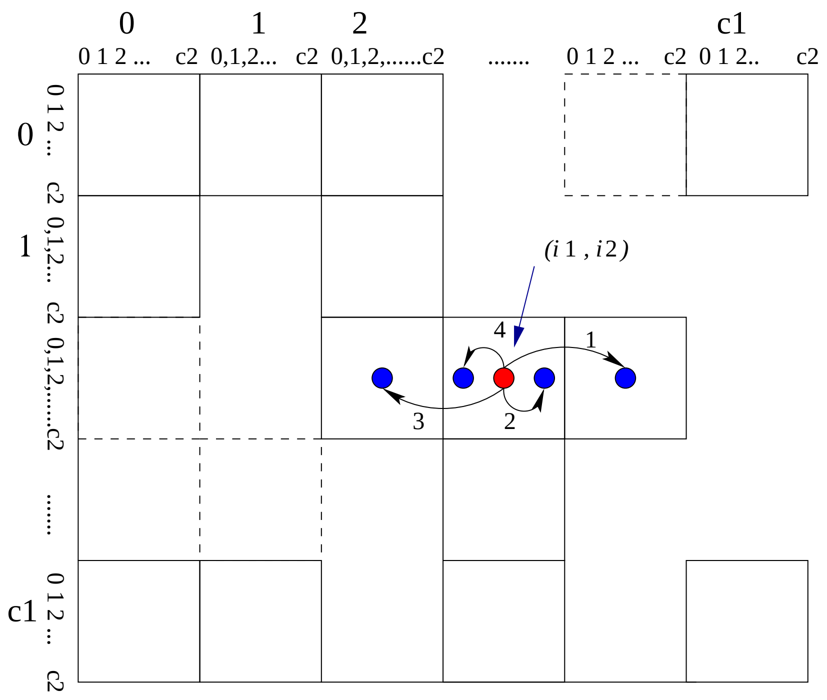


Figure 5.3: Building transition probability matrix

the state (i_1, i_2) by $\hat{i}$. Correspondingly, the transition probabilities $p_{i,j}$ evolve into $p_{\hat{i}}$. In Figure 5.3, We may look these state transitions in the form of a transition probability matrix.

The larger numbers from 0 to c_1 above each square represent the number of flows in BS1 and the small number from 0 to c_2 over an individual square for the number of flows in BS2. Thus, each state can be viewed as a circle in Figure 5.3. Take the same state (i_1, i_2) as in Figure 5.2, for example, its next possible states are restricted to states $(\tilde{i}_1, \tilde{i}_2)$, where $| \tilde{i}_n - i_n | \leq 1, n = 1, 2$. And the action in this state is still assumed to route the flexible flow to the BS2. We enumerate the four possible transitions with their probabilities to make it clearer,

1. Arrival to BS1 $(i_1, i_2) \to (i_1 + 1, i_2)$ with probability $\frac{\lambda_1}{\lambda_1+\lambda_2+\mu_1+\mu_2+\nu}$
2. Arrival to BS2 $(i_1, i_2) \to (i_1, i_2 + 1)$ with probability $\frac{\lambda_2+\nu}{\lambda_1+\lambda_2+\mu_1+\mu_2+\nu}$
3. Departure from BS1 $(i_1, i_2) \to (i_1 - 1, i_2)$ with probability $\frac{\mu_1}{\lambda_1+\lambda_2+\mu_1+\mu_2+\nu}$
4. Departure from BS2 $(i_1, i_2) \to (i_1, i_2 - 1)$ with probability $\frac{\mu_2}{\lambda_1+\lambda_2+\mu_1+\mu_2+\nu}$.

5.2 General method

In Figure 5.1.a, the question is what is the appropriate scheme to route the flexible flow whenever there is one entering the system, provided all the information is known as introduced before. In this context, scheme shares the same meaning with the policy, which is discussed in the previous chapter. The goal of the proposed scheme is to minimize the flow delay for the entire system, such that delay problem can be alleviated by an optimized load balancing scheme.

Hajek [6] has studied the general case $(\mu_1 \geq \mu_2)$ and also proved that in the two station context, a *switch-over* strategy is optimal. It is defined by a curve $l(i_1)$, where i_1 is the user number in station 1. Such a policy can minimize an average cost in the long run, such as blocking probability [13] and mean delay [1]. Actually, (4.17), a scheme from such two station problem in the previous chapter provides a convincing verification of the "switch-over curve" argument.

In [13], the same model is examined at the call level. A robust and sub-optimal scheme is proposed by ignoring the flexible flow generated in the overlapping area. We shall explore whether it still works effectively at the flow level so that it could be proposed as an efficient load balancing scheme to significantly mitigate the flow delay.

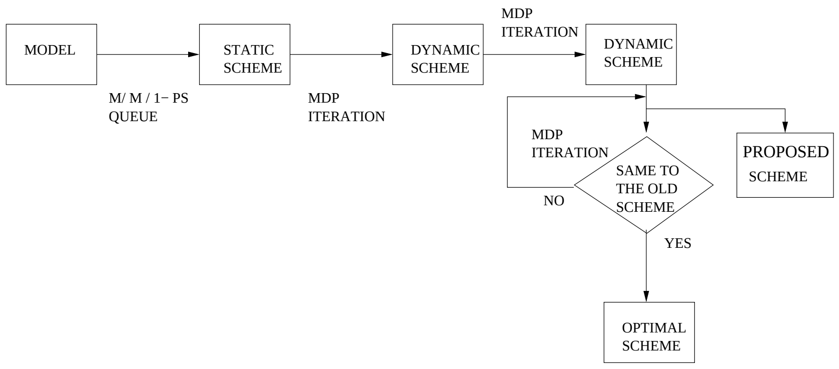


Figure 5.4: General method to address problem in the study model

Up to now, we have actually explored almost all the knowledge and information to address such a problem in the study model. We go through the general method depicted in Figure 5.4 to solve the problem, which is served as an outline of the coming chapters.

In the beginning of this chapter, we have already introduced the assumptions and notations to model a packet-switched network. Next, with the knowledge of the $M/M/1-PS$ queue, we shall derive an optimized randomized scheme which is denoted by α_0. It serves as a basis of the further scheme optimization to obtain a dynamic (stationary) scheme α_i as i iterations are implemented. Policy iteration continues until the iterated policy α_{i+1} is totally the same as the original policy α_i. Thus, we define the iterated policy α_{i+1} as the optimal scheme α^*. However, when we take account of the easy accessibility, it might be a little bit frustrating because probably a few steps are required before deriving it. In reality, when the number of the system states is considerably large, such workload problem exaggerates inevitably and the approach of α^* tends to be arduous. This is the motivation that we are trying to propose a suboptimal scheme which can be achieved within limited steps.

Chapter 6

Symmetric case study

This chapter studies the delay performance at flow level in packet-switched cellular networks described in chapter 2. The problem has already been formulated in the previous chapter. Furthermore, in this chapter, capacities of both base stations are assumed to be equal, denoted by μ ($\mu_1 = \mu_2$). The asymmetric case will be discussed in the next chapter. As described in the general method in Figure 5.4, we first study Optimized Random Routing (ORR), a static scheme as our starting point for the policy optimization. Then for first policy iteration (FPI) and its variation FPI*, we more or less mimic the procedures described in section 4.2.3 on page 34. For the later policies, we use policy iteration in MDP to achieve the smaller average delay of the model, whose general ideas are introduced in this chapter. However, the numerical results will be given and commented in chapter 8.

6.1 Static policy

In this section we mainly consider Optimized Random Routing (ORR) policy, a basic static policy.

Optimized Random Routing (ORR) is a basic static policy and its main idea is to divide the flexible flows into the two base stations as evenly as possible.

A random routing policy is here described by probabilities p, $1-p$ that are mentioned in Figure 5.1. Denote $p_1 = p$ to be the probability by which the flexible flow is routed to BS1 and similarly let $p_2 = 1-p$ denote the probability related to BS2.

Under a static policy, the system can be modeled as two separate $M/M/1-$

PS queues. Thus, the mean delay for an $M/M/1 - PS$ queue, is

$$E[D_i] = \frac{1}{\mu - \lambda_i - p_i\nu}. \tag{6.1}$$

The mean delay of the model is actually the weighted sum of these two separate queues' delay as

$$E[D] = \sum_{i=1}^{2} \frac{\lambda_i + p_i\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu - \lambda_i - p_i\nu} \tag{6.2}$$

In the remainder of this section, write $E[D]$ as a function of p, the probability to route the flexible flow to BS1.

$$f(p) = \frac{\lambda_1 + p\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu - \lambda_1 - p\nu} + \frac{\lambda_2 + (1-p)\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu - \lambda_2 - (1-p)\nu} \tag{6.3}$$

To solve the optimal $p^* \in [0,1]$ which minimizes (6.3) is a mathematical routine under the assumptions claimed before.

Write the first derivative of $f(p)$ as

$$\begin{aligned} \frac{df(p)}{dp} &= \frac{\nu(\mu - \lambda_1 - p\nu) + \nu(\lambda_1 + p\nu)}{(\lambda_1 + \lambda_2 + \nu)(\mu - \lambda_1 - p\nu)^2} \\ &- \frac{\nu(\mu - \lambda_2 - (1-p)\nu) + \nu(\lambda_2 + (1-p)\nu)}{(\lambda_1 + \lambda_2 + \nu)(\mu - \lambda_2 - (1-p)\nu)^2} \end{aligned} \tag{6.4}$$

Rewrite the equation above and try to find the p^* by setting (6.4) to zero,

$$\frac{\nu\mu}{\lambda_1 + \lambda_2 + \nu}\left[\frac{1}{(\mu - \lambda_1 - p\nu)^2} - \frac{1}{(\mu - \lambda_2 - (1-p)\nu)^2}\right] = 0 \tag{6.5}$$

The optimal probability p^* turns out to be

$$p^* = \begin{cases} \frac{1}{2} - \frac{\lambda_1 - \lambda_2}{2\nu}, & \nu \geq \lambda_1 - \lambda_2, \quad \text{LB} \\ 0, & \nu < \lambda_1 - \lambda_2, \quad \text{NLB} \end{cases} \tag{6.6}$$

which shares the same form as that in [13].

The condition $\nu \geq \lambda_1 - \lambda_2$ in (7.6) is defined as load balance (LB) condition. Literally, with LB, the system is always able to find a certain p to equalize the entire flow traffic in the model by allocating the flexible stream by probability p to BS1 and the rest to BS2. On the other hand, in not load balance (NLB) condition, system is not capable to balance the loads completely and from (7.6) all the flexible flows are routed to the BS2, the one with the less dedicated flow. Apparently, $\lambda_2 + \nu < \lambda_1$ in NLB, which means that the sum of the BS2's dedicated flow and the total flexible flow intensities is still less than the intensity of the BS1's dedicated flow. In this sense, it is understandable that BS1 only needs to deal with its own dedicated flow and the system forwards all the flexible flows to BS2. In the remainder of this chapter, both LB and NLB cases will be considered.

6.2 Dynamic policies

Now we try to design a scheme based on the time dependent information by policy iteration after introducing a fundamental and robust dynamic policy.

6.2.1 Join the shortest queue scheme

Join the shortest queue (JSQ) or shortest discipline has been well studied for a long time [20, 6]. Winston [20] considered the symmetric case, where servers have the same capacities ($\mu_1 = \mu_2$). Hajek [6] covered the general case ($\mu_1 \geq \mu_2$) and proved that a switch-over strategy is optimal for this two stations problems. In the context of a basic dynamic load balancing model, JSQ can be described as:

- When a user arrives, it is assigned to a server with the minimum queue length. If multiple servers achieve the minimum length, the user is assigned to one of them randomly with equal probability.

The switch-over curve in the context of model in Figure 5.1.a is defined by

$$l(i_1) = i_1, \tag{6.7}$$

where i_1 is the number of flows in BS1.

Hajek [6] discovered that an optimal policy can always be characterized by a switch-over curve. Therefore, the object for designing a load balancing scheme can embody such curves derived by policy iterations mathematically or numerically.

6.2.2 First policy iteration

In this section, we will derive the scheme after first policy iteration (FPI) based on ORR.

First, we begin with an $M/M/1 - PS$ queue with arrival rate λ and service rate μ as described in Figure 4.1.

As discussed in the case study in section 4.2.2, the marginal cost of the additional user is

$$v_{i+1}(\alpha) - v_i(\alpha) = \frac{i+1}{\mu - \lambda} \quad \forall i \geq 0.$$

Now, come back to the whole model given in Figure 5.1. It is very similar to the model that is discussed in the case study in section 4.2.3 on page 34. The only difference lies in the existence of the dedicated flow for each base station in the research model.

Based on the ORR policy, the entire system can be modeled by two separate $M/M/1$ queues, which is characterized by arrival intensities $\lambda_1 + p_1\nu$, $\lambda_2 + p_2\nu$ and capacity μ. To form a better policy, when process is in state i, it needs to choose an action that is associated with less expected cost according to (4.15):

$$\alpha_i^{'} = \arg\min_{a \in A} \{ \frac{1}{q_i} R_i(a) - \frac{1}{q_i} \bar{R}(\alpha) + \sum_{j \neq i} p_{i,j}(a) v_j(\alpha) \}.$$

The immediate cost $\frac{1}{q_i} R_i(a)$ here is the product of accumulating rate and the average staying time in state i, which is the same for a particular state no matter what action is finally taken. The average cost $\bar{C}_i(\alpha)$ can be derived as the weighted sum of the two separate average costs under ORR. So, the selecting criterion is again the relative costs of the next possible states.

Thus, it is very handy to directly utilize the result derived in (4.16). However, in the current case, the arrival intensities for either BS need to change a little bit according to the previous static policy, ORR.

$$t(i_1, i_2) = \frac{i_1 + 1}{\mu - \lambda_1 - p_1\nu} - \frac{i_2 + 1}{\mu - \lambda_2 - p_2\nu}. \tag{6.8}$$

In a given state $\hat{i} = (i_1, i_2)$, if $t(i_1, i_2) < 0$, it reflects that in the long run less cost can be achieved to route the flexible flow to BS1. Otherwise, to BS2 is the better alternative.

Similarly, by setting (6.8) to zero, a switch-over curve $l(i_1)$ is easy to derive:

$$l(i_1) = \frac{\mu - \lambda_2 - p_2\nu}{\mu - \lambda_1 - p_1\nu} i_1 + \frac{\mu - \lambda_2 - p_2\nu}{\mu - \lambda_1 - p_1\nu} - 1. \tag{6.9}$$

Because the FPI is based on the ORR, a more explicit expression of this switch-over curve is easily obtainable if the optimal probability p^* is applied in (6.9). Corresponding to the two cases for p^* , we also have two sets of switch-over curves in LB and NLB. In the LB case, $p^* = \frac{1}{2} - \frac{\lambda_1 - \lambda_2}{2\nu}$. Thus, the switch-over curve is:

$$\frac{\mu - \lambda_2 - p_2\nu}{\mu - \lambda 1 - p_1\nu} = 1.$$

We now rewrite the switch-over curve for LB as

$$l(i_1) = i_1, \tag{6.10}$$

which exactly turns out to be JSQ. Winston [20] has proved that it is the optimal policy for the symmetric case ($\lambda_1 = \lambda_2$).

In the NLB case, $p^* = 0$, therefore,

$$\frac{\mu - \lambda_2 - (1 - p^*)\nu}{\mu - \lambda_1 - p^*\nu} = \frac{\mu - \lambda_2 - \nu}{\mu - \lambda_1} > 1.$$

The switch-over curve in NLB case turns out to be

$$\frac{\mu - \lambda_2 - \nu}{\mu - \lambda_1} i_1 + \frac{\mu - \lambda_2 - \nu}{\mu - \lambda_1} - 1 \tag{6.11}$$

In conclusion,

$$l(i_1) = \begin{cases} i_1, & \nu \geq \lambda_1 - \lambda_2 \quad \text{LB} \\ \frac{\mu-\lambda_2-\nu}{\mu-\lambda_1} i_1 + \frac{\mu-\lambda_2-\nu}{\mu-\lambda_1} - 1 & \nu < \lambda_1 - \lambda_2 \quad \text{NLB} \end{cases} \tag{6.12}$$

which completely characterizes the FPI based on ORR.

6.2.3 FPI*, a variation of FPI

From [13], a study in the same model at the call level, a notable observation is that when totally ignoring the flexible flows in the model, the resulting FPI denoted by FPI* can much more closely resemble the optimal policy. Similar to such operation in [13], the section tries to derive the FPI* at the flow level by ignoring the flexible flows.

In FPI^* case, the only criteria to select an action is the variant marginal cost like

$$\frac{i_n + 1}{\mu - \lambda_n}$$

for BSn, where $n = 1, 2$. The flexible flows are completely ignored.

Thus, the comparison of two expected costs or two variant marginal costs in this context turns out to be

$$t(i_1, i_2) = \frac{i_1 + 1}{\mu - \lambda_1} - \frac{i_2 + 1}{\mu - \lambda_2} \tag{6.13}$$

Still by setting (6.13) to zero, the switch-over curve is given by:

$$l(i_1) = \frac{\mu - \lambda_2}{\mu - \lambda_1} i_1 + \frac{\mu - \lambda_2}{\mu - \lambda_1} - 1 \tag{6.14}$$

Only if the arriving rates of the two dedicated flows are the same ($\lambda_1 = \lambda_2$), the switch-over curve evolves into

$$l(i_1) = i_1.$$

In FPI*, we fully ignore the flexible stream. Another option is to admit the new generated flexible flows with a probability p into the system. Such, there are still quite a few variations of FPI which are most likely to have different delay performance. Before we look into them in Chapter 9, we first introduce two parameters to facilitate the later discussion. Let f be the fraction of the flexible flow to be taken into consideration and p be the probability by which the flexible flow is routed to BS1. Each different combination of f and p forms a unique FPI variation. The delay performances of all these FPI variations are reported in Chapter 8.

6.2.4 Further policy iterations

In the previous section, we derived the switch-over curves that can fully characterize FPI and FPI*. In this section, the focus is on the further iterations and the derivation of the optimal policy α^*. However the further policy iterations are based on the dynamic policy rather than static, the action taken in each state thus can not be randomized selecting by some probability any more. Consequently, the average delay is not so straightforward as a weighted sum of the two $M/M/1$ queues and the relative cost of a specific state $\hat{i}$ cannot hold like that in (4.14) since its separate style model even breaks. Therefore, we need to think the numbers of flows in both base stations as a whole when considering the cost rate of each state. Indeed, it is more practical to solve such a problem numerically. This section is more or less served as an introduction about practical methods used in deriving the further iterations. Results of numerical experiments are implemented and given in chapter 8.

The basic ideas of deriving an iterated policy are the same as introduced in the previous chapter. We start with an original policy α that prescribes the actions in each state like $a = \alpha_i$, $\forall i \in I$. Correspondingly, the state sequence is a Markov chain by transition probabilities $p_{\hat{i},\hat{j}}$. The iterated policy is composed by all the actions $a^{'} = \alpha^{'}_{i}$, $\forall i \in I$ re-selected according to the equation:

$$\alpha_i^{'} = \arg\min_{a \in A}\{\frac{1}{q_i}R_i(a) - \frac{1}{q_i}\bar{R}(\alpha) + \sum_{j \neq i} p_{i,j}(a)v_j(\alpha)\}.$$

For the convenience of the calculation, we often choose a reference state and set its relative cost zero. The number of unknown relative costs thus decreases by one likewise, however, the average cost $\bar{C}(\alpha)$ is also an unknown parameter that is solely dependent on the policy α. In principle, by building up Howard's equations in all the states, we can solve the relative values and average cost of the system under a given policy so as to select less costly action $a \in A$ in each state and construct an optimized policy $\alpha^{'}$.

When implementing Howard's equation numerically, we truncate the infinite number of flows into a specific finite number c_n, $n = 1, 2$ for feasible operation. Then, we split the states in Figure 5.2 into several parts according to their different transition degrees, reflecting the possible number of their next destinations, namely,

- $0 < i < c,\ 0 < j < c$
- $i = 0,\ 0 < j < c$
- $j = 0,\ 0 < i < c$
- $i = 0,\ j = 0$
- $i = c,\ 0 < j < c$
- $j = c,\ 0 < j < c$
- $i = c,\ j = c$
- $i = 0,\ j = c$
- $i = c,\ j = 0.$

When the process is in state $(0, 0)$, for example, the only possible moves are to state $(1, 0)$ and $(0, 1)$ in Figure 5.2 and no departure could ever happen in this state.

Howard's equation in each state is built up according to (4.9) under a given policy α. An interior state (i_1, i_2), for example, has four possible states for next transition. The average period until next event (departure or arrival) is $1/(\lambda_1 + \lambda_2 + \nu + 2\mu)$. However, in a dynamic policy, the cost rate is composed by the number of flows in both BSs as a whole. So, in each state

(i_1, i_2), it turns out to be $i_1 + i_2$. Now, for simplicity, denote the relative cost in state i_1, i_2 by $v(i_1, i_2)$. And assume that the new one still keeps the original policy α during the procedures of establishing Howard's equations and solving relative costs. For a certain state (i_1, i_2), we assume that the flexible flow is entirely routed to BS1. Therefore, Howard's equation in this state is given:

$$\begin{aligned} v(i,j) = {} & \frac{i+j}{\lambda_1+\lambda_2+\nu+2\mu} - \frac{g}{\lambda_1+\lambda_2+\nu+2\mu} \\ & + \frac{\lambda_1+\nu}{\lambda_1+\lambda_2+\nu+2\mu} v(i+1,j) \\ & + \frac{\lambda_2}{\lambda_1+\lambda_2+\nu+2\mu} v(i,j+1) \\ & + \frac{\mu}{\lambda_1+\lambda_2+\nu+2\mu} v(i-1,j) \\ & + \frac{\mu}{\lambda_1+\lambda_2+\nu+2\mu} v(i,j-1). \end{aligned} \tag{6.15}$$

The transition probabilities for transferring to state $v(i+1, j)$ or $v(i, j+1)$ change correspondingly if the flexible flow is routed to BS2.

Like in section 4.2.2, the average time until the next event in different parts (as suggested in the previous page) varies accordingly. State $(0, 0)$, for example, its mean period until next happening is only $1/(\lambda_1 + \lambda_2 + \nu)$. The average staying time also changes in the boundary states compared with those in the interior states.

As said, we truncate the number of flowsin either BS by c_n, $n = 1, 2$ to approximate the infinite number of flows in the system. However, some tricks are needed to deal with the adverse consequences from the truncation. When constructing Howard's equation for states in the upper or right bounder, in theory, all the relative costs of possible next states should be involved to build up Howard's equation even though some outer states are excluded in the consideration because of the truncation.

We assume that the relative costs increase linearly. In such a case, operation is able to carry on with the exemption of the states (i_1, i_2), where $i_n > c$, $n = 1, 2$. Take state (c, c), where $c = c_1 = c_2$ in Figure 5.2 for an example, we now write the modified Howard's equation for boundary states:

$$\begin{aligned} v(i,j) = {} & \frac{c+c}{\lambda_1+\lambda_2+\nu+2\mu} - \frac{g}{\lambda_1+\lambda_2+\nu+2\mu} \\ & + \frac{\lambda_1+\nu}{\lambda_1+\lambda_2+\nu+2\mu} \{v(c,c) + \frac{1}{c}[v(c,c) - v(c,0)]\} \end{aligned}$$

$$
\begin{aligned}
&+ \frac{\lambda_2}{\lambda_1 + \lambda_2 + \nu + 2\mu}\{v(c,c) + \frac{1}{c}[v(c,c) - v(0,c)]\} \\
&+ \frac{\mu}{\lambda_1 + \lambda_2 + \nu + 2\mu} v(i-1,j) \\
&+ \frac{\mu}{\lambda_1 + \lambda_2 + \nu + 2\mu} v(i,j-1). \qquad (6.16)
\end{aligned}
$$

With (6.15) and (6.16), Howard's equations are built up in each state based on the given policy α. For calculation convenience, state $(0,0)$ is regarded as the reference state whose relative cost is thus set to be zero. Thus, all the other relative costs $v(i_1, i_2)$ along with the average queue length can be solved from all Howard's equations afterwards.

Deriving the optimal policy needs to repeat the policy iteration for several time. However, the basic ideas of the iterations are the same as introduced in section 6.2.3. The immediate cost for a given state $C_{\hat{\imath}}(\alpha)$ keeps the same for either action $a \in \{1, 2\}$. Furthermore, the average cost rate is constant for a fixed policy α. Thus it is sufficient to compare the relative costs for the next states to derive the action $a^{'}$ for the iterated policy $\alpha^{'}$. The action which leads to the state with less expected cost is picked according to the policy iteration algorithm on page 34 and an iterated policy $\alpha^{'}$ is formed by repeating this action selection in every state.

This iteration continues until nothing changes, indicating the derivation of the optimal policy α^* eventually.

Chapter 7

Asymmetric case study

In the previous chapter, we assumed that the base stations have the same capacities. However, capacities of the base stations in one system are more likely to vary due to the different manufactures and settings. In this chapter, we discuss the delay performance when two BSs have different capacities. Without loss of generality, $\mu_1 > \mu_2$. First, Optimized Random Routing (ORR) scheme is studied which features the optimal probability p^* in the new context. Secondly, a basic dynamic load balancing scheme Least Ratio Routing (LRR) is introduced briefly and we deduce FPI and FPI* as in the symmetric case.

7.1 Static policy

In a randomized manner, the system can be modeled as two separate $M/M/1$ queues when considering the static allocation scheme. We are now trying to find out the optimal probability p^* to minimize the mean queue length. Let $p_1 = p$ denote the probability by which the flexible flow is routed to BS1 and $p_2 = 1 - p$ to BS2. The mean delays of the system is the weighted sum of the two separate $M/M/1$ queues' delay:

$$E[D] = \sum_{i=1}^{2} \frac{\lambda_i + p_i\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu_i - \lambda_i - p_i\nu}. \tag{7.1}$$

As a function of p, the mean delay reads as:

$$\begin{aligned} f(p) = & \frac{\lambda_1 + p\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu_1 - \lambda_1 - p\nu} \\ & + \frac{\lambda_2 + (1-p)\nu}{\lambda_1 + \lambda_2 + \nu} \cdot \frac{1}{\mu_2 - \lambda_2 - (1-p)\nu}. \end{aligned} \tag{7.2}$$

The first derivative of the equation above turns out to be,

$$\begin{aligned}\frac{df(p)}{dp} &= \frac{1}{\lambda_1+\lambda_2+\nu}\cdot\frac{\nu(\mu_1-\lambda_1-p\nu)+\nu(\lambda_1+p\nu)}{(\mu_1-\lambda_1-p\nu)^2}\\ &- \frac{1}{\lambda_1+\lambda_2+\nu}\cdot\frac{\nu(\mu_2-\lambda_2-(1-p)\nu)+\nu(\lambda_2+(1-p)\nu)}{(\mu_2-\lambda_2-(1-p)\nu)^2}\\ &= \frac{\nu}{\lambda_1+\lambda_2+\nu}[\frac{\mu_1}{(\mu_1-\lambda_1-p\nu)^2}-\frac{\mu_2}{(\mu_2-\lambda_2-(1-p)\nu)^2}]. \qquad (7.3)\end{aligned}$$

Set (7.3) to be zero to derive the p to minimize the mean delay, then

$$\frac{\sqrt{\mu_1}}{\mu_1-\lambda_1-p\nu}=\frac{\sqrt{\mu_2}}{\mu_2-\lambda_2-(1-p)\nu}. \qquad (7.4)$$

From the equation above, we calculate the optimal probability p^*,

$$p^*=\frac{\sqrt{\mu_2}}{\nu(\sqrt{\mu_2}+\sqrt{\mu_1})}(\mu_1-\lambda_1-\sqrt{\mu_1\mu_2}+\sqrt{\frac{\mu_1}{\mu_2}}(\lambda_2+\nu)). \qquad (7.5)$$

We bound $p^* \in$ [0,1]. In other words, if p^* is less than zero, we set p^* to be zero. On the other hand, if p^* is greater than one, we set p^* to be one. If we fix the capacities of both base stations ($\mu_1 \geq \mu_2$), then $p^* = 0$ if

$$\lambda_1 \geq \sqrt{\frac{\mu_1}{\mu_2}}\lambda_2+\sqrt{\mu_1}(\sqrt{\mu_1}-\sqrt{\mu_2})+\sqrt{\frac{\mu_1}{\mu_2}}\nu.$$

On the other hand, $p = 1$ if

$$\lambda_1 \leq \sqrt{\frac{\mu_1}{\mu_2}}\lambda_2+\sqrt{\mu_1}(\sqrt{\mu_1}-\sqrt{\mu_2})-\nu.$$

In brief, p^* is given as:

$$p^*=\begin{cases}1, & \lambda_1 \leq \sqrt{\frac{\mu_1}{\mu_2}}\lambda_2+\sqrt{\mu_1}(\sqrt{\mu_1}-\sqrt{\mu_2})-\nu\\ 0, & \lambda_1 \geq \sqrt{\frac{\mu_1}{\mu_2}}\lambda_2+\sqrt{\mu_1}(\sqrt{\mu_1}-\sqrt{\mu_2})+\sqrt{\frac{\mu_1}{\mu_2}}\nu\\ \frac{\sqrt{\mu_2}}{\nu(\sqrt{\mu_2}+\sqrt{\mu_1})}(\mu_1-\lambda_1-\sqrt{\mu_1\mu_2}+\sqrt{\frac{\mu_1}{\mu_2}}(\lambda_2+\nu)), & \text{otherwise.}\end{cases} \qquad (7.6)$$

7.2 Dynamic policy

7.2.1 Least ratio routing

The Least Ratio Routing (LRR) scheme, another basic dynamic routing model is also a variation of the Least Load Routing [2]. When an arriving flexible flow enters the system, it is assigned to the base station with the least *relative load*. It is defined for each BSn as the ratio of the flow number in the base station to the capacity of it, namely i_n/μ_n, $n = 1, 2$. The switch-over curve in this case is defined by

$$l(i_1) = \frac{\mu_2}{\mu_1} \cdot i_1.$$

In the symmetric case where $\mu_1 = \mu_2$, the switch-over curve of LRR evolves to $l(i_1) = i_1$ which is exactly the JSQ allocation policy.

7.2.2 First policy iteration

We have already obtained the optimal probability p^* of ORR in asymmetric case. Now based on this scheme, we are going to reveal the switch-over curve of first policy iteration (FPI). Refer to (6.9), we write

$$l(i_1) = \frac{\mu_2 - \lambda_2 - p_2\nu}{\mu_1 - \lambda_1 - p_1\nu} i_1 + \frac{\mu_2 - \lambda_2 - p_2\nu}{\mu_1 - \lambda_1 - p_1\nu} - 1, \tag{7.7}$$

where $p_1 = p^*, p_2 = 1 - p^*$.

Because p^* is in a rather complex form in (7.5), the term

$$\frac{\mu_2 - \lambda_2 - p_2\nu}{\mu_1 - \lambda_1 - p_1\nu}$$

is not as explicit as that in (6.12). However, from (7.7) we can speculate that FPI can still be characterized by a switch-over curve.

7.2.3 FPI*, a variation of FPI

As in the symmetric case, we neglect the flexible stream totally when comparing the marginal costs of the two base stations:

$$t(i_1, i_2) = \frac{i_1 + 1}{\mu_1 - \lambda_1} - \frac{i_2 + 1}{\mu_2 - \lambda_2}. \tag{7.8}$$

Denote the iterated policy to be FPI*. If we set the above expression to be zero, the final switch-over curve is given as

$$l(i_1) = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} i_1 + \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} - 1. \tag{7.9}$$

Chapter 8

Numerical experiments

Now, we report some numerical experiments to support the results derived in the previous two chapters. The system model considered here is the same as described in Figure 5.1. Generally, we have two sets of experiments. One is for the symmetric case ($\mu_1 = \mu_2$) and the other is for the asymmetric case in which the capacities are different ($\mu_1 > \mu_2$).

8.1 Performance of the symmetric case

In this section, we present the system performances in the symmetric case.

First, we discuss the appropriate number in truncation to approximate the infinite user number in the system. In Table 8.1 and 8.2, the experiments are implemented while using different number for truncation under the same system parameter assumptions. The results for using 60, 70 as truncation number are completely the same as using 50, which illuminates that 50 is sufficient to approximate the infinite flow number scenario because increasing the considered number for truncation does not render any change in the delay performance. Thus, in Figure 8.1 and 8.2, we take the flow number limitation as 50 to approximate the infinite flow number. From these plots (a)-(d) in these figures , we still see flaws remain in the upper boundary states, however,

	N_{ORR}	N_{FPI}	N_{FPI^*}	N^*
50	2.210	2.010	1.984	1.984
60	2.210	2.010	1.984	1.984
70	2.210	2.010	1.984	1.984

Table 8.1: Comparison of performances applying different truncation number with system parameters: $\mu = 20$, $\lambda_1 = 10$, $\lambda_2 = 6$ and $\nu = 5$.

	N_{ORR}	N_{FPI}	N_{FPI^*}	N^*
50	2.583	2.467	2.464	2.463
60	2.583	2.467	2.464	2.463
70	2.583	2.467	2.464	2.463

Table 8.2: Comparison of performances applying different truncation number with system parameters: $\mu = 25$, $\lambda_1 = 15$, $\lambda_2 = 8$ and $\nu = 5$

$(\lambda_1, \lambda_2, \nu)$	μ	p^*	N_{ORR}	N_{FPI}	N_{FPI^*}	N^*
(5,5,5)	15	0.5	2	1.689	1.689	1.689
(10,8,5)	15	0.3	6.571	5.220	5.178	5.172
(10,5,4)	20	0	1.818	1.719	1.719	1.718
(15,10,4)	20	0	5.333	5.048	5.025	5.017

Table 8.3: Performance of FPI and FPI* compared with the optimal policy

it does not adversely affect the delay performance results.

8.1.1 FPI and further iterations

Table 8.3 shows the performance of FPI and FPI* in some instances, where N stands for the average queue length in the model. It is proportional to the average delay in a stable system according to Little's Result. And the subscripts like ORR and FPI represent the different load balancing schemes applied. Note that N^* is the result from the optimal policy. The results in the table imply that compared with FPI, FPI* is equally good or even better. Furthermore, it is very close to the optimal result.

The four plots from (a) to (d) in Figure 8.1 correspond to some specific iteration policies, in which the latter one is derived numerically based on the previous policy. Especially, (a) stands for the FPI that has been discussed in Chapter 7. Each plot is completely characterized by a switch-over curve between the dark grey and light grey sections. The x-coordinate represents the number of flows in BS1 while y-coordinate stands for the flow number in BS2. The dark grey part in these plots shows the decisions in which the flexible flows are routed to BS1 while light grey part for situation in which flexible flows to BS2. Some black points in the switch-over curve indicate that it is equal for the flexible flow to be routed to either base station. For each plot (a)-(d), the transition rate matrix is built up according to the policy because from the plot it is straightforward to identify which BS is the destination of the flexible flows when the process is in a certain state. Through the balance equations, the probability that the system stays in each state can be solved out numerically. Then, under a certain policy α, the average cost rate of the system, is the mean number of flows $i_1 + i_2$ in each state multiplies the state probability as

$$\sum_{\hat{i} \in I} \pi_{\hat{i}}(\alpha)(i_1 + i_2),$$

where $\hat{i}$ stands for the state (i_1, i_2). In Figure 8.1.e, a comparison of average delay performance of each policy is then displayed to illustrate how great improvement is achieved after each policy. For ORR, policy α_0, which does not appear in Figure 8.1, the mean cost rate for LB case corresponds to $E[N] = 2 \cdot \frac{\lambda_1 + p\nu}{\mu - \lambda_1 - p\nu}$. From the plot, the curve plunges in the first iteration which stands for the fact that a great improvement is achieved in the first iteration. However, it is in the second iteration that mean delay bottoms to the optimal result.

Figure 8.1.f reports the corresponding load of either base station under each policy iteration in Figure 8.1.(a)-(d). The load under the ORR is calculated as $\rho_i = \frac{\lambda_i + p_i\nu}{\mu_i}$ for BSi while the load under a dynamic algorithm is calculated as $\rho_i = P\{N_i > 0\}$ for BSi, meaning the probability that the considered base station is not empty. From the plot, the loads of the two BSs are the same because ORR tends to evenly distribute the flexible flow by an optimal probability. However, when the system achieves the lowest mean queue length the load of BS1 is higher than that of BS2.

Figure 8.2 correspond to the NLB case ($\nu > \lambda_1 - \lambda_2$), where $\mu = 25$, $\lambda_1 = 15$, $\lambda_2 = 8$, $\nu = 5$. From (a)-(d), each plot stands for a policy while the notation of the figure is the same as that in the LB case. Also, the method to calculate the average queue length of system under a certain policy α is the same. A slight difference lies in the calculation of the $E[N]$ of the policy α_0, ORR. The mean queue length is $E[N] = \frac{\lambda_1}{\mu - \lambda_1} + \frac{\lambda_2 + \nu}{\mu - \lambda_2 - \nu}$, because in the NLB case all the flexible flows are routed to BS2. One notable observation is that, compared that in LB case, the FPI in the NLB case can more closely resemble the optimal policy. It is further verified in Figure 8.2.e, a comparison among different iterated policies, in which the average delay almost bottoms to the optimal result in FPI in the NLB case. Figure 8.3.f verifies that the load of either base station cannot be balanced in the NLB case. The load under the ORR is calculated as $\rho_i = \frac{\lambda_i + p_i\nu}{\mu_i}$. Even though the system distributes all the flexible flows to BS2, the load of BS1 is still higher than that of BS2 in ORR in the symmetric case because $\nu > \lambda_1 - \lambda_2$. In the latter policy iterations, the load is $\rho_i = P\{N_i > 0\}$ for BSi.

In both cases, the iterations bottom to the optimal result completely in the second iteration and significant decrease in the average delay is achieved in the FPI. In the NLB case, a greater improvement is achieved in the first iteration compared with the LB case in Figure 8.2.e. The reason behind this can be explored by reviewing (6.12). In the LB case, the switch-over curve only depends on the flow number in BS1. The load balancing process covers the important information of the difference between λ_1 and λ_2 which can originally aid the optimization in policy iterations. In the NLB case, the flexible stream is not able to completely balance the difference between λ_1 and λ_2. Such unbalanced dedicated flow information takes part in the decision of the switch-over curve to contribute better performance in the first iteration policy in the NLB case compared with LB case.

In brief, from these two figures, the mean delay plunges in the first iteration and bottoms to the optimal result in the second iteration.

8.1.2 FPI* and further iterations

Now, we present the average delay performance of FPI* as described in (6.14) and its iterated policies.

Figure 8.3 is for the LB case while Figure 8.4 is for the the NLB case. Furthermore, we make two pairs of the mentioned figures:

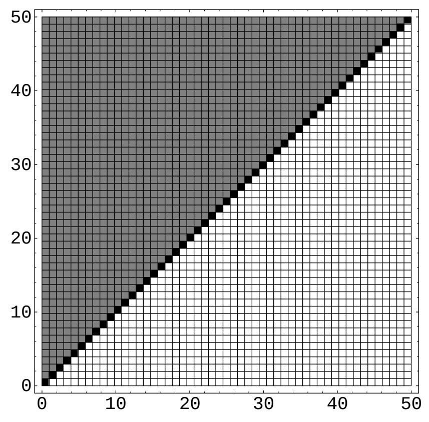
1. Figure 8.1 and Figure 8.3 LB

2. Figure 8.2 and Figure 8.4 NLB,

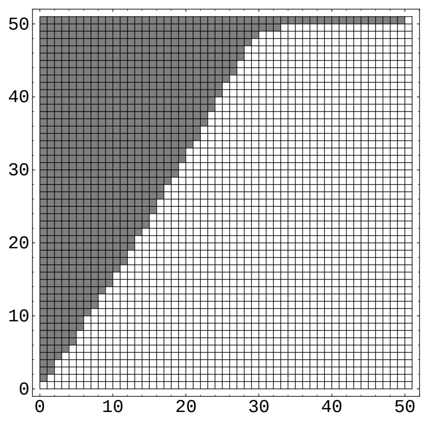
where each pair has the same system parameters.

From FPI and FPI* in each pair, we observe two phenomena. First, compared with FPI, FPI* more closely resembles the optimal policy. Secondly, in Figure 8.3.e and 8.4.e, both FPI*s bottom to the optimal result, representing a better performance than the FPIs. When applying FPI, significant decrease can be achieved based on the ORR policy but it takes another iteration to derive the optimal policy. In short, normally, FPI* can almost achieve the optimal result, which is one step faster than the FPI. Figure 8.3.f and 8.4.f indicate that the load of each BS can also be balanced in the first iteration when FPI* achieves the sub-optimal average delay in the system.

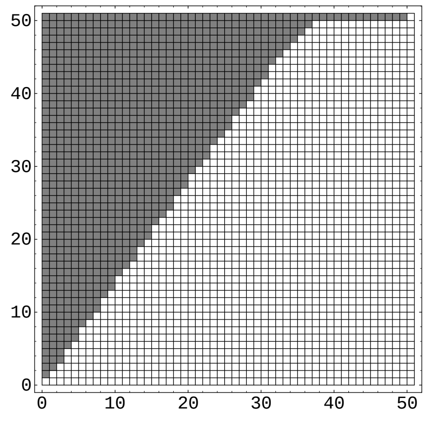
Two tables below summarize these two sets of experiments that we have explored. Table 8.4 is for the experiments in the LB case while Table 8.5 is for the experiments in the NLB case. N in the tables means the average queue length under a corresponding load balancing policy. Both tables verifies that FPI* is superior among all the considered policies except the optimal one.



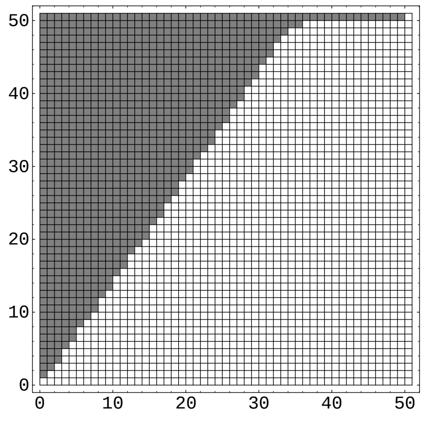
(a)



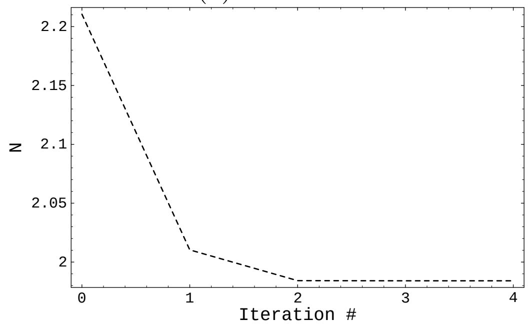
(b)



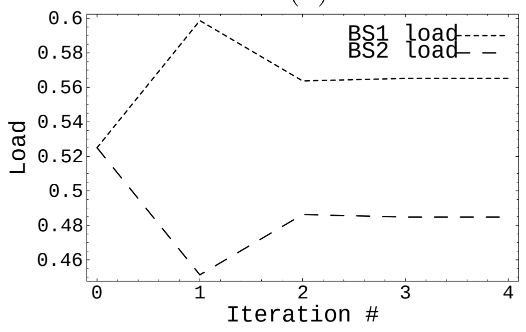
(c)



(d)



(e)



(f)

Figure 8.1: Load-Balance case ($\nu \geq \lambda_1 - \lambda_2$): (a) FPI (b) second policy iteration (c) third policy iteration (d)fourth policy iteration (e) mean queue length of the model vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu = 20, \lambda_1 = 10\lambda_2 = 6$ and $\nu = 5$

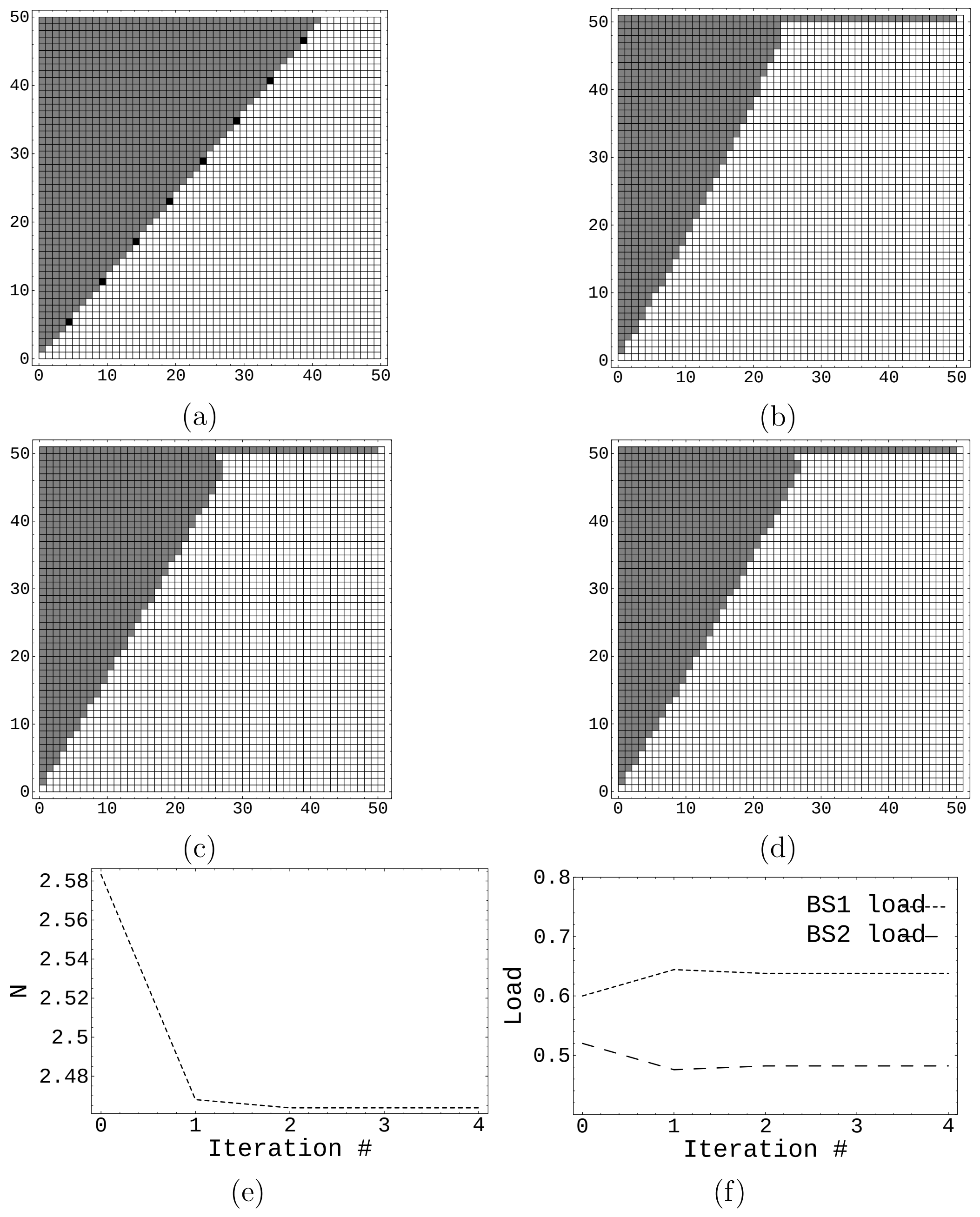


Figure 8.2: No-Load-Balance case ($\nu < \lambda_1 - \lambda_2$): (a) FPI (b) second policy iteration (c) third policy iteration (d) fourth policy iteration (e) mean queue length of the model vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu = 25, \lambda_1 = 15$, $\lambda_2 = 8$ and $\nu = 5$

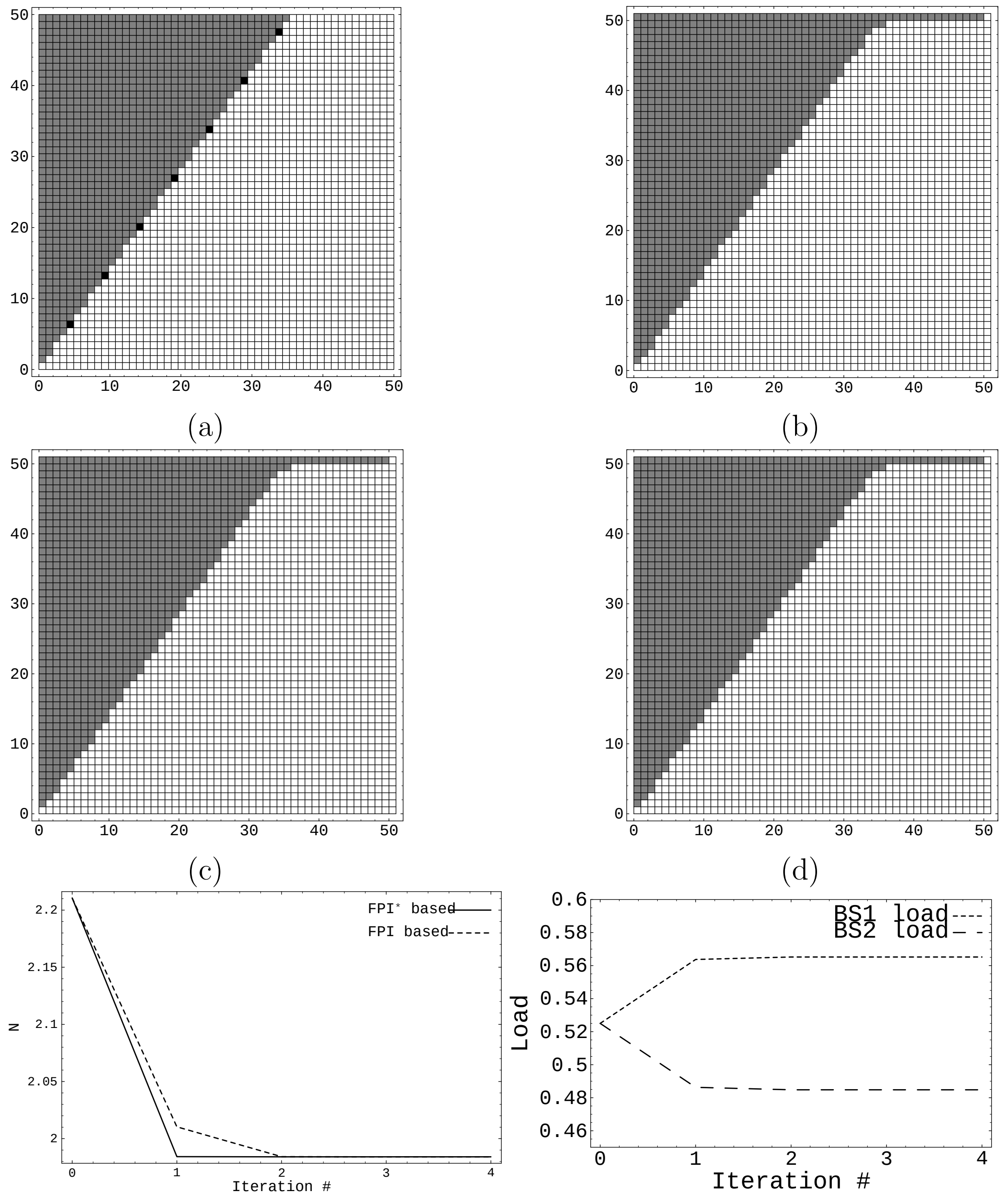


Figure 8.3: Load-Balance case ($\nu \geq \lambda_1 - \lambda_2$): (a) FPI* (b) second policy iteration (c) third policy iteration (d) fourth policy iteration (e) mean queue length of the model vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu = 20, \lambda_1 = 10, \lambda_2 = 6$ and $\nu = 5$

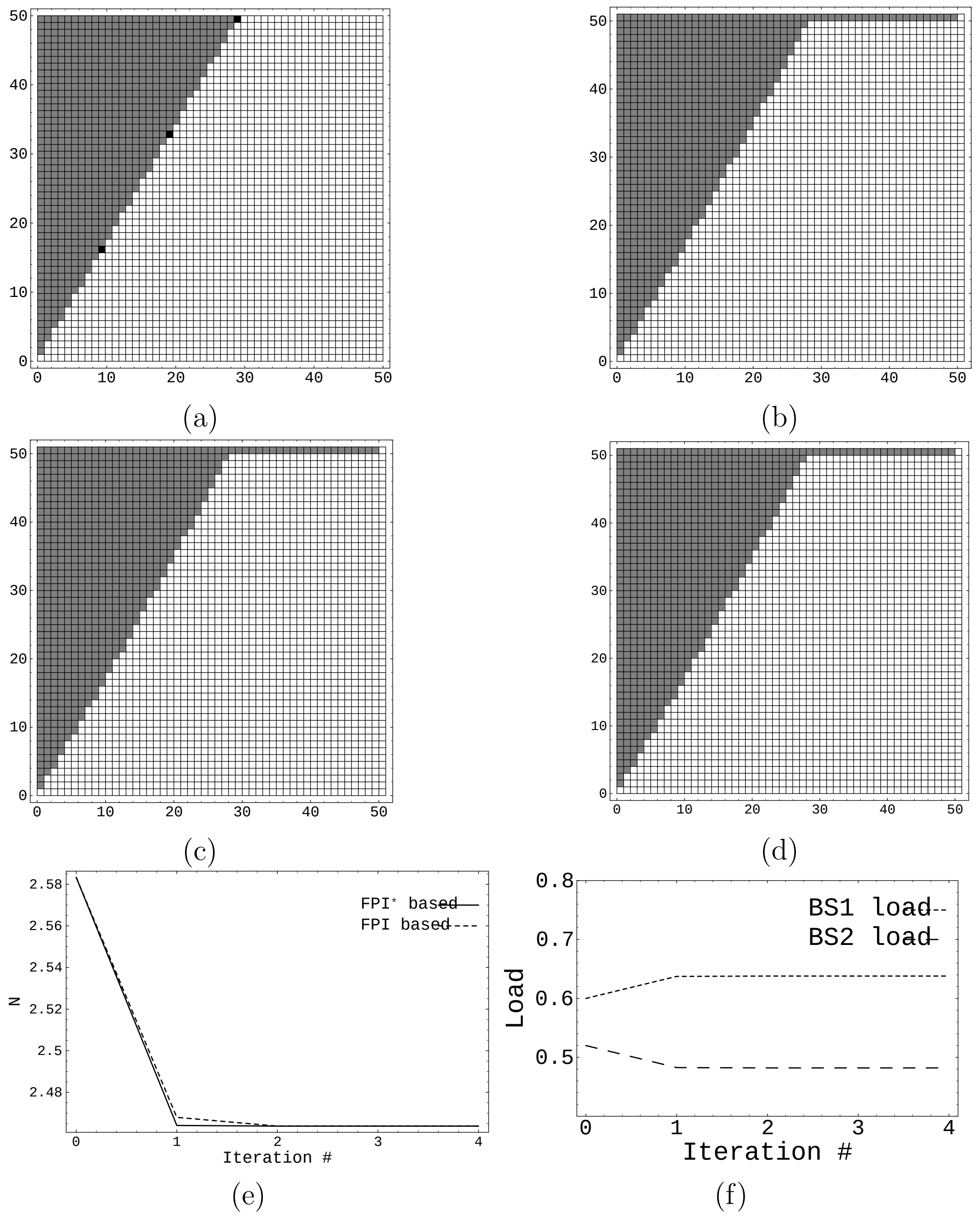


Figure 8.4: No-Load-Balance case ($\nu < \lambda_1 - \lambda_2$): (a) FPI* (b) second policy iteration (c) third policy iteration (d) fourth policy iteration (e) mean queue length of the model vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu = 25, \lambda_1 = 15$, $\lambda_2 = 8$ and $\nu = 5$

$\mu = 20, \lambda_1 = 10,$ $\lambda_2 = 6, \nu = 5$	p^* or switch-over curve	BS1 load	BS2 load	N
ORR	$p^* = 0.1$	0.525	0.525	2.211
JSQ	$l(i_i) = i_1$	0.599	0.451	2.010
FPI	$l(i_i) = i_1$	0.599	0.451	2.010
FPI*	$l(i_i) = 1.4i_1 + 1$	0.564	0.486	1.984
Optimal	$l(i_i) = 1.4i_1 + 1$	0.565	0.485	1.984

Table 8.4: Summary of experiments in the LB case

$\mu = 25, \lambda_1 = 15,$ $\lambda_2 = 8, \nu = 5$	p^* or switch-over curve	BS1 load	BS2 load	N
ORR	$p^* = 0$	0.600	0.520	2.583
JSQ	$l(i_i) = i_1$	0.668	0.452	2.503
FPI	$l(i_i) = 1.2i_1 + 1$	0.644	0.476	2.468
FPI*	$l(i_i) = 1.75i_1 + 1$	0.637	0.483	2.464
Optimal	$l(i_i) = 1.75i_1 + 1$	0.638	0.482	2.464

Table 8.5: Summary of experiments in the NLB case

8.1.3 FPI variations

Figure 8.5.a shows the average queue length $E[N]$ performance of a plenty of the FPI variations, see section 6.2.3 on page 47. The $E[N]$ jumps dramatically around $p = 0$, $f = 1$ area. In Figure 8.1.a, the entire flexible flow is accounted for in the FPI and the optimal probability is $p^* = 0.1$ in those system parameters, which exactly lies in the mentioned convex area near $p = 0$, $f = 1$. For FPI*, it totally ignores the flexible flows, corresponding to $f = 0$ in the same figure. This pocily achieves the minimal average queue length. Thus, it is not surprising that the FPI performance is poorer compared with that of FPI*.

Compared with the FPI (Figure 8.2.a), FPI* (Figure 8.3.a) more likely resembles the optimal policy (Figure 8.2.d or 8.4.d). But performance difference is very slight between FPI and FPI*, either of which greatly decreases and almost achieve the optimal result. This can also be explained by Figure 8.5.b), the optimal probability p^* of FPI is zero according to (7.6) in the NLB case. So the point that corresponds to FPI in Figure 8.5.b is $p = 0$ and $f = 1$. And the line of $f = 0$ can represent the mean queue length performance of the FPI*, which is only a little bit lower than the FPI performance ($p = 0$, $f = 1$). This tiny difference in the the height corresponds to the very small difference of FPI and FPI* manifested by Figure 8.4.e.

Figures 8.5.a and 8.5.b indicate that it always render a better first policy iteration if we ignore the flexible stream totally when considering the marginal cost of an additional flow in (6.8).

8.1.4 Systematic study

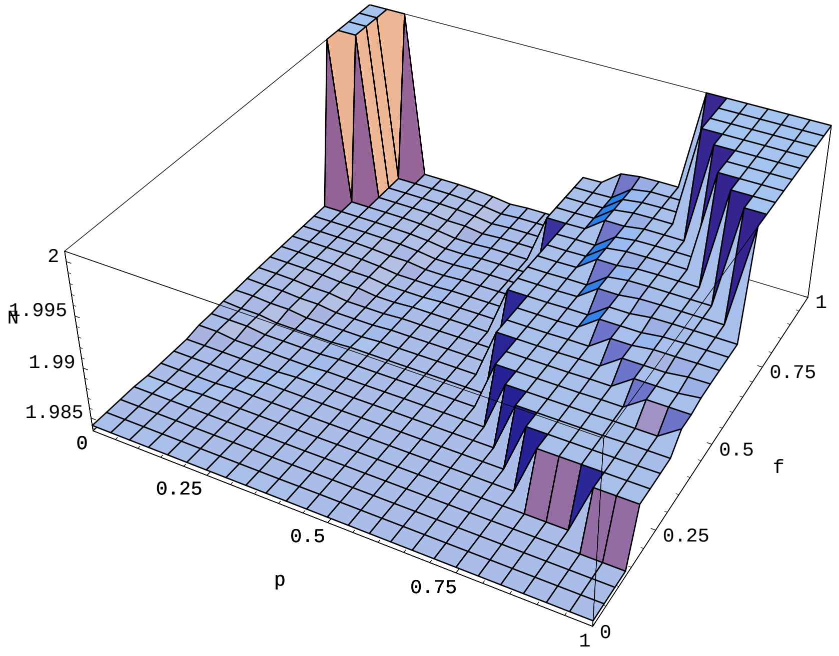
Two systematic experiments are implemented to examine how each allocating policy varies in the mean queue length performance when increasing λ_1 or ν and keeping the other system parameters unchanged. Performance of each policy is normalized by the optimal queue length from the optimal policy. And in these experiments the varying parameters may not always take all the possible values because of numerical reasons.

In Figure 8.6, we fix all the system parameters except the intensity λ_1 of dedicated flow to BS1. It also means that we gradually increase the difference between the two dedicated flows. Note that when we increase λ_1, the load of the system also increases proportionally. ORR, as the basic static policy, has the weakest performance in almost all the situations. As a basic dynamic policy, JSQ achieves much better performance compared with ORR, however, increasing λ_1 also considerably deteriorates JSQ's performance. FPI* is the best policy which almost perfectly resembles the optimal policy which always derived within one or two steps in this case. FPI works well when the system is in the NLB case, however, it conincides with the JSQ in LB case.

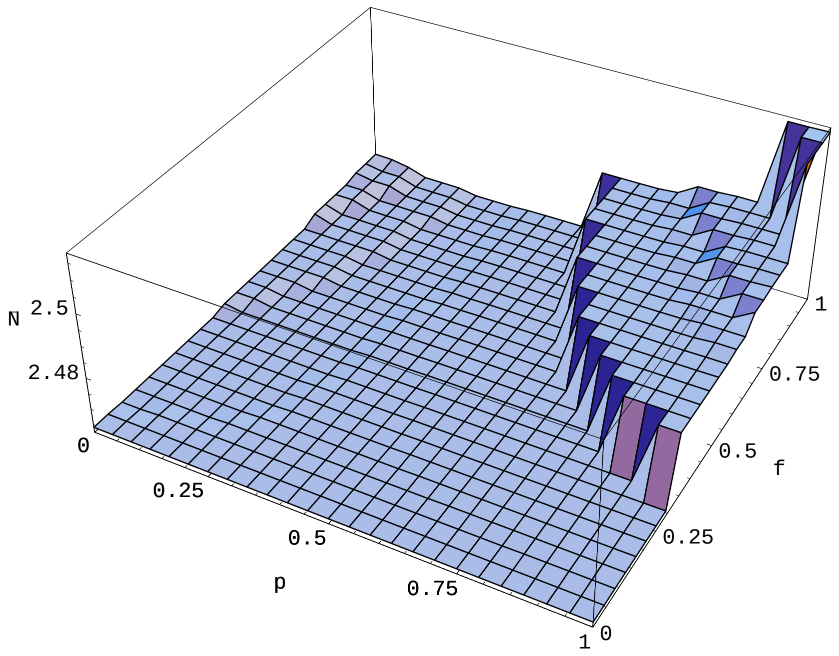
Another experiment is illustrated in Figure 8.7. We increase the intensity of the flexible flow ν while keeping the other parameters unchanged. The ORR still has the poorest performance among these four allocating policies. Moreover, its performance deteriorates proportionally when ν grows higher. In contrast, JSQ and FPI in this case can almost achieve the optimal result. But, inevitably, FPI* performs the best among the four.

8.2 Performance of the asymmetric case

This section gives the numerical experiment results under the asymmetric assumption. Similar to the symmetric section, a comparison is implemented to observe the mean delay performances when applying different truncation



(a)



(b)

Figure 8.5: (a) Load-Balance case ($\nu \geq \lambda_1 - \lambda_2$): Performances of variations of FPI for different combinations of f and p. Parameters: $\mu = 20, \lambda_1 = 10, \lambda_2 = 6$ and $\nu = 5$ (b) No-Load-Balance case ($\nu < \lambda_1 - \lambda_2$): Performances of variations of FPI for different combinations of f and p. Parameters: $\mu = 25, \lambda_1 = 15, \lambda_2 = 8$ and $\nu = 5$

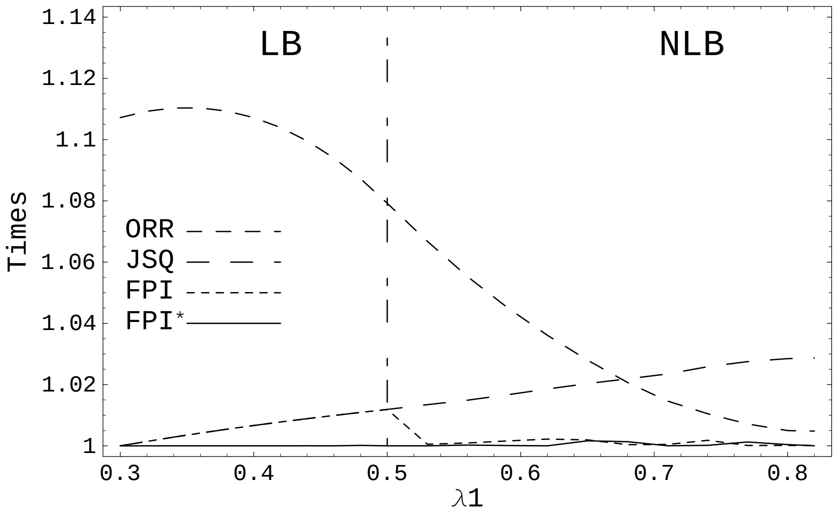


Figure 8.6: Comparison of different policies when increasing $\lambda_1 \in [0.3, 0.82]$ and $\mu = 1.0, \lambda_2 = 0.3$ and $\nu = 0.2$. All results are normalized by the optimal result. The left part corresponds to the NLB case and the right part is the LB case.

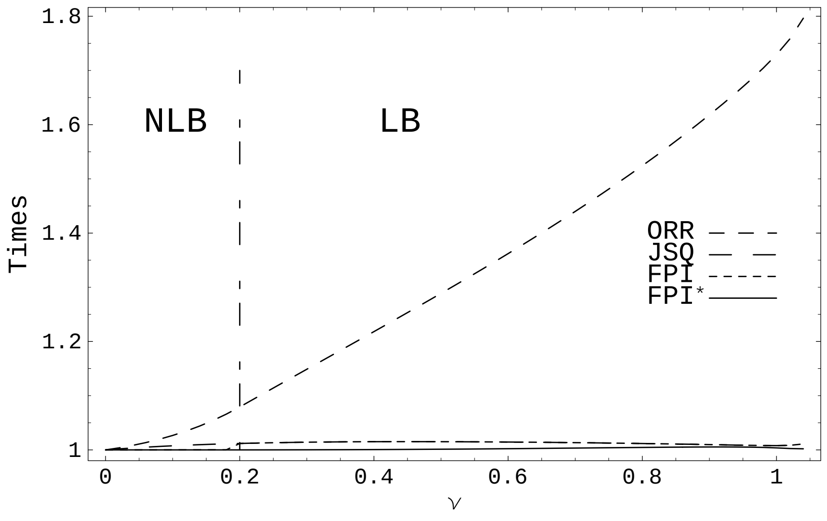


Figure 8.7: Comparison of different policies when increasing $\nu \in [0, 1.2]$, values plotted by divided by optimal result. Parameters: $\mu = 1.0$, $\lambda_1 = 0.5$ and $\lambda_2 = 0.3$. All results are normalized by the optimal result. The left part corresponds to the LB case and the right part is the NLB case.

	N_{ORR}	N_{FPI}	N_{FPI_*}	N^*
50	3.50758	3.23556	3.22496	3.22488
60	3.50758	3.23556	3.22496	3.22488
70	3.50758	3.23556	3.22496	3.22488

Table 8.6: Comparison of performances applying different truncation number with system parameters: $\mu_1 = 25$, $\mu_2 = 15$, $\lambda_1 = 12$, $\lambda_2 = 12$ and $\nu = 5$

number. The results reported in Table 8.6 imply that 50 number of flows in the system is enough to approximate the infinite number scenario. Increasing the truncation number larger than 50 does not influence the entire average flow performance as observed in Table 8.6.

8.2.1 FPI and further iterations

Figure 8.8 presents the FPI and further iterations based on ORR. The mean queue length of ORR is calculated as:

$$E[N] = \frac{\lambda_1 + p\nu}{\mu_1 - \lambda_1 - p\nu} + \frac{\lambda_2 + (1-p)\nu}{\mu_2 - \lambda_2 - (1-p)\nu}.$$

Then, the average number of flows in the system under an iterated policy α is the weighted sum of queue length in each state as:

$$\sum_{\hat{i} \in I} \pi_{\hat{i}}(\alpha)(i_1 + i_2),$$

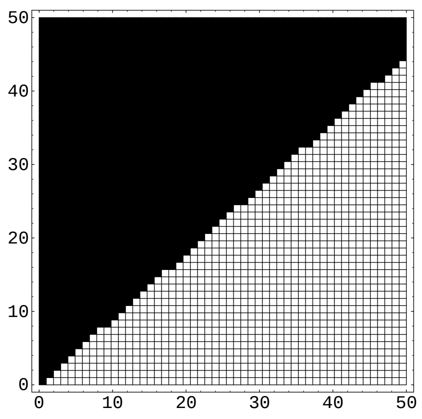
where $\hat{i}$ stands for the state (i_1, i_2). Furthermore, the load for each BS is calculated in the same way as in the symmetric case.

In Figure 8.8.e, FPI plunges greatly, reflecting a significant reduce in the mean queue length from that of ORR policy. The load of each BS does not vary too much even though a great decrease is obtained in the mean queue length.

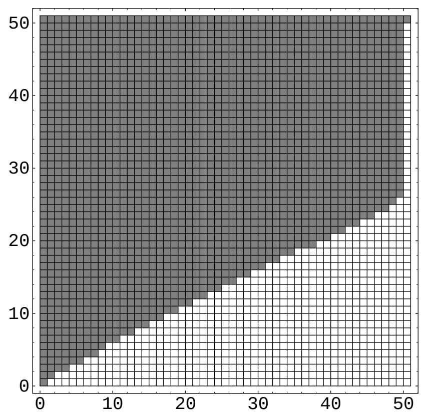
The dedicated flows in this parameter setting are the same ($\lambda_1 = \lambda_2 = 12$) while the capacities of BSs are different ($\mu_1 > \mu_2$). Naturally, the load balancing scheme tends to allocate more flexible flow to BS1 which is associated with a higher capacity. This explains why the system tends to makes decisions to route the flexible flow to BS1 in most states in Figure 8.8.d.

8.2.2 FPI* and further iterations

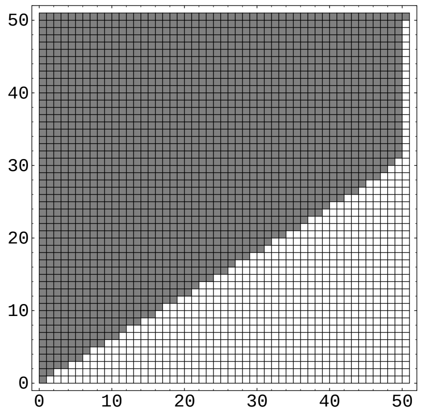
Figure 8.9 is based on the FPI* given in (7.9). Observe FPI and FPI* depicted in Figure 8.8.a and 8.9.a, respectively, we shall see FPI* resembles the optimal



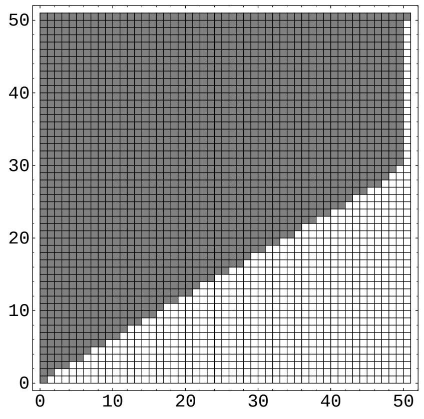
(a)



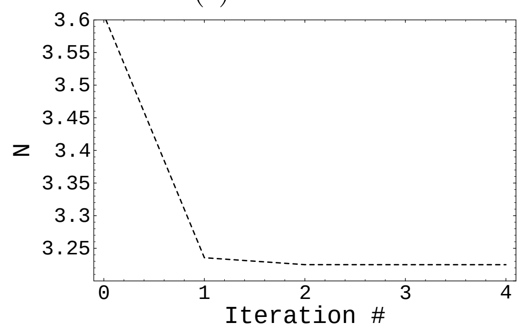
(b)



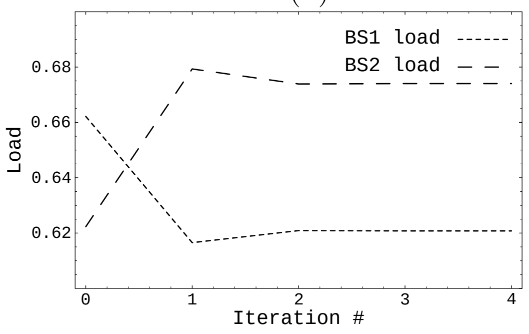
(c)



(d)



(e)



(f)

Figure 8.8: (a) FPI (b) second policy iteration (c) third policy iteration (d) fourth policy iteration (e) mean queue length of the model vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu_1 = 25, \mu_2 = 20, \lambda_1 = 12$, $\lambda_2 = 12$ and $\nu = 5$

$\mu_1 = 25, \mu_2 = 20$ $\lambda_1 = 12, \lambda_2 = 12, \nu = 5$	p^* or switch-over curve	BS1 load	BS2 load	N
ORR	$p^* = 0.911$	0.662	0.622	3.608
LRR	$l(i_i) = 0.8i_1$	0.606	0.693	3.247
FPI	$l(i_i) = 0.88i_1$	0.617	0.679	3.236
FPI*	$l(i_i) = 0.62i_1$	0.626	0.667	3.225
Optimal	$l(i_i) = 0.6i_1$	0.621	0.674	3.225

Table 8.7: Summary of experiments in the asymmetric case

policy much better than FPI does. Despite of the considerable difference in these two policies, in Figure 8.9.e, only a slight improvement in mean queue length is achieved by FPI* compared with the FPI. Besides, the load of each BS does not change too much under ORR or the different iterated policies. From FPI* to the second iteration (Figure 8.9.b), the derivative of the switch-over curve decreases. It also indicates an increase in the number of the states, in which the flows are better to route to BS1. This should lead a rise in the load of BS1 and a decrease in load of BS2 in the meantime. However, the results from the load condition in Figure 8.9.f do not correspond to the arguement above. The unexpected load change is probably caused by numerical problems.

Table 8.7 summarizes the conducted experiments in the asymmetric case, which indicates that the FPI* is a very robust policy. Note that N in the table means the average queue length under a corresponding load balancing policy.

8.2.3 FPI variations

As in the symmetric case, we implement the comparison of different FPI variations.

From Figure 8.10, the mean queue length jumps dramatically around $p \in [0, 0.25]$, $f \in [0.6, 1]$ area. In Figure 8.8.a, FPI takes full consideration of the flexible flow and the optimal probability is $p^* = 0.9108$ from (7.5) in those system parameters. Its performance is slightly poorer compared with the performance of FPI*, where $f = 0$.

It provides the evidence why there is only little progress in Figure 8.9.e between FPI and FPI*.

Figure 8.10 indicates that it also leads to a better first policy iteration in the asymmetric case if we ignore the flexible stream totally when considering the marginal cost of an additional flow in (6.8).

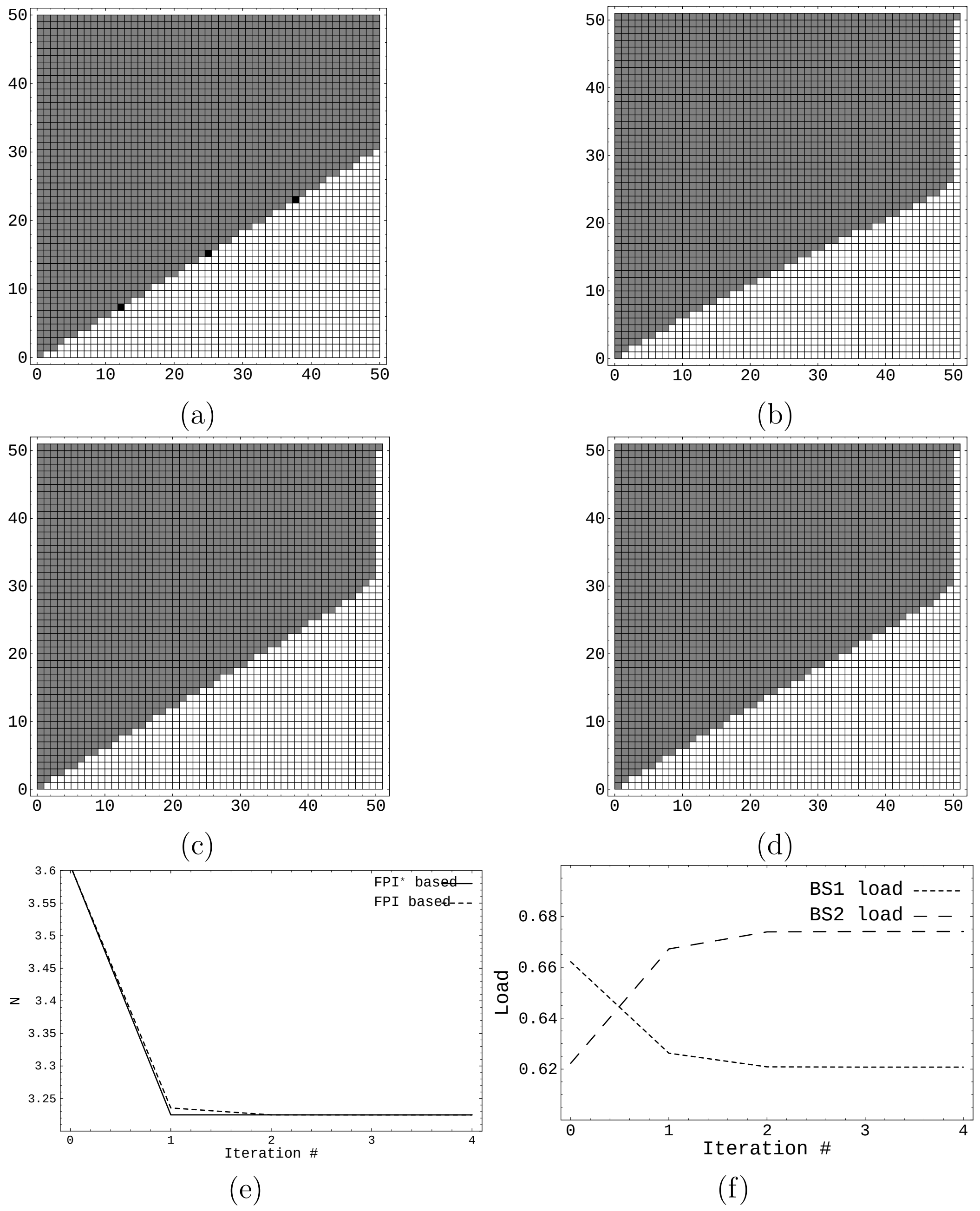


Figure 8.9: (a) FPI* (b) second policy iteration (c) third policy iteration (d) fourth policy iteration (e) mean queue length of the mocel vs. iteration (f) load condition. The dark grey blocks correspond to α=1, the light grey blocks to α=2. System parameters: $\mu_1 = 25, \mu_2 = 20, \lambda_1 = 12$, $\lambda_2 = 12$ and $\nu = 5$

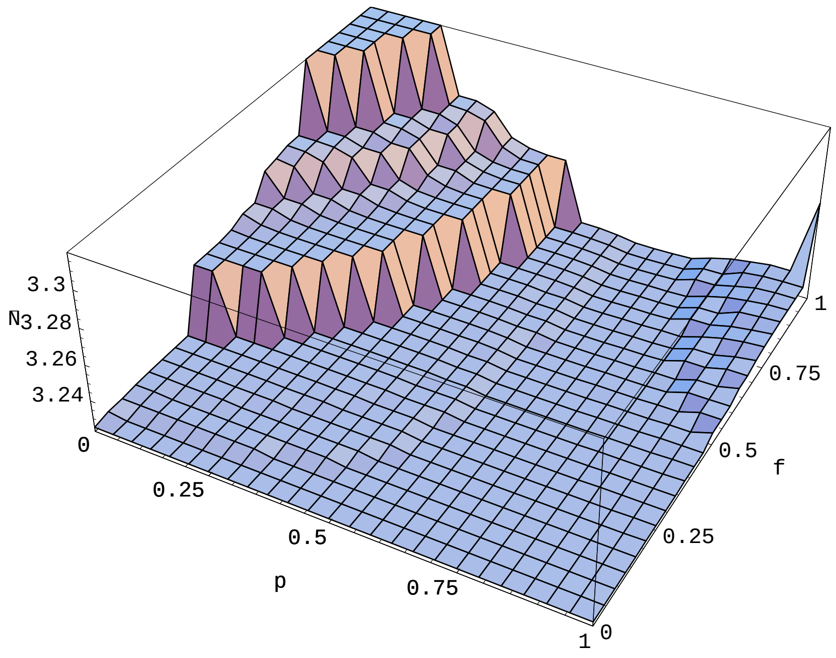


Figure 8.10: Performances of variations of FPI for different combinations of f and p. Parameters: $\mu_1 = 25, \mu_2 = 20, \lambda_1 = 12, \lambda_2 = 12$ and $\nu = 5$

8.2.4 Systematic study

This section exhibits the performances of different load balancing policies, including FPI, FPI*, LRR, JSQ, ORR. They are all normalized by the optimal policy.

Figure 8.11 discloses the difference between policies when the flexible flow intensity increases while keeping the other system parameters fixed. In this experiment, the dedicated flow intensities of each base station are the same. In Figure 8.11, ORR has the weakest performance. This situation gets even worse when ν increases. LRR has the best performance among all the policies in this case. FPI and FPI* both can closely resemble the optimal policy with the growth of the flexible flow intensity. FPI* performs a little bit better than FPI does.

In Figure 8.12, the intensities of dedicated flows are not the same ($\lambda_1 > \lambda_2$). It shows that all the dynamic policies group together and all tend to

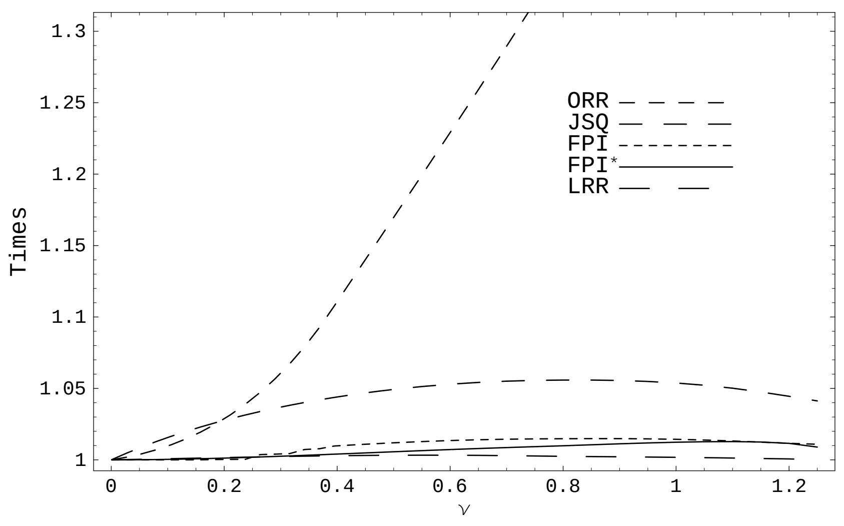


Figure 8.11: Comparison of different policies when increasing $\nu \in [0, 1.25]$. Parameters: $\mu_1 = 1.5, \mu_2 = 1.0, \lambda_1 = 0.5$ and $\lambda_2 = 0.5$.

have a good representation of the optimal policy. JSQ jumps at $\nu = 0.75$, which results in a weaker performance. LRR works as well as FPI* does. Actually, for FPI*, the decision rule is $\frac{n_1+1}{\mu_1-\lambda_1} - \frac{n_2+1}{\mu_2-\lambda_2}$ and in this experiment's context it turns out to be $\frac{n_1+1}{1.0} - \frac{n_2+1}{0.7}$. When both two terms multiple 2/3, the result is given as $\frac{n_1+1}{1.5} - \frac{n_2+1}{1.05}$. It is quite similar to the decision rule of LRR $\frac{n_1}{1.5} - \frac{n_2}{1.0}$ especially when n_1, n_2 are fairly large. In this sense, LRR is as suboptimal as FPI* is.

In Figure 8.13, the performances of different policies are compared when increasing the difference between two dedicated flow intensities ($\lambda_1 > \lambda_2$). Generally speaking, JSQ and LRR perform worse than the other load balancing policies. And FPI and FPI* both closely resemble the optimal policy though near $\lambda_1 = 0.8$, the performance deteriorates a little bit.

The numerical experiments illustrate that FPI* can always resemble the optimal policy closely no matter how the system load varies. In most of the real applications, the number of states tend to grow very large so that the further policy iterations are almost impossible to implement. In this context, FPI* can efficiently be served as a reasonable approximation of the optimal load balancing scheme regarding to its superior mean delay performance at the flow level.

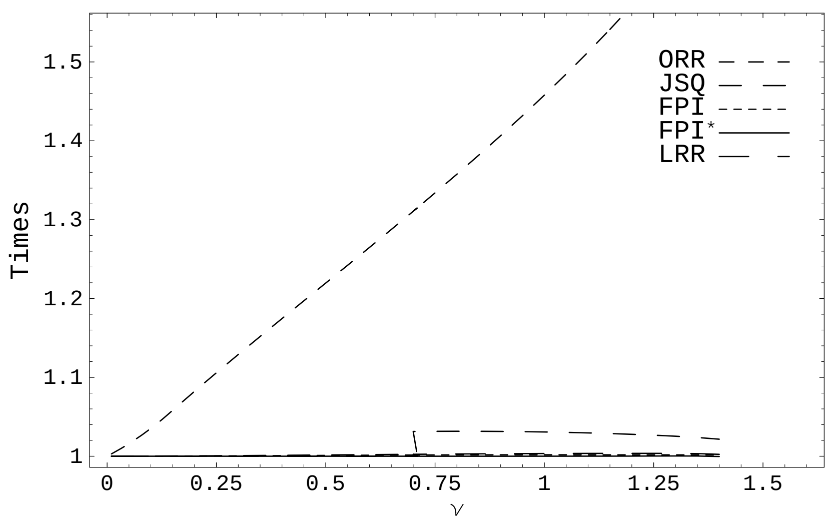


Figure 8.12: Comparison of different policies when increasing $\nu \in [0, 1.45]$. Parameters: $\mu_1 = 1.5, \mu_2 = 1.0, \lambda_1 = 0.5$ and $\lambda_2 = 0.3$.

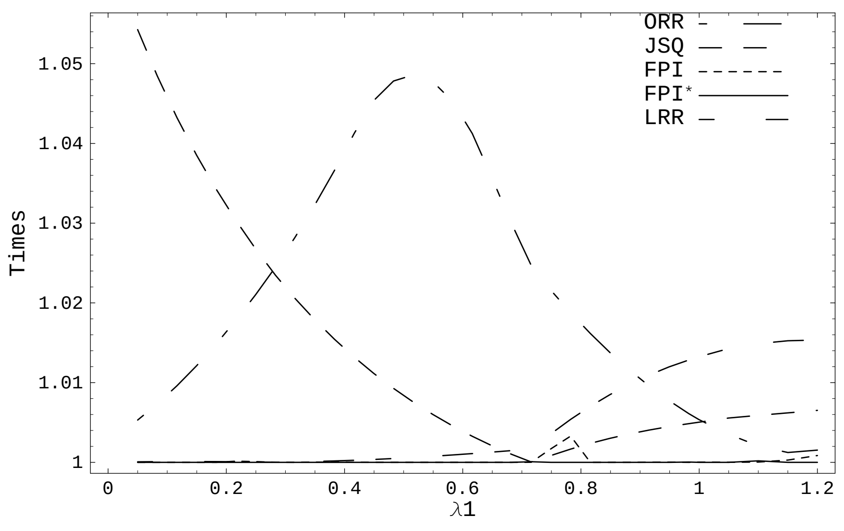


Figure 8.13: Comparison of different policies when increasing $\lambda_1 \in [0.05, 1.2]$. Parameters: $\mu_1 = 1.5, \mu_2 = 1.0, \lambda_2 = 0.2$ and $\nu = 0.1$.

Chapter 9

Conclusion

9.1 Summary

So far we have explored a basic cellular model of two adjacent base stations both in symmetric and asymmetric case at the flow level. And we found that load balancing algorithm can significantly improve the flow delay performance in the cellular network by applying MDP as an optimization tool.

In both cases, ORR as a basic and simple static load balancing scheme is first examined. Later, based on ORR, FPI is derived by comparing marginal costs of an additional coming flow in either base station. Furthermore, FPI*, as a variation of FPI is also obtained by ignoring the flexible flow totally in the comparison. Some other variations of FPI have also been observed but it proves that omitting the flexible flows tends to achieve a greater improvement than the other variations in terms of the first policy iteration.

The results from the numerical experiments are in line with Hajek's conclusion [6] that a switch-over strategy is optimal, minimizing the long run cost. Furthermore, from the experiments, all the dynamic policies can be characterized by a linear curve when we ignore the truncation effect.

In the symmetric case in section 8.1.2, FPI* performs better than FPI does in both the LB and NLB cases. The comparison among different FPI variations proves this observation again in section 8.1.3. However, in the NLB case, the improvement from FPI* is more significant than that in the LB case. It is verified by the figures in the systematic study part later. No matter how we increase the load by adding dedicated or flexible flow intensity, FPI works almost like FPI* does when system in the NLB case. Roughly speaking, ORR has the weakest performance while FPI* is always next to optimal.

In the asymmetric case, FPI performs fairly well as FPI* does though little improvement is achieved by the latter policy. This argument still holds in the systematic study, in which the performance of these two policies are almost the same while the system load varies. FPI* is also more stable policy and the variation of its performance is even negligible compared with other polices.

In brief, FPI* is an effective and efficient load balancing scheme to decrease the mean delay of entire system by balancing the elastic flows in the overlap between two adjacent cells.

9.2 Future work

As mentioned, the optimal policies from the numerical experiments can be characterized by switch-over curves, which corresponds to Hajek's [6] statement. Furthermore, these curves are also linear, which most probably comes from the mathematical relationship between system parameters. Up to now, we have only derived the results in explicit mathematical form for the first iteration based on static policy. If the analytic results for further iterations are obtainable, we can verify our numerical experiments in a converse way.

The flow size in the data network is often modeled as heavy-tailed and the common one is the Pareto distribution [14, 3] rather than the exponential distribution we assume in the thesis. However, such heavy-tailed distributions like Pareto are not easy to analyze because the 'memoryless' property does not hold so that MDP may not apply in this case. Nevertheless, one approach is to approximate such heavy-tailed distribution by the convenient short-tailed distribution like hyperexponential distribution.

For simplicity, only two adjacent cells are considered in the thesis. However, the more real situation could be that flow is generated in the overlapping area among three cells. Generalizing the two cells into more cells might be more practical.

In this thesis, flows are not classified to different priorities. As Borst [4] suggested, it is more real that in certain cases, the flow-level performance may be evaluated by means of a multi-class PS model where the total service rate varies with the total number of the flows. Results in this context may be more practical.

Bibliography

[1] S. Aalto, J. Virtamo, Basic packet routing problem, COMBINE/VTT/WP 1/027/1, April 1994.

[2] M. Alanyali and B. Hajek, “Analysis of Simple Algorithms for Dynamic Load Balancing,” vol. 22, lss. 4, Nov 1997.

[3] S. Ben Fredj, T. Bonald, A. Proutiere, G. R*é*gni*é*, J.W. Roberts, Statistical Bandwidth Sharing: A study of congestion at flow level, Proceedings of ACM SIGCOMN, 2001.

[4] S. Borst, User-level performance of channel-aware scheduling algorithms in wireless data networks, Performance Evaluation vol. 49, issue 1-4, Sep 2002.

[5] G. Bianchi, I. Tinnirello, “Improving Load Balancing mechanisms in Wireless Packet Networks,” Communivations, 2002. ICC 2002, IEEE International Conference on, vol. 2, 28 April-2 May 2002.

[6] B. Hajek, “Optimal Control of Two Interacting Services Stations,” IEEE Transactions on Automatic Control, vol. AC-29, No. 6, June 1984.

[7] H. Holma, A. Toskala, WCDMA For UMTS Radio Access For Third Generation Mobile Communication, John Wiely & Sons, LTD 2001.

[8] R. A. Howard, Dynamic Programming and Markov Process, New York, NY: John Wiley & Sons, 1960.

[9] K. R. Krishnan, “Markov Decision Algorithms for Dynamic Routing.” IEEE Communication Magazine, Oct 1990.

[10] L. Kleinrock, Queueing System, Vol. 2, Wiley, New York, 1975.

[11] G. Koole, “A simple proof of the optimality of a threshold policy in a two-server queueing system,” System & Control Letters 26, 1995.

[12] J. Laiho, A. Wacker, T. Novosad, Radio network planning and optimization for UMTS, Chichester Wiley cop. 2002.

[13] J. v. Leeuwaarden, S. Aalto, J. Virtamo, “Load balancing in cellular networks using first policy iteration,” COST279 TD(02)23, 2002.

[14] L. Massoulié and J. W Roberts, “Bandwidth sharing and admission control for elastic traffic,” Telecommunication System 15, 2000.

[15] M. L. Puterman, Markov Decision Processes Discrete Stochastic Dynamic Programming, New York Wiley, 1994.

[16] J.W. Roberts, “A survey on statistical bandwidth sharing,” Computer Networks, vol. 45, 319-332, 2004.

[17] A. S Tanenbaum, Computer Networks, Englewood Cliffs NJ Prentice-Hall, 1981.

[18] H. Velayos, V. Aleo, G. Karlsson, “Load balancing in overlapping wireless LAN cells,” Communications, 2004 IEEE International Conference on, vol. 7, 20-24 June 2004.

[19] J. Virtamo, Teletraffic theory Lecture notes, Helsinki University of Technology, Spring 2004.

[20] W. Winston, Optimality of the shortest line discipline, J.Appl. Prob. 14, 181-189, 1977.

[21] A. Y. Zomaya and Y.H. Teh, “Observations on using genetic algorithms for dynamic load-balancing,” IEEE Trans. Parallel and distributed systems, vol. 12, no. 9, Sep 2001.