Analysis, Design and Control of Communication Systems

First Edition

Ernst Nordström

Department of Culture, Media, Computer Science, Dalarna University

SE-781 88 Borlänge, Sweden

eno@du.se

February 21, 2006

Contents

Chapter 1

Communication principles

1.1 Single-service communication networks

The remote communication model shown in Figure 1.1 shows how users are interconnected via access and core (WAN) networks. The access networks are either wired or wireless, but the core network is typically wired, except when the WAN is implemented using satellites. The remote communication model applies to both telephone networks and the global Internet.

Traditionally, WAN communication networks have developed along two tracks: circuit-switched telephone networks and packet-switched data networks. In the former, communication is carried out over “circuits” with dedicated transfer capacity or bandwidth.

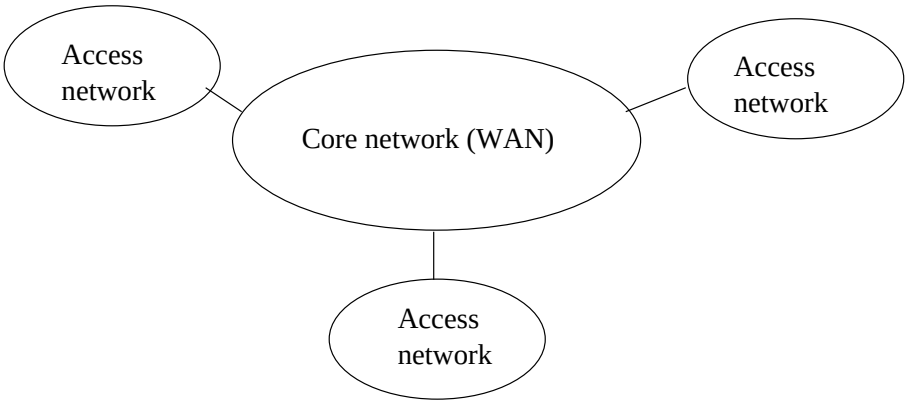


Figure 1.1: Remote communication model

Apart from constant throughput, circuit-switching also gives 100 % reliability (zero information

loss), constant delay and zero delay variability (jitter). In the latter track, information is broken up into pieces of 10s to 1000s of bytes. Each piece is added control information which helps the packet-switching network to guide or route the packet to the appropriate destination. At each switch, the packet may be broken up into smaller pieces which are encapsulated into data link frames, for transmission to the next switch or the terminal.

User data terminals, attached to a packet-switched network, send packets asynchronously, according to their current communication needs. Although the packets are inserted in the network on an asynchronous basis, the transmission of the packets is synchronous. A packet-switch connects multiple input links to multiple output links. Without any special mechanisms switching conflicts would arise when packets from input links simultaneously request access to the same output link. Different solutions with input queues, central queues, and/or output queues are possible. Switch buffers have finite size and excessive periods of overload would result in "buffer overflow" forcing packets to be discarded. The queueing of packets in the switches also introduce extra delay and jitter.

The anatomy of a packet switch with input buffers and output buffers is shown in Figure 1.2.

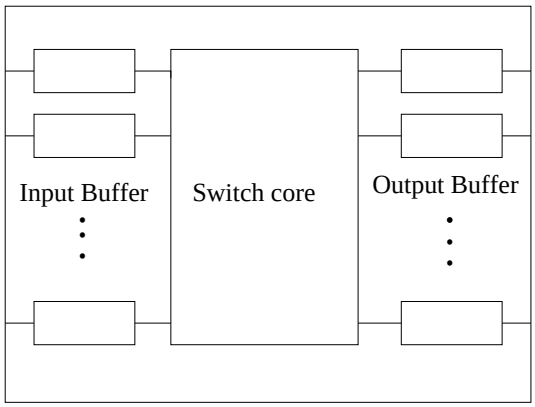


Figure 1.2: Anatomy of a packet switch/router

In connectionless information transfer, the packets are sent into the network without any prior overhead due to connection set up. Each packet can chose its own path to the destination. That is, a packet switch might forward packets with the same source and destination address to different output links. In the connectionless packet-switching mode there is not possible to obtain tight bounds on the packet loss probability, packet delay or jitter.

The global Internet is the most well known example of a connectionless packet-switched network. It is primarily used for data transport purposes such as transfer of files and world wide web documents.

Internet is defined as the world-wide network of smaller networks inter-connected using the Internet Protocol (IP). IP is a network protocol based on connectionless switching of variable-size packets. The packet switch in IP networks is called *router*. The main drawback of the Internet used today is that it only provides best-effort IP service. Best-effort means that the network does its best to deliver the packets to the destination host. However, there is no guarantees on certain packet delay and jitter, or to say the least, on successful packet delivery. Reliability in Internet is provided by a transport protocol called *Transmission Control Protocol* (TCP). TCP is an end-to-end protocol, only implemented at the network hosts.

In connection-oriented information transfer the communication is carried out over connections, called *virtual circuits* (VCs) in packet-switched networks. The packets sent over a VC will normally use the same path to the destination. Traditional packet-switched networks such as X.25 do not reserve any resources at VC set up. Reliability is based on error detection and error recovery. Error detection and error recovery can in general be provided on three levels: link level, network level, or end-to-end (host) level.

Error detection is based on giving each packet a sequence number, in order to detect lost packets, duplicate packets, and packets out of sequence. Bit errors can be detected by a checksum.

Error recovery is implemented by *forward error correction* (FEC) or *backward error correction* (BEC). FEC is suitable for real-time services which have no time for retransmissions. FEC is based on error correcting codes such as parity codes and Reed-Solomon codes. BEC is suitable for non-real-time services. BEC-based error recovery relies on positive and/or negative acknowledgments. Normally, a timer is set when the packet is transmitted. If the positive acknowledgment takes too long time before it reaches the sender, the timer will expire and the corresponding packet be re-transmitted. If the round trip time has a large variance compared to its mean, the timers can not be tightly set, in which case negative ackowledgements will speed up the re-transmission process.

Flow control adapts the sender's transmission rate to the available storage and processing capacity at the receiver. Error detection, BEC and flow control are often integrated.

Early LAN networks for data communication include Ethernet networks, introduced in the mid 70s. Ethernet connects stations via a shared media. The media access scheme is said to be of the broadcast type. This means that destinations can be reached without switching, by sending frames on media which all stations listens to. No real time service is provided. The users can chose between unreliable and reliable Ethernet data service. Reliability is based on error detection and BEC. Ethernet

can also provide flow control.

1.2 Multi-service communication networks

From the start of the data communication era in beginning of the 70s and until the mid-80s physically separate WAN networks was used for data- and telecommunication. This was changed in 1984 when the International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), standardized the Narrowband Integrated Services Digital Network (N-ISDN). The goal of N-ISDN was to provide voice and nonvoice services over a digital circuit-switched network. N-ISDN connections can have a bit rate of 144 kbps in case of basic access (US and Europe), and 1488 kbps (US) or 1936 kbps (Europe) in case of primary access.

Broadband ISDN was standardized in 1988 by ITU-T. B-ISDN was anticipated to become an universal network providing any kind of communication service, including multimedia service. Asynchronous Transfer Mode (ATM) was chosen by ITU-T as the switching and multiplexing technique for implementing B-ISDN. ATM is based on switching of fixed-sized packets (cells) over virtual circuits. ATM cells are transferred over a synchronous (slotted) time division multiplex (TDM) fiber channels. The TDM channels have bandwidths of multiples of 155 Mbps.

In order for Internet to become a truly multi-service network suitable for the 21st century, it must be extended with real-time service capabilities. To this end, in the late 90s, the Internet Engineering Task Force (IETF) standardized two complementary Quality of Service (QoS) architectures or frameworks called Integrated Services (IntServ) and Differentiated Services (DiffServ). Both these architectures define an IP flow as a form of connectionless equivalent to a VC that carries packets with same QoS requirements between a certain origin-destination host pair. The future Internet is assumed to use ‘’route pinning”, i.e. not to change the route for successive packets within a flow unless necessary.

LAN and MAN multi-service networks were introduced in the late 80s and early 90s. Token bus LANs both provide real-time service as well as data service. The same is true for FDDI and DQDB MAN networks. All these LANs and MANs interconnects the hosts over a shared medium.

Multi-service ATM networks, IP networks, broadcast LANs and MANs all rely on reservation of resources for real-time traffic. In broadcast LANs and MANs resources are allocated deterministically. Hence, it is possible to obtain zero frame loss, and worst case bounds on delay and jitter. In packet-

switched networks such as ATM and IP networks, resource usage is based on *statistical multiplexing* or *deterministic multiplexing*. In the former, the network takes advantage of that different users will have overlapping periods of high and low bandwidth demand. The network need not to reserve capacity according to the aggregate peak demand, but to a lower demand, closer to the aggregate average bandwidth demand. Tight bounds on the packet loss probability, packet delay and jitter are possible in statistical multiplexing. In deterministic multiplexing the network allocate resources according to the aggregate peak demand of the users. The result is virtually zero packet loss and worst case bounds on delay and jitter.

Multi-service packet-switched networks will also monitor and enforce the traffic stream entering the network to make sure that the declared traffic parameters are not violated. Excessive packets will be discarded or marked as low priority. During congestion periods in the network, packet switches first drop the low priority packets.

Multi-service networks must charge the users for their use of the network in order to avoid that too many users request the best and most expensive services. It is believed that usage-based charging, in contrast to flat-rate charging, is most suitable for multi-service packet-switched networks.

1.3 Layered services and protocols

Communication in packet-switched networks is carried out by means of protocols implemented by the switching nodes and the terminals or hosts. A *protocol* is a set of rules defining how information should be exchanged between two entities. Protocols normally are organized in a layered model. Each layer has its own protocol.The details of the protocol on a certain layer is hidden for other layers. Layer $n+1$ use the service of the layer n immediately below it. The entities comprising the corresponding layers on different machines are called *peers*. In other words, it is the peers that communicate using the protocol. Between each pair of adjacent layers there is an *interface*. The interface defines which primitive operations and services the lower layer offers to the upper layer.

The set of layers and protocols is called *network architecture*. Neither the details of the protocol implementation nor the specification of the interface are part of the network architecture. The set of protocols used by one system with one protocol per layer is called *protocol stack*.

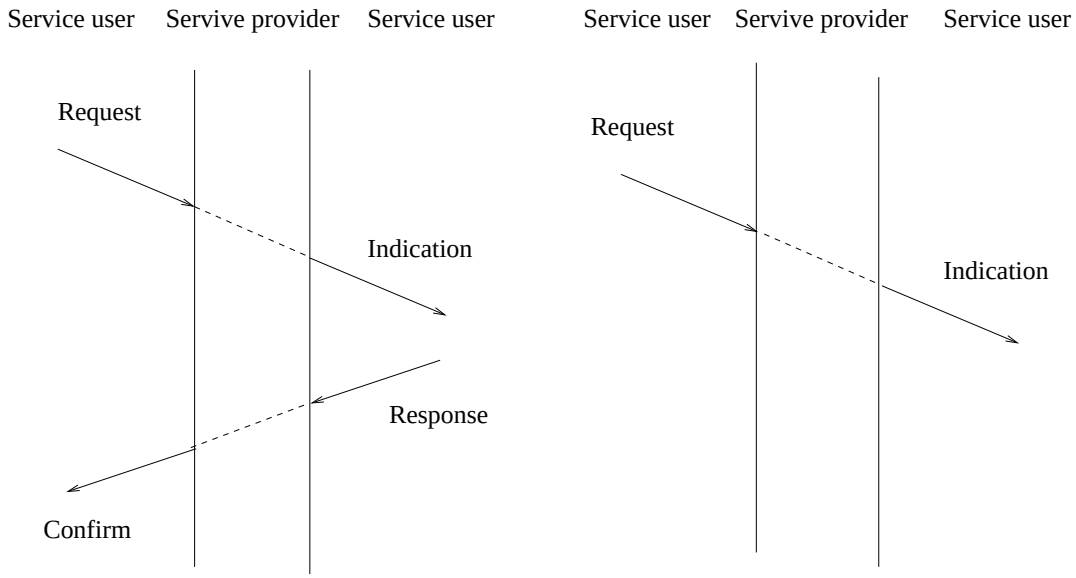
Services are available at *Service Access Points* (SAPs). Each SAP has an address that uniquely identifies it. To make things clearer, the SAPs in the telephone system are the sockets in which modular

telephones can be plugged, and the SAP addresses are the telephone numbers of these sockets.

There are four basic types of service primitives:

- **Request**: A primitive issued by a service user to invoke some service and to pass the parameters needed to specify fully the requested service.
- **Indication**: A primitive issued by a service provider either to:
  1. indicate that a procedure has been invoked by the peer service user on the connection and to provide the associated parameters, or
  2. notify the service user of a provider-initiated action.
- **Response**: A primitive issued by a service user to acknowledge or complete some procedure previously invoked by an indication to that user.
- **Confirm**: A primitive issued by a service provider to acknowledge or complete some procedure previously invoked by a request by the service user.

Figure 1.3 shows the time sequence diagrams for confirmed and non-confirmed service.



(a) Confirmed service (b) Nonconfirmed service

Figure 1.3: Time sequence diagrams for service primitives.

The primitives invoked on layer $n+1$ take control and data parameters as input and pass them as a layer n *Interface Data Unit* (n-IDU) to layer n. The data consists of the layer-n *Service Data Unit* (n-SDU). The control parameters are of two types. First, control parameters can be for internal layer control, e.g. the number of bytes in the n-SDU. Such control parameters are contained in the layer-n *Interface Control Information* (n-ICI). Second, control parameters can contribute to *Protocol Control Information* (n-PCI) to layer n. An example of such control parameters are the source and destination host address. The n-PCI forms together with the n-SDU form the layer-n *Protocol Data Unit* (n-PDU). The PCI are conveyed in a PDU header and/or trailer. It is used by the peer entities to carry out their peer protocol. They identify which PDUs contain data and which contain control information, provide sequence numbers, checksums and so on.

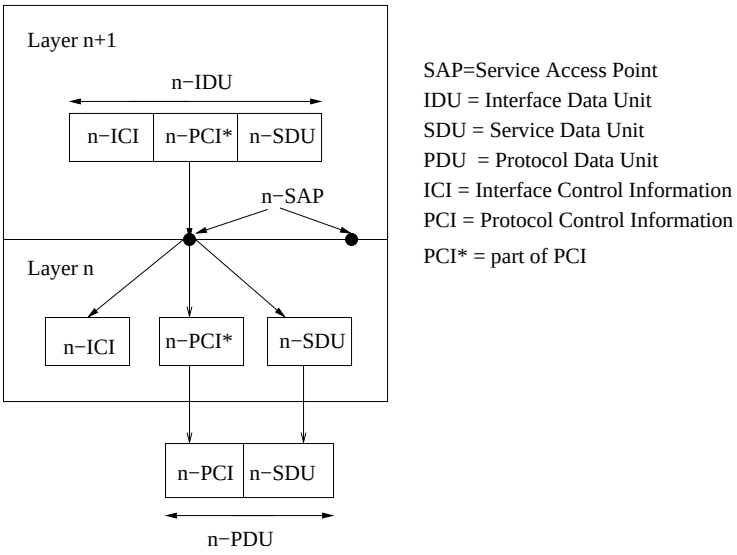


Figure 1.4: Relation between layers at an interface

1.4 ISO OSI reference model

The International Standards Organization (ISO) proposed in 1984 a reference model for layered organization of communication protocols. The model was called the the Open Systems Interconnection (OSI) reference model. The ISO OSI model contains seven layers, see Figure 1.5.

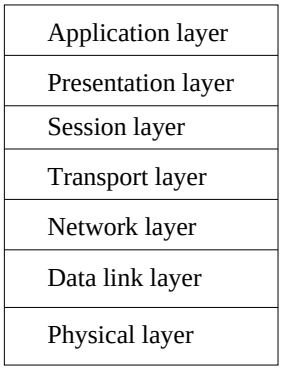


Figure 1.5: The ISO OSI reference model

1.4.1 Physical layer

The physical layer is concerned with transmitting raw bits over a communication channel. The design issues large deal with mechanical, electrical, and procedural interfaces, and the physical transmission medium, which lies below the physical layer. Issues include reliable transfer of bits, modulation of the signal, bit voltage levels, time duration of a bit, how many pins the network connector has, and what each pin is used for. A final issue is security, e.g. detection of wiretapping.

1.4.2 Data link layer

The data link layer takes the raw transmission facility and transforms it into a line that appears free of undetected transmission errors to the network layer. The data link layer performs framing, which collects the bits from the physical layers into frames, typically contain 10s to 1000s of bytes. The framing can be done with a special bit sequence that detects start and end of the frame. The data link layer also deals with error detection, error recovery and flow control, and security in the form of privacy, integrity and authentification.

Broadcast networks have two additional issues in the data link layer: how to control access to a shared channel and how to address the hosts attached to the shared channel. The access control deals with issues such as bandwidth and buffer scheduling. A special sub layer of the data link layer, the medium access control (MAC) sub layer, deals with this problem.

1.4.3 Network layer

The network layer is responsible for functions such as addressing, call admission control, routing, fragmentation and reassembly, error detection, error recovery, flow control, congestion control, traffic enforcement (policing), traffic shaping, scheduling of bandwidth and buffers, network dimensioning, charging and security.

1.4.4 Transport layer

The transport layer is the first end-to-end protocol in the reference model. It is only implemented by the hosts, not inside the network. Its main purpose is to facilitate reliable transport of transport PDUs called segments between hosts. Transport protocols enhances the unreliable service of the network layer. Its functions include error detection, error recovery, flow and congestion control, user process identification, and security. Some transport protocols do not provide error recovery. If the transport connection requires high throughput, the transport layer might create multiple network connections to improve throughput. On the other hand, if creating or maintaining a network connections is expensive, the transport layer might multiplex several transport connections onto the same network connection.

1.4.5 Session layer

The session layer allows users on different hosts to establish *sessions* between them. A sessions allows ordinary data transport, as does the transport layer, but it also provides enhanced services useful in some applications. A session might be used to allow a user to log into a remote timesharing system or to transfer a file between two hosts. One of the services of the session layer is to manage *dialogue control*. Sessions can allow traffic to go in both directions at the same time, or in only one direction at a time. A related session service i *token management*. For some protocols, it is essential that both sides do not attempt the same operations at the same time. To manage between activities, the session layer provides a token that can be exchanged. Only the side holding the token may perform the critical operation. Another service is *synchronization*. The session layer may provide a way to insert checkpoints into the data stream, so that after a system crash, only the data transferred after the last checkpoint have to be repeated.

1.4.6 Presentation layer

The presentation layer is, among other things, concerned with the syntax and semantics of the information transmitted. A typical example is encoding data in standard agreed upon way. Most user programs do not exchange random binary strings. They exchange things such as people's names, dates, amounts of money, and invoices. These items are represented as character strings, integers, floating-point numbers, and data structures composed of several simpler items. In order to make it possible for computers with different representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire". The presentation layer manages these abstract data structures and converts from the representation used inside the computer to the network standard representation and back.

The presentation layer also is responsible for coding and compression of images, audio and video. Compression is used to reduce the bandwidth requirement of multimedia packet streams. Compression works by reducing the redundancy in the information flow.

1.4.7 Application layer

The application layer contains a variety of protocols that are commonly needed. One service is the *network virtual terminal*. Similar to having abstract data structures, a network virtual terminal is a technology independent terminal that has general functionality. Special software in the application layer maps the virtual terminal functions onto the real terminal. A second application layer service is *file transfer*. Different file systems have file naming conventions, different ways of representing text lines, and so on. Transferring a file between two systems requires handling these and other incompatibilities. Other application layer services are electronic mail, word wide web (WWW), remote job entry, directory lockup, and multimedia. Security is also an issue in the application layer, such as secure transfer of files, electronic mails and WWW documents.

Chapter 2

History of communication networks

2.1 Telephone networks

Ever since the advent of the telephone in 1876 by Alexander Graham Bell, voice communication between distant locations have been possible. The first telephone networks introduced a few years later consisted of manually operated switching offices connected to each others and to the customers' telephones.The copper-based twisted pair was introduced in the local loop between the customer and the local switching office in the 1890s. The human operators where replaced by electro-mechanical switches in the 1940s. The advent of the transistor in 1948 paved the way for computer controlled switches which were introduced in the 1960s. Starting in the 1940s until the 1980s, the local, regional and central switches were inter-connected by coax cable. In the mid-1980s high-speed optical fibers were introduced in the switching network.

2.2 Internet

Paul Baran, employed at the RAND corporation in US, introduced in 1962 the concepts of packet-switching and distributed networks in his attempt to design a fault-tolerant US communication network which should survive a nuclear war. September 1969 marked the birth of ARPANET, the predecessor of Internet. ARPANET connected US universities which were supported by the US Department of Defense. In the late 70s the first version of the Internet Protocol (IP) and the Transmission Control Protocol (TCP) were introduced in ARPANET. In 1983 ARPANET became Internet. The definition of Internet being the global inter-connected collection of smaller networks which implements the

TCP/IP protocol suite still holds today. The number of hosts in the Domain Name System (DNS) has evolved from 200 in 1981 to about 320 million in January 2005. The number if autonomous systems (domains) was 16,000 in July 2005, and the number of IP subnetworks was 210,000. The history of Internet applications has landmarks such as the introduction of the first email system in 1971 and the introduction of world wide web, invented at CERN, in 1991. Early attempts of video conferencing and IP telephony over Internet were carried out in the late 1990s.

2.3 Data networks

Data networks offers a packet-oriented transport service without any guarantees on transfer delay. Historically, data network standards have evolved for both LANs and WANs. The first LAN standard, the Ethernet standard, came in 1976. Later, in 1985, the Token ring LAN standard was approved. In both these two LAN standards, stations are connected to a shared media which they access according to some control scheme. The first WAN standard, the X.25 standard, was approved by the ITU in 1976. X.25 is a packet- and connection-oriented transfer technique offering low bit rate (64 kbps) connections. Most Automatic Teller Machines (ATMs) are connected through X.25 over the D-channel using N-ISDN basic access. The Switched Multimegabit Data Service (SMDS) is a WAN network standard primary aimed at LAN connectivity. SMDS was developed by Bellcore in the early 1990s. SMDS is packet-oriented and connectionless.

2.4 Wireless networks

The first radio transmission across Atlantic Ocean was demonstrated in 1901 by Marconi. In 1920 the first public radio transmission took place in Germany. In the 1920s police radio in cars were introduced in metropolitan New York area. In 1946 the first mobile telephone service in USA was introduced by AT&T. In 1949 Claude Shannon et al. developed the basic ideas of CDMA. In 1979 and 1981 the first generation (1G) analog mobile systems AMPS and NMT was introduced in USA and Sweden, respectively. Second generation (2G) mobile systems such as GSM, IS-95 and PDC were introduced in 1991 (Europe), 1993 (USA) and 1994 (Japan), respectively. In 2000 countries all over the world gave out licenses to network operators for operation of third generation (3G) mobile systems. European 3G mobile systems will be based on UMTS. The UMTS networks will be interoperable with current GSM/GPRS networks. Important events in the history of wireless LANs are the

introduction of the wireless Ethernet (IEEE 802.11) in 1997 and Bluetooth in 2000.

2.5 Multi-service networks

Multi-service networks have an important advantage over pure data networks, namely real-time service capabilities. Multi-service networks have evolved since the mid 1980s and standards exist today for LANs, MANs and WANs. The Token bus LAN standard was approved in 1985. An important application environment for Token bus is factory automation. The FDDI and DQDB MAN network were developed during the early 1990s. Multi-service WANs include Frame relay, Narrowband ISDN, Broadband ISDN/ATM and the QoS enhanced Internet. The Frame relay standard was developed in 1984 and was first seen as a competitor to the X.25 standard. Frame relay is designed to be efficient for fiber communication characterized by low transmission error rates. The N-ISDN and B-ISDN are standards of the ITU approved in 1984 and 1988, respectively. N-ISDN is circuit-switched and offers bandwidths up to 2 Mbps. B-ISDN is based on ATM which is a packet-switched technology. B-ISDN offers high bit rates, starting from 155 Mbps. It is not clear when (and even if) B-ISDN will offer international connectivity. If this happens B-ISDN will be a direct competitor to the QoS enhanced Internet, which is believed to be gradually introduced in the coming decade.

Chapter 3

Overview of resource management

3.1 Introduction

In this chapter we present an overview of resource management in multi-service networks. The starting point is a network consisting of a set of nodes, interconnected by links according to some network topology. In case of ATM or IP networks the network topology is often partly meshed. We concentrate on resource management on the OSI network and transport layers. Such a layer is either connection-oriented or connectionless. We use the term *virtual circuit* (VC) to denote a network layer connection in an ATM network, and the term *flow* to denote the connectionless equivalent to the VC in an IP network. The ATM VC is identified by the VPI/VCI value in the ATM cell header. The IP flow is identified by the source and destination IP address, source and destination port number and the transport protocol (TCP/UDP). Alternatively, the flow identifier in the IPv6 packet header can be used. Often, the presentation refers to both ATM VCs and IP flows. In this case we use the term *call* to refer to both ATM VCs and IP flows.

We use an hierarchical model to describe traffic in the network [38]. The model consists of three layers operating on different time scales, see Figure 3.1. The upper layer is the *call layer* which operates in a time scale of seconds. The traffic on this layer is characterized by the call arrival process and the call holding time distribution. Calls generate packets, either at a constant rate or at variable rate. In the later case, the *burst layer* is used to model the burst arrival process. The burst layer is an intermediate traffic layer which operates in the time scale of milli seconds. The lowest layer is the packet layer which operates in the time scale of micro seconds.

The network operates either in on demand or delayed call set up mode. In on demand call set up

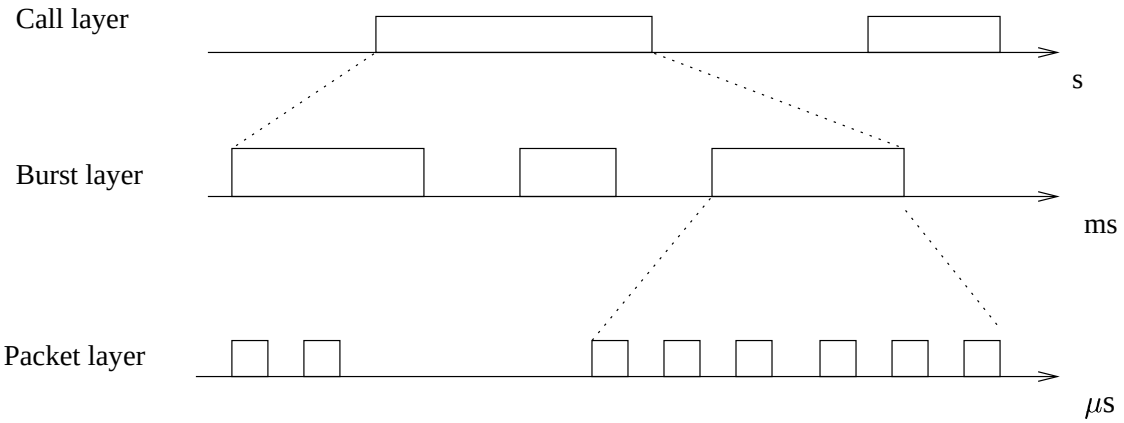


Figure 3.1: Hierarchical traffic model

calls set up blocked calls are cleared while in delayed call set up they are delayed in a queue until sufficient capacity becomes available in the network.

The network performance on the call layer is called Grade of Service (GoS) and consists of the call blocking probability and the call set up delay. The call blocking probability measures the network availability. Telephone networks are usually dimensioned to block (reject) less than 1 % of the call requests.The call set up delay in ATM networks should be in the order of tens of milliseconds in case of on demand call set up.

Network performance on the burst and packet layer is called Quality of Service (QoS) and consists of end-to-end packet loss probability, delay, and delay variability (jitter). The delay can be described by the mean delay or the $(1-\alpha)$ quantile of the delay distribution. The jitter is defined as the difference between the $(1-\alpha)$ quantile of the end-to-end delay and the minimum end-to-end delay.

Network services are divided into guaranteed services and elastic services. Guaranteed services offers deterministic or statistical QoS guarantees. A deterministic guarantee could be zero packet loss and guaranteed packet delivery within 100 ms. Statistical guarantees could be packet loss probability of 10^{-6}, delivery of 98 % of the packets within 100 ms, and jitter less than 50 ms. Elastic services can tolerate relatively large variations in throughput and delay. However, they typically require low packet loss.

Resource management is of utmost importance in multi-service communication networks. Resource management is concerned with the three-way relationship between traffic volume, network resources and realized QoS and GoS. Fixing two of them, the third quantity can be determined. The

goal of resource management is to maintain the QoS and network availability, while using the network resources as efficiently as possible. Then latter behavior is crucial for network operators who want to maximize their revenue in a competitive environment. The resource management algorithms for realizing resource management in B-ISDN and future Internet will not be standardized.

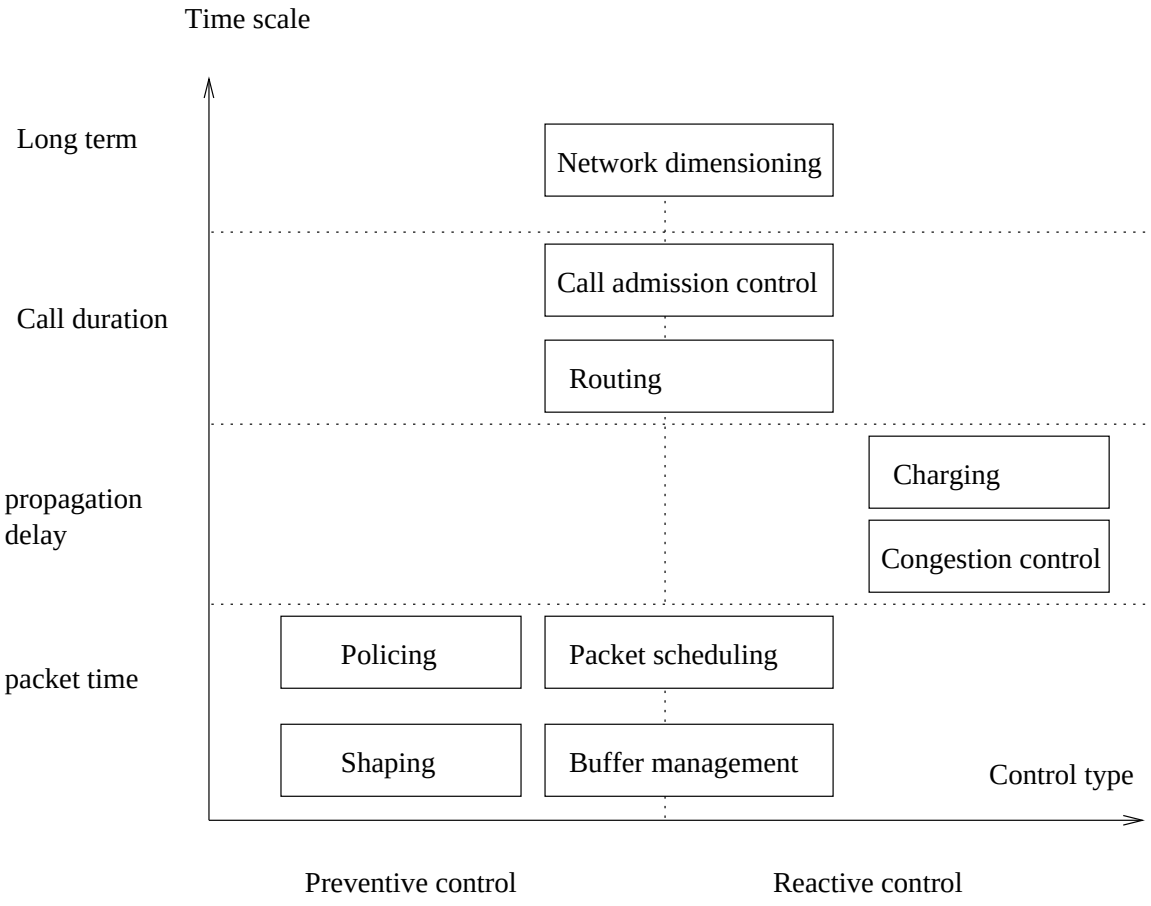


Figure 3.2: Classification of resource management functions

Figure 3.2 shows a classification of the resource management functions studied in this report. Resource management functions are classified into three main categories: preventive resource management, reactive resource management, and combined preventive/reactive resource management functions. Preventive resource management aims at avoiding congestion from occurring too often. Reactive resource management resolves congestion situations occurring during the information transfer phase.

Resource management functions are also classified according to the time scale they operate on. We can distinguish the packet time scale, the propagation delay time scale, the call duration time scale,

and the long term time scale.

The teletraffic engineer designing the various resource management functions faces an accuracy-simplicity dilemma. Accurate QoS and GoS performance models are desired to enable efficient resource allocation decisions, which are neither too pessimistic nor too optimistic. Simple QoS and GoS performance models are necessary to allow practical implementations which meet the timing constraints of the decision making.

3.2 End-to-end traffic model

The network is assumed to consist of a set of switching nodes interconnected by L uni-directional links. The capacity of physical link $s \in S = \{1, \ldots, L\}$ is denoted C^s.

The network is offered traffic from K classes which are assumed to be subject to statistical multiplexing. The j-th class, $j \in J = \{1, \ldots, K\}$, is characterized by the following:

- Origin-destination (OD) node pair,
- Packet traffic descriptor (p_j, m_j, B_j),
- Packet QoS requirements $(P^c_{loss,j}, D^c_j, V^c_j)$,
- Renewal call arrival process with rate λ_j [s^{-1}],
- Generally distributed call holding time with mean $1/\mu_j$ [s],
- Set of alternative routes, W_j,
- Reward parameter $r_j \in (0, \infty)$ for CAC and routing function, and
- Revenue parameter $w_j \in (0, \infty)$ for dimensioning function.

The packet traffic descriptor include:

- Peak bit rate p_j [bits/s],
- Mean bit rate m_j [bits/s],
- Maximum burst size B_j [bits],

The packet QoS requirements include:

- Packet loss probability tolerance $P^c_{loss,j}$,
- Packet delay tolerance D^c_j [s],
- Packet delay variability tolerance V^c_j [s],

The parameter r_j is a CAC and routing control parameter that can be used to achieve several different objectives of the network operator. In particular it can be used to maximize the network revenue if the reward parameters are proportional to the call charging. It can also be used to achieve fairness in network access by increasing the reward parameters for handicapped calls and vice versa.

The parameter w_j is a dimensioning control parameter used to model the revenue generated by a call. Note that the revenue parameter w_j can be different from the reward parameter r_j if the latter is used to control the fairness.

3.3 End-to-end QoS measures

In this section we present formulas for the end-to-end packet loss probability (ratio), end-to-end buffer overflow probability, end-to-end mean delay, end-to-end $(1-\alpha)$ quantile of delay, and end-to-end packet delay variation.

Let ϵ_i denote the packet loss probability or the buffer overflow probability in node i. Assume the end-to-end path consists of N nodes. If we assume the buffer occupancy on successive links to be independent, we can compute the end-to-end packet loss probability or buffer overflow probability as:

$$\epsilon_{path} = 1 - \prod_{i=1}^{N}(1-\epsilon_i) \tag{3.1}$$

The end-to-end mean packet delay, D, is computed as:

$$D = D_{min} + D_{queueing} \tag{3.2}$$

where D_{min} denotes the minimum end-to-end packet delay given by [64]

$$D_{min} = D_{packetization} + D_{propagation} + D_{transmission} + D_{reassembly} + \sum_{i=1}^{N} D_{switching,i} \tag{3.3}$$

and $D_{queueing}$ denotes the sum of queueing delay experienced along the path:

$$D_{queueing} = \sum_{i=1}^{N} \mu_i \tag{3.4}$$

The packetization delay is the time it takes to accumulate enough bits to fill a packet. The propagation delay occurs due to the speed of light in the transmission medium and depends on the distance between the source and destination. Transmission delay depends on the speed (capacity, bandwidth) of the link and becomes negligible as the transmission speeds increases. The reassembly delay occurs occurs when several packets of a frame (e.g. AAL frame) are collected before they are passed to the upper protocol layers. The switching delay is the total time it takes for a packet to traverse the switch/router. The queueing delay are due to the fact the some packets must wait in a queue to resolve switching conflicts.

The switch/router employs either cut-through switching or store-and-forward switching. Cut-through switching means that the switch/router starts to forward bits from the packet as soon as it arrives. Store-and-forward switching means that the router/switch waits for the complete packet to arrive before it starts to forward it. In the first case $D_{transmission}$ is given by the transmission delay at the source. In the second case $D_{transmission}$ is given by the sum of transmission delay at the source and the transmission delay at each store-and-forward switch/router.

Theoretically, the probability density function of the end-to-end packet delay can be obtained by taking the convolution of individual packet delay density functions in the switches/routers. But due to signaling constraints, this is not a feasible method to derive the end-to-end $(1-\alpha)$ quantile of delay. Instead some approximative method can be used.

The end-to-end $(1-\alpha)$ delay is obtained by adding the fixed minimum delay, D_{min}, to the end-to-end $(1-\alpha)$ queueing delay:

$$D(\alpha) = D_{min} + D_{queueing}(\alpha) \tag{3.5}$$

Note that if a playout buffer is used, the playback time point is normally set to $D(\alpha)$.

The effective end-to-end delay D_{eff} experienced by the application user is given by:

$$D_{eff} = D' + D_{coding} + D_{upper-protocols} \tag{3.6}$$

where $D' = D$ or $D' = D(\alpha)$. The coding delay includes any the time to convert a non digital signal to digital bit patterns. For example, an analog signal generated by voice, audio or video source is

sampled and digitized before transmission in the network. The coding delay also includes any delay of the application compression/decompression algorithm (e.g. JPEG, JPEG, MP3). The upper-protocol delay is due to protocol processing at higher layers (e.g. transport layer, application layer).

The end-to-end peak-to-peak packet delay variation (PDV) can also be obtained by some approximative method.

3.4 Traffic policing and traffic shaping

Traffic policing is done at the ingress node of the network. The objective of policing is to assure that the user does not violate the declared traffic parameters negotiated at call set up.

In ATM networks traffic policing is done by the Generic Cell Rate Algorithm (GCRA). For each cell arrival, the GCRA determines whether or not the cell conforms to the traffic contract of the connection [5]. The GCRA is realized as a virtual scheduling algorithm or a continuous-state Leaky Bucket Algorithm. For VBR ATM connections, the peak cell rate is enforced by one GCRA and the maximum burst size and mean cell rate is enforced by another GCRA.

In IP networks traffic policing is done by the token bucket algorithm. Two buckets are used. The first is used to enforce the mean byte rate and maximum burst size, and second is used to enforce the peak byte rate.

Traffic shaping is implemented at the users terminals. This function adapts the user's sending rate to the declared traffic profile. The shaping function can be implemented by a packet buffer and traffic policing unit. The packets wait in the buffer queue until they are admitted by the policing function.

3.5 Call admission control and routing

3.5.1 General considerations

Call Admission Control (CAC) is used for networks which provide QoS guarantees such as circuit-switched networks, ATM networks and QoS enhanced IP networks. CAC is part of preventive resource management which also includes traffic policing. The purpose of CAC is to decide for each new call request, if the call should be accepted or if it must be rejected. The decision is based on three criteria:

- the QoS requirements for network calls,
- the GoS requirements for network call classes,

- the call reward requirements.

CAC is realized by two sub functions: CAC_{QoS} and CAC_{GoS}. The purpose of the CAC_{QoS} function is to decide whether a particular path is expected to offer sufficient QoS to existing calls as well as the new call. If sufficient resources are expected, the call may be accepted on the particular path, otherwise it must be rejected. The purpose of the CAC_{GoS} function is to maintain the GoS and maximize the average reward rate. Different call types will be charged differently and will thus generate a different reward for the network. The CAC_{GoS} function decides if carrying a call on a certain path is economical or not. For example, assume the networks is offered one category of narrow-band calls with a small reward, and one category of wide-band calls with a large reward. In this case, the CAC_{GoS} function may decide to reject narrow-band calls when the path just has free capacity to one more wide-band call.

The CAC function is carried out along with the routing function. The routing function chooses only a path which passes the CAC_{QoS} test. Moreover, it is up to the CAC_{GoS} function to decide whether the path recommended by routing function is consistent with GoS requirements and long-term objectives on average reward.

3.5.2 Design of CAC_{QoS}

The CAC_{QoS} function make use of a model of packet transport performance in the network. A packet network is a complex network of queues. The behavior of a packet stream, e.g. the packet interarrival times, will typically change as the packets cross the network. The reason is that at each switch/router the packets share the output multiplexer with packets from other calls. The mixing of traffic streams can result in clumping and dispersion of packets since packets may have to wait in the queue for their turn to be transmitted. A multiplexer inside the network will be offered traffic streams that have passed zero or more multiplexing steps before arriving to this multiplexer. However, to consider the “sharing effects” experienced by each call is believed to be too complex. Instead, each switch/router in the network is analyzed as if it is offered fresh user traffic which only has passed the policing function. The packet traffic on successive links are assumed to be independent. When a new call is accepted to the network only the traffic on the path of the new call is assumed to be changed.

Switching conflicts can give rise to *packet scale congestion* and *burst scale congestion*. These types of congestion are due to overload on different time scales. Packet scale congestion are due to the random fluctuations on a short time scale due to variable packet interarrival times. The output link can

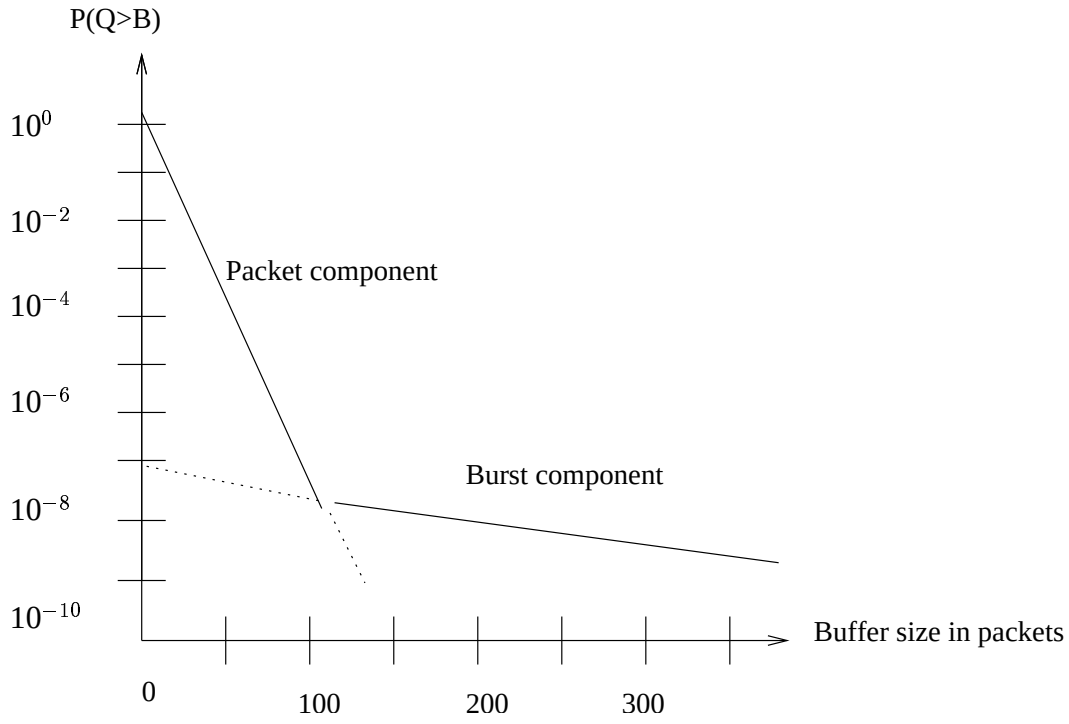


Figure 3.3: Buffer overflow distribution

only transmit one packet at a time, and if many packets simultaneously want access to the same output link, all but one packet must wait in a queue. Burst scale congestion occurs at a larger time scale when the total input rate to the multiplexer exceeds the output capacity for some time period. In this case, the buffer occupancy level will increase until the buffer is saturated or the total input rate decreases. Switching conflicts that arise when the buffer is saturated can not be resolved, forcing arriving packets to be dropped.

The probability of packet loss due to packet scale congestion can be reduced by using a small central or output buffer which can hold in the order of 100 packets per output link. The buffer requirements are proportional to the number of input links. To obtain a low probability of packet loss due to burst scale congestion much larger buffers may be required. The buffer requirement is in this case counted in hundreds of packets. See Figure 3.3 which shows the general characteristics of a buffer overflow distribution.

The fluid flow queueing model captures the behavior of burst scale congestion. Hence, performance measures derived from the fluid flow queueing model will be accurate in the case of large buffers. For smaller buffer sizes, the queue analysis should be performed assuming Markov modu-

lated Poisson process (MMPP) packet arrivals. The MMPP queueing model captures the behavior of both packet and burst scale congestion.

In general, performance evaluation faces an *accuracy-simplicity dilemma*. High accuracy is desired to to admit correct and efficient call admission control. High accuracy enables call admission control which is neither too optimistic nor too pessimistic. Low complexity (high simplicity) is required since call admission control must operate in real time and provide admission decisions in in the order of tens of milliseconds. Hence, the designer of call admission control must make a tradeoff between accuracy and simplicity.

3.5.3 Design of CAC$_{GoS}$

The CAC$_{GoS}$ function decides whether the path recommended by the routing function is consistent with GoS requirements and long-term objectives on average reward.

The CAC function should provide an efficient access control of direct and alternative routed calls to the links. The *trunk reservation* method is used to restrict the access of alternative routed calls when the link load is high. In particular, when the free capacity is less or equal to a threshold t_j the link will not accept alternative routed calls. Another method is called *external blocking*. With this method only a fraction ϕ_j of the calls that are rejected on the direct path are allowed to request a non-direct path. Trunk reservation and external blocking are part of CAC$_{GoS}$. The best overall blocking performance is achieved if trunk reservation and external blocking are combined.

Another issue for the CAC fuction is the distribution of blocking probabilities, or access fairness, among the call classes. Without any special control mechanisms, calls with higher bandwidths demands (i.e. wide-band calls) will suffer higher blocking probabilities than calls with lower bandwidth demands (i.e. narrow-band calls). To level out the blocking probabilities trunk reservation can be used to protect wide-band calls from bandwidth depletion. That is, direct or alternative routed narrow-band calls are rejected when the free capacity is below or equal to a threshold s_j. Trunk reservation to level out the per-class call blocking probabilities is part of CAC$_{GoS}$.

3.5.4 Routing in circuit-switched networks

Fixed routing is the simplest form of routing [43]. In this method calls are only offered to a single path, i.e. no alternative routing is used. If the single path is busy, the call is rejected.

Load sharing routing distributes the calls among the alternative paths according to the load sharing

probabilities [43]. Each call is offered to one path only, and if this path is busy, the call is rejected. The load sharing probabilities may be determined from optimization.

Sticky random routing provides a simple form of alternative routing [32]. The algorithm works as follows. The call is first offered to the direct path. If it is available, the call is established. Otherwise, the call is offered to a previously chosen alternative path. If the alternative path is available, the call is established. Otherwise, the call is blocked and the alternative path is reselected. The new alternative route is used by the next call that find the direct path busy. British Telecom employed in 1996 a sticky random routing algorithm called Dynamic Alternative Routing (DAR) in the British telephone network.

Sequential routing chose paths for new calls according to fixed alternative sequences [33]. The first path in the sequence to be tried is the direct path. If it is busy, the remaining paths are tried, in order, until a path that is able to accept the call is found, or there is not more paths in the sequence in which case the call is rejected. The sequence of paths tried by the routing function is obtained from an optimization procedure. The Dynamic Nonhierachical Routing (DNHR) employed by AT&T in the US telephone network during the 1980s was based on sequential routing.

Least loaded routing (LLR) is the first example of state-dependent alternative routing [4]. The call is first offered to the direct path. If it is busy, an alternative path is searched for according to a state-dependent routing rule. If no such path is found the call is rejected. The state-dependent routing rule selects a path with sufficient capacity that also has the largest free capacity of its bottleneck link. The bottleneck link is the link with least free capacity along the path. Real-Time Network Routing (RTNR) replaced DNHR in the AT&T network in 1991. RTNR is based on LLR routing with real-time state observations.

Markov decision process (MDP) routing is the second example of state-dependent alternative routing [24]. Given a model of the call traffic, MDP theory is able to compute a state-dependent routing rule which achieves *optimal* average reward rate. The routing rule selects the path with has the largest positive long-term reward gain for the new call. If no path has a positive gain, the call is rejected. The idea of MDP routing is simple: control the Markov chain such that it visits high income network states more often than low income network states. The optimal set of decisions can be computed using the Policy Iteration algorithm from Markov decision theory.

3.5.5 Routing in IP networks

Internet consists of a huge number of individual IP subnetworks with unique network addresses (about 210,000 in July 2005). To make the best effort routing in Internet manageable, the IP subnetworks are divided into a set of Autonomous Systems (ASs). An AS is a portion of the network which is under control of a single administrative unit such as a campus. Today (July 2005) Internet has about 16,000 ASs. Special routing protocols are used within the ASs and between the ASs, called intra-domain routing and inter-domain routing protocols, respectively [68].

The intra-domain routing protocols are either based on distance vector routing or on link-state routing. The protocols use some algorithm to compute *shortest path routes* between the source and destination. The context for the shortest path routing is a network with two weight associated with each link – one weight for each direction of the link. The weights measure the “distance “ or cost of crossing the link. The weight could be either packet delay, bandwidth, reliability or some link cost. The distance between two nodes are computed as a sum of link metrics along the path connecting the node pair. Figure 3.4 shows an example network depicted as a graph with links weights.

Inter-domain routing is more focused on finding loop-free paths which can reach the destinations than on finding optimal paths. The ASs use a set of policies to guide the selection of paths.

The future Internet will not only offer best effort service but also QoS service. To meet this goal Internet has to be extended with resource reservation, traffic policing and QoS intra- and inter-domain routing. QoS routing finds paths which meets multiple constraints on path bandwidth, delay, jitter, loss, cost etc.

3.6 Congestion control

The virtual circuit (VC) from the source end-system (SES) to the destination end-system (DES) is composed of sequence of source-destination pairs interconnected by direct links. Congestion control in ATM networks is implemented by an end-to-end rate mechanism. ABR SESs adjust their transmission rates dynamically between a pre-determined MCR and Peak Cell Rate (PCR), based on the amount of network bandwidth left unused by higher priority service categories (CBR, rt-VBR and nrt-VBR). The rate adjustment is done using a closed-loop feedback mechanism, using Resource Management (RM) cells. RM cells convey control information to the SESs about the state of the network, such as congestion state and bandwidth availability.

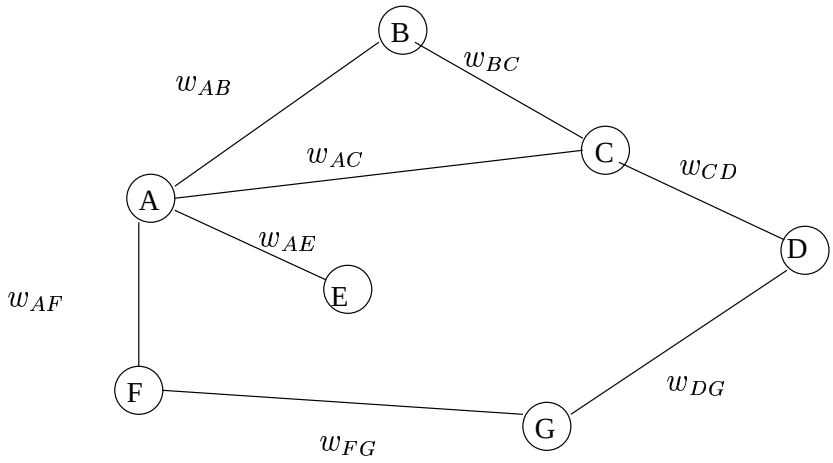


Figure 3.4: Example network graph with link weights. Only one of the two weights for each bi-directional link is shown.

Forward RM (FRM) cells are generated by the SESs and inserted into the outgoing data stream, see Figure 3.5. One FRM cells are sent periodically – one FRM cell after every N_{rm} data cell (e.g. N_{rm}=32). On their way to the DES and back from the DES to the SES, RM cells are processed by the switches. When an RM cell arrives at the DES, the destination changes the direction bit (DIR) in the cell and returns it to the SES. RM cells traveling from DES to the SES are called Backward RM (BRM) cells. BRM cells bring updated network state information to the SESs.

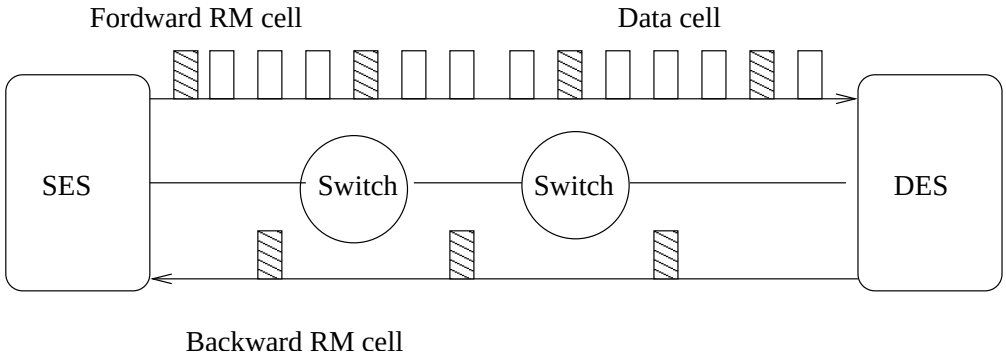


Figure 3.5: ABR rate based flow control

The rate at which an ABR source is allowed to schedule cells for transmission is denoted by ACR (Allowed Cell Rate). The ACR is initially set to the Initial Cell Rate (ICR) and is always bounded between MCR and PCR. The Current Cell Rate (CCR) value in the FRM cell is set to the ACR, and the

Explicit Rate (ER) value is set to the PCR. The SES dynamically adjusts the ACR to the congestion information it receives via the BRM cells. A switch convey information about congestion status or desired rate via RM cells.

An important objective in the allocation of bandwidth to ABR connections is *fairness*. Two main fairness criteria have been considered in the literature: max-min fairness and proportional fairness. The former criteria has been chosen by ATM forum and it is adopted by many ER allocation algorithms.

The max-min allocation is defined as follows [41]. Given a configuration with n contending sources, suppose the i^{th} source is allocated a bandwidth x_i. The allocation vector $\{x_1, x_2, ..., x_n\}$ is feasible if all link load levels are less than or equal to 100 %. Given an allocation vector, the source that is getting the least allocation is, in some sense, the "unhappiest source". We need to find the feasible vectors that gives the maximum allocation to this unhappiest source. Now we remove this "unhappiest source" and reduce the problem to that of the remaining $n - 1$ sources operating on a network with reduced link capacities. Again, we find the unhappiest source among these $n - 1$ sources, give that source the max- minimum allocation and reduce the problem by one source. We repeat this process until all sources have been allocated the maximum they can get.

3.7 Packet scheduling

3.7.1 Buffering strategies

Consider an ATM switch or a IP router with multiple input and output links. Without any special mechanisms, switching conflicts would arise when packets arriving at different input ports simultaneously want access to the same output link. To resolve the switching conflicts buffers are used to queue packets waiting for transmission. The buffers can be placed at three places in a switch/router: at the inputs, at the outputs or centrally [69].

Input queueing is least efficient; it suffers from Head Of Line (HOL) blocking. Lets assume input port i and j has a packet destined for output link p. Further, assume that in queue j there is a second packet destined for output link q for which there is currently no packet waiting in the other input queues. HOL blocking manifests itself by the fact that while the packet first in queue j waits for transmission, it also forces the packets behind it to wait. That is, the packet destined for output link q must wait although there would be no switching conflicts for that packet.

The central queueing solution behaves exactly as the output queueing solution. This means that the mean waiting time is exactly that of the output queueing system. However, since multiples queues are combined in a single physical memory, the major advantage of the central queueing system is reflected in the number of packets to be stored in the central memory (central queue). A more complicated control logic is required to ensure that the single central memory performs the First-In-First-Out (FIFO) discipline to all outlets. The memory size can be computed as the convolution of N individual output queues.

Indeed, since the buffer memory will be shared, a more effective use of the memory can be made. It can be shown that with the size of the central memory and the sum of the output buffer memory sizes being equal, the central queueing yields lower packet loss probability than output queueing.

3.7.2 First-in-First-Out queueing

In the First-In-First-Out (FIFO) queueing discipline packets are served according to the order at which they arrive. That is, the server operates on a First-Come-First-Served (FCFS) basis. FIFO/FCFS provides statistical sharing among calls. Each call receives a statistical (long-term) fraction of the server capacity. On the short term, no guarantees on service are given, since the likelihood of obtaining service depends on the momentary behavior of other sources using the same multiplexer. In particular, if too many sources send at their peak rates the buffer will overflow and packets will have to be dropped. The FCFS discipline is work conserving, which means that the server is never idle when it has work to do.

The FCFS discipline may also change the time spacing between successive packets leaving the switch/router. The obvious reason is that packets may have to wait in the queue before being served. Packets may suffer from clumping or dispersion. In clumping the time distance between two successive packet is reduced, and in dispersion the time distance is increased. The packet clumping/dispersion introduces delay variability which might be critical for real-time services.

To smooth out delay variability, the receiver can use a playout buffer. Packets which arrive at the receiver is placed in buffer which is served in a periodic manner. Packets have time stamps which indicate the time when they were transmitted from the source. The receiver reads out a packet from the buffer a fixed time after the packet was sent from the source. Note that the smoothing ability, i.e. the size of the buffer, is limited by the maximum tolerated delay.

3.7.3 Priority queueing

In (time) Priority Queueing (PQ) separate FIFO queues are used for each call class,which may contain one or more calls. The queues have different priorities which control the order in which queues are served. A low priority queue is only served if there are no packets in the higher priority queues. Starvation of low priority packets is therefore possible if the packet arrival rates of higher priority call classes are too high. However, by proper CAC all calls should be guaranteed a certain statistical, long-term fraction of the server capacity. The PQ discipline is work conserving. Like FIFO, packet streams may suffer from delay variability.

3.7.4 Weighted fair queueing

In Weighted Fair Queueing (WFQ) each call class has its own FIFO queue [66]. Each queue is guaranteed to obtain a certain fraction of the bandwidth (given by the weight of queue) on both the short-term and long-term time scale. When no packets arrive within a call class the excess capacity is shared fairly, according to the weights, among active call classes. The WFQ discipline schedules packets according to the time the packets would have been sent by bit-by-bit round robin. Hence, if all queues contain only one packet each, and all weights are equal, the packets are transmitted in increasing packet size order. When a packet is transmitted the packet behind it becomes first in the queue. This new packet joins the bit-by-bit service emulation and if the packet is short enough it may be transmitted before other longer packets which already were first in their queue when the short packet reached the head of its queue. The WFQ discipline is work conserving. Due to the “isolation” between call classes the packet delay variability under WFQ becomes lower than under FIFO or PQ.

3.8 Buffer management

In the simplest form of buffer management all classes completely share the buffer capacity. On the other hand, space priority queueing assigns buffer space to packets according to their priority. Two examples of such schemes are the Push out scheme and the Partial buffer sharing scheme [64].

3.8.1 Push out scheme

The push out scheme operates only when the buffer is full. When a low priority packet arrives at a full buffer, it is dropped. When a high priority packet arrives at a full buffer and a lower priority packet is

in the buffer, the lower priority packet is “pushed out”, and the arriving high priority packet is added to the buffer. Simulation experiments show that when two priority classes contribute equally to the load, the packet loss for the higher priority class is dramatically reduced while the packet loss for the lower priority class increases slightly.

3.8.2 Partial buffer sharing scheme

The partial buffer sharing scheme reserves a fixed number of buffer spaces for high priority packets. Packets from all classes share the buffer space until the buffer content reaches a threshold. If the number of packets in the buffer exceeds the threshold then only higher priority packets will be accepted (as long as there is space). Simulation results indicate that partial buffer sharing performs only slightly worse than the push out approach.

3.9 Dimensioning in circuit-switched networks

3.9.1 General considerations

In this section we describe dimensioning in circuit-switched networks. The algorithms also apply to packet-switched networks with virtual circuits such as ATM networks. Even QoS enhanced IP networks which use the concept of flows, which are the equivalents of ATM virtual circuits, may use the dimensioning algorithms designed for circuit-switched networks.

A set of virtual networks can be overlayed on the physical network to simplify resource management. A given virtual network is typically targeted at a specific QoS class. For example, in ATM networks, separate virtual networks can be maintained for CBR, VBR, ABR and UBR services, respectively. The topology of the virtual networks may be different from the physical network topology. Each virtual network link may be carried by one or several physical links. Each physical link is in general shared between multiple virtual networks. The bandwidth allocation at the call layer may be based on complete sharing (CS), complete partitioning (CP) or partial sharing (PS). The bandwidth allocation at the packet layer may be based on deterministic multiplexing or statistical multiplexing.

Resource allocation to virtual and physical networks is referred to as virtual/physical link capacity dimensioning [27, 43]. The first step in network dimensioning is the design of the topological structure of the network, i.e. where to place the nodes and how to interconnect them. In most cases the location of the nodes is given for political or historical reasons. So only the structure of the intercon-

nection graph has to be determined. This can be done through methods of topological optimization and graph theory. By performing this step, connectivity and reliability constraints and link costs have to be regarded. The link cost information is simply a fixed node interconnection cost per unit length, depending on the used technology. For ATM in most cases the result of the topological design phase will lead to a partly meshed backbone network structure. The second step of network dimensioning determines, given the network topology, traffic demand and GoS requirements, the capacities (bandwidths) of the physical and virtual network links.

3.9.2 Problem formulation

The objective used in dimensioning of the virtual and physical networks can be of several types. Two common examples are maximization of the average revenue and minimization of total network link cost. The first type objective of objective function can in case of physical network dimensioning be written as:

$$W(\mathbf{C}, \Upsilon) = \sum_{j \in J} \lambda_j w_j [1 - B_j(\mathbf{C}, \Upsilon)] \tag{3.7}$$

where $\mathbf{C} = (C^1, \ldots, C^L)$ denotes the vector of physical link capacities and Υ denotes a set of CAC and routing parameters. For example, in MDP routing Υ refers to the set of reward parameters, and in LLR routing Υ refers to the set of trunk reservation parameters or external blocking factors. Note that although the CAC and routing strategy is specified, its optimal parameters may be a function of network flow distributions and link dimensions which are not known in advance. Thus, in general, the CAC and routing parameters, Υ, should also be treated as optimization variables.

The second type of objective function can in case of physical network dimensioning be written as:

$$c(\mathbf{C}, \Upsilon) = \sum_{s \in S} c_s(C^s, \Upsilon) \tag{3.8}$$

where c_s denotes a cost function for link s.

The objective function should be optimized subject to a constraint. An absolute GoS constraint specifies that the absolute call blocking probability for a given call category and OD pair should be less than or equal to a given value:

$$B_j(\mathbf{C}, \Upsilon) \leq B_j^c \tag{3.9}$$

3.9.3 Grade of Service models

In general the network dimensioning procedures require network performance models since the network design should meet the call blocking probability constraints. The fixed point equations is based on statistical link independence resulting in decomposition of the network model into a set of link loading functions, f_l, and link performance functions, f_p:

$$\mathbf{A}^s = f_l(\mathbf{\Pi}, \pi) \tag{3.10}$$

$$\Pi^s = f_p(\mathbf{A}^s, C^s) \tag{3.11}$$

where $\mathbf{A}^s = [A_j^s, j \in J]$ denotes the offered traffic to link s, Π^s denotes the set of required performance characteristics of link s (e.g. state probabilities or blocking state probabilities), π denotes the routing policy, and C^s denote the link capacity of link s. The fixed point equations can be solved using repeated substitutions.

3.9.4 Optimization framework

The problem of finding vector of link capacities where $\mathbf{C} = (C^1, \ldots, C^L)$ that optimizes the non-linear objective function under the given set of non-linear constraints can be solved using Sequential Quadratic Programming (SQP).

The basic idea of SQP is to model the nonlinear programming problem (NLP) at a given approximate solution, say x^k, by a quadratic programming subproblem, and then to use the solution to the subproblem to construct a better approximation x^{k+1}. This process is iterated to create a sequence of approximations that, it is hoped for, will converge to the solution x^*. The nonlinear constraints are replaced by a linear first order Taylor series approximation and the nonlinear objective is replaced by a second order Taylor series approximation augmented by second order information from the constraints. Perhaps the key to understanding the performance and theory of SQP is that fact that, with an appropriate choice of quadratic subproblem, the method can be viewed as the natural extension of Newton and quasi-Newton methods to the constrained optimization setting.

3.10 Charging and pricing

Charging, pricing, accounting and billing are crucial features of telecommunication services. How should the network provider design tariffs for the range of services offered? This is partly a marketing decision – tariffs must be attractive to customers – but network providers are also concerned with efficiency and cost-recovery. Charging schemes should encourage efficient use of the network and should generate revenue in a fair way according to the relative usage of customers.

In multi-service networks, tariffs might depend on a number of parameters defining the traffic and QoS characteristics of a call, in order that charges should reflect network resource usage. The way that a customer uses the network depends on the tariffs and also on how the customers values each type of call (the customer's *utility*, in the language of economics). The interplay between tariffs, network resource usage, and customer incentives is a fertile area for economic and mathematical models.

Multi-service networks need to include facilities for charging, pricing, accounting, and billing [74]. In this context, *charging* designates the evaluation of costs for the call. The cost is calculated based on some characteristics of the call, according to a *charging scheme*, which in turn is part of a tariffing policy. Pricing is the process of assigning a price (expressed in monetary units) to a specific service. This process combine technical considerations, such as the amount of resources used for a service, and economical considerations, such as applying tariffing theory and marketing methods. *Accounting* involves gathering information necessary so that the total charge can be itemized against tariffs and usage measurements. *Billing* involves collecting charge information over a given period and communicating this to the customer in the form of a bill. Another important concept is *advise of charge*, where a customer can be given on request the charge for a specific call (whether intended, ongoing or just completed).

A tariffing policy may include both call charges and subscription charges. We will say that charging is *usage-based* when the call charges are included. Usage-based charges may also include subscription charges that are not related to usage.

3.10.1 Multi-service network costs

The cost of multi-service networks consists of the following [74]:

- The incremental cost of send an extra packet. In the absence of congestion this is essentially zero;

- The congestion costs, or social costs of delaying other users' packets. As above, this cost is zero when there is no congestion;
- The fixed costs of the network infrastructure (e.g. routers, communication lines, maintenance, and management;
- The incremental costs of connection to the network. This involves the cost of the access lines and customer premises equipment needed to connect to the network. This cost represent the largest portion of the total cost for an organization to connect to the multi-service network;
- The cost of expanding the capacity of the network.

Fixed costs constitute the major percentage of the total costs. On the other hand, the marginal or incremental costs are non-zero only in the presence of network congestion. The observation leads to the proposition than under normal (i.e. uncongested) operation, the network charges should include only fixed charges. However, in the presence of congestion, charges should also include a non-zero usage charge which depends on the level of congestion and the magnitude of the users' contribution to it. This effect of congestion on prices can be expressed either with dynamically adjusted prices or time-of-day sensitive prices.

Chapter 4

Traffic modeling

4.1 Definition of traffic processes

Simple traffic consist of single arrivals of discrete entities (call or connection requests, packets etc.) [30]. It can be mathematically described a point process, consisting of a sequence of arrival instants $\tau_1, \tau_2, ..., \tau_n, ...$ measured from the origin 0; by convention, $\tau_0 = 0$. There are two additional equivalent descriptions of point processes: counting processes and inter-arrival time processes. A counting process $\{N(t)\}_{t=0}^{\infty}$ is a continuous-time, non-negative integer-valued stochastic process, where $N(t) = \text{Max}\{n : \tau_n \leq t\}$ is the number of (traffic) arrivals in the interval $(0, t]$. An inter-arrival time process is a non-negative random sequence $\{t_n\}_{n=1}^{\infty}$, where $t_n = \tau_n - \tau_{n-1}$ is the length of the time interval separating the n-th arrival from the previous one. The equivalence of these descriptions follows from the equality of events:

$$\{N(t) = n\} = \{\tau_n \leq t \leq \tau_{n+1}\} = \{\sum_{k=1}^{n} t_k \leq t < \sum_{k=1}^{n+1} t_k\} \tag{4.1}$$

since $\tau_n = \sum_{k=1}^{n} t_k$. Unless otherwise stated, we assume that $\{t_n\}$ is a stationary sequence and that the common variance of t_n is finite.

4.2 Second-order statistics

Second-order statistics express relations between traffic occurencies at two time instants. The following are second order properties of a stochastic process $\{X_n\}$ in discrete time [59].

- *Autocovariance C(k)*:

$$C(k) = E[(X_n - \overline{X})(X_{n+k} - \overline{X})] = E[X_n X_{n+k}] - \overline{X}^2 \tag{4.2}$$

- *Autocorrelation R(k)*:

$$R(k) = E[X_n X_{n+k}] \tag{4.3}$$

- *Autocorrelation coefficient* $r(k)$:

$$r(k) = C(k)/Var[X_k] \tag{4.4}$$

- *Power spectrum* $\Phi(\omega)$:

$$\Phi(\omega) = \mathcal{F}\{R(k)\} = \sum_{k=-\infty}^{\infty} R(k) e^{-i\omega k} \tag{4.5}$$

- *Autocovariance spectrum* $S(\omega)$:

$$S(\omega) = \mathcal{F}\{C(k)\} = \sum_{k=-\infty}^{\infty} C(k) e^{-i\omega k} \tag{4.6}$$

The autocovariance spectrum $S(\omega)$ can be expressed as

$$S(\omega) = lim_{M\to\infty} E\left\{ \frac{1}{2M+1} \left| \sum_{n=-M}^{M} x(n) e^{-i\omega n} \right|^2 \right\} \tag{4.7}$$

The squared $\mathcal{F}$-transform of $x(n)$ (divided by the record length) may be used as an estimator of the autocovariance spectrum. As such, it is known as the *periodogram* $\hat{S}_{PER}(\omega)$:

$$\hat{S}_{PER}(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-i\omega n}\right|^2 \tag{4.8}$$

4.3 Traffic burstiness

A recurrent theme relating to traffic in multi-service networks is the traffic "burstiness" exhibited by key services such as compressed video, file transfer etc. Burtiness is present in a traffic process if the arrival points $\{\tau_n\}$ appear to form visual clusters; that is, $\{t_n\}$ tends to give rise to runs of several relatively short interarrival times followed by a relatively long one. The mathematical underpinning of burstiness is more complex. Two main sources of burstiness are due to the shapes of the marginal distribution and the autocorrelation function of $\{t_n\}$. For example, burstiness would be facilitated by a bimodal marginal distribution of $\{t_n\}$, or by short-term autocorrelation in $\{t_n\}$. Strong positive autocorrelation are a particularly major cause of burstiness. Since there seems to be no single widely-accepted notion of burstiness, we briefly describe some of the commonly-used mathematical measures that attempt to capture it [30].

The two simples measures of burstiness take account only of first-order properties of traffic (they are each a function of the marginal distribution only of interarrival times). The first one is the ratio of the peak rate to the mean rate – a very crude measure, which also has the shortcoming of dependence on the interarrival length utilized for rate measurement. A more elaborate measure of burstiness is the coefficient of variation, defined as the ratio of the standard deviation to mean $c_A = \sigma[t_n]/E[t_n]$ of interarrival times.

In contrast, the peakedness measure and the index-of-dispersion measures do take account of temporal dependence in traffic (second-order properties). For a given time interval of length τ, the index of dispersion for counts (IDC) is the function $I_c(\tau) = Var[N(\tau)]/E[N(\tau)]$; i.e., the variance-to-mean ratio of the number of arrivals in the interval $[0, \tau]$. Since the number of arrivals is related to the sum of interarrival intervals via Equation (4.1), the numerator of the IDC includes the autocorrelation of $\{t_n\}$. The index of dispersion for intervals (IDI) is defined as

$$J_k = \frac{Var\left[\sum_{n=1}^{k} t_{i+n}\right]}{kE^2[\tilde{t}]} \tag{4.9}$$

where $\{t_i\}$ is a stationary sequence of interarrival times.

The peakedness concept is related, but more involved. Assume that the traffic stream $\{t_n\}$ is

offered to an infinite server group consisting of independent servers with common service times distribution F. Let S be the equilibrium number of busy servers. The peakedness is the functional $z_A[F] = Var[S]/E[S]$, which maps a service time distribution to a real number. A commonly used peakedness is $z_{exp}(0)$, obtained as a limiting case for an exponential service distribution with service rate approaching 0.

Finally, the Hurst parameter H can be used as measure of burstiness via the concept of self-similarity.

4.4 Renewal models

In a renewal process, the t_n are independent, identically distributed (IID), but their distribution is allowed to be general. Unfortunately, with a few exceptions, the superposition of independent renewal processes does not yield a renewal process. The ones that do however, occupy a special position in traffic theory and practice. Queueing models historically have continually assumed renewal-offered traffic.

Renewal processes, while simple analytically, have a severe modeling drawback – the autocorrelation function of $\{t_n\}$ vanishes for all nonzero lags. The importance of capturing autocorrelations stems from the role of the autocorrelation function as a statistical proxy for temporal dependence in time series. Since burstiness can be explained to a large extent by positive autocorrelation in $\{t_n\}$, renewal processes are not used to model packet arrival processes. However, for call arrival processes, the renewal model is believed to be adequate.

The *Poisson model* is the oldest traffic model, dating back to the advent of telephony and the renowned pioneering telephone engineer A.K. Erlang. A Poisson process can be characterized as a renewal process whose interarrival times $\{t_n\}$ are exponentially distributed with rate parameter λ: $\mathrm{P}(t_n \leq t) = 1 - \exp(-\lambda t)$. Equivalently, it is a counting process, satisfying $\mathrm{P}(N(t) = n) = \exp(-\lambda t)(\lambda t)^n/n!$, and the number of arrivals in disjoint intervals is statistically independent (a property known as independent increments).

Poisson processes enjoy some elegant analytical properties. First, the superposition of independent Poisson processes results in new Poisson process whose rate is the sum of the component rates. Second, the independent increment property renders Poisson a memoryless property. This, in turn, greatly simplifies queueing problems involving Poisson arrivals. Third, Poisson process are fairly common

in traffic applications that physically comprise a large number of independent traffic streams, each of which can be quite general. The theoretical basis for this phenomenon is known as Palm's Theorem. It roughly states that under suitable but mild regularity conditions, such multiplexed streams approach a Poisson process as the number of streams grows, but the individual rates decrease so as to keep the aggregate rate constant.

4.5 Markov-based models

A.A. Markov and A. Kolmogorov laid the foundation to the theory of Markov processes in the early 20th century. A discrete-state stochastic process $\{X(t)|t \geq 0\}$ is called *Markov chain* if for $t_0 < t_1 < t_2 < \ldots < t_n < t$, the state transition probability function satisfies the following Markov property:

$$P[X(t) = x|X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_0) = x_0) = P[X(t) = x|X(t_n) = x_n] \quad (4.10)$$

That is, the current state summarizes all relevant information about past states. Unlike renewal traffic models, Markov traffic models introduce dependence into the random sequence of interarrival times [30]. Consequently, they can potentially capture traffic burstiness, due to non-zero autocorrelations in the interarrival time process. We may interpret transitions of the Markov chain as signalling arrivals.

4.5.1 Discrete-time Markov chains

A discrete-time Markov chain (DTMC) is characterized by state changes that can take place only at discrete times. The DTMC has geometrically distributed state sojourn times.

A DTMC can be characterized by the properties if its states. Let $f_{ij}^{(n)}$ denote the probability that the first return to state E_j occurs at the nth step. The probability of *at least* one return to state E_j is given by

$$f_{jj} = \sum_{n=1}^{\infty} f_{jj}^{(n)} \quad (4.11)$$

Thus the system is certain to return to E_j if $f_{jj} = 1$. In this case μ_{jj} defines the mean return (recurrence) time:

$$\mu_{jj} = \sum_{n=1}^{\infty} n f_{jj}^{(n)} \quad (4.12)$$

The states of a discrete-time Markov chain can be classified based on the definition of the first return times as follows:

1. A state is *transient* if $f_{jj} < 1$,
2. A state is *recurrent* (persistent) if $f_{jj} = 1$,
3. A recurrent state is *null* if $\mu_{jj} = \infty$ and *nonnull* if $\mu_{jj} < \infty$,
4. A state is *periodic* with period t if a return only is possible in $t, 2t, 3t, \ldots$ steps,
5. A recurrent state is *ergodic* if it is nonnull and aperiodic.

An *irreducible* DTMC is one which all states are reachable from all other states. The states of an irreducible DTMC are either all transient or all recurrent nonnull or all recurrent null. An *ergodic* DTMC is irreducible, recurrent nonnull, and aperiodic. Most of the systems in which we are interested are modeled by ergodic DTMC chains because this corresponds to well-defined steady state behavior. If the transition probabilities are independent of the time index n the DTMC is *homogeneous*.

In a DTMC, a set C of states is said to be *closed* if the system, once it is in one of the states of C will remain there indefinitely. A special example of a closed set is a single state E_j with transition probability $p_{jj} = 1$. In this case, E_j is called *absorbing state*.

Theorem

Let $\pi_j^{(n)} \triangleq P[X_n = j]$ be the probability of finding the system in state E_j at the nth step. In an irreducible and aperiodic homogeneous Markov chain the limiting probabilities

$$\pi_j = \lim_{n \to \infty} \pi_j^{(n)} \tag{4.13}$$

always exists.

Moreover, either:

1. all states are transient or all states are recurrent null in which case $\pi_j = 0$ for all j and there exists no stationary distribution, or
2. all states are recurrent nonnull and then $\pi_j > 0$ for all j, in which case the set $\{\pi_j\}_{j \in S}$ is a stationary probability distribution which are uniquely determined from the following linear equation system:

$$\begin{cases} \pi = \pi \mathbf{P} \\ \sum_{i \in S} \pi_i = 1 \end{cases} \tag{4.14}$$

where $\pi = (\pi_1, \ldots, \pi_N)$ and $\mathbf{P}$ denotes the state transition probability matrix $\mathbf{P} = (p_{ij})$ and p_{ij} denotes the transition probability between state i and state j.

4.5.2 Continuous-time Markov chains

A continuous-time Markov chain (CTMC) is characterized by state changes that can occur at arbitrary points in time. A CTMC can be completely described by an initial state probability vector for $X(t_0)$ and transition probability functions. The CTMC has exponentially distributed state sojourn times. A CTMC is said to be *irreducible* if every state can be reached from every other state, with non-zero probability. A state is said to be absorbing if no other state can be reached from it with non-zero probability. Notion of *transient, recurrent nonnull, recurrent null* carry over from DTMCs. There is no notion of periodicity for a CTMC, however. An *irreducible* CTMC is *ergodic*.

If the CTMC is ergodic the equilibrium probability distribution $\{\pi_i\}_{i \in S}$ can be obtained from a set of linear equations subject to the probability conservation constraint:

$$\begin{cases} \pi \mathbf{Q} = 0, \\ \sum_{i \in S} \pi_i = 1 \end{cases} \tag{4.15}$$

where $\pi = (\pi_1, \ldots, \pi_N)$ and $\mathbf{Q}$ denotes the infinitesimal generator matrix $\mathbf{Q} = (q_{ij})$ and q_{ij} denotes the transition rate between state i and state j. The diagonal elements q_{ii} are determined such that the sum of all elements in each row is zero.

4.5.3 Markov renewal process

Markov renewal processes (MRPs) generalize renewal processes and Markovian arrival processes (MAPs). Markov-renewal models are more general than discrete-state Markov processes, yet retain a measure of simplicity and analytical tractability. A Markov renewal process $R = \{(X_n, \tau_n)\}_{n=0}^{\infty}$ is defined by a Markov chain $\{X_n\}$ and its jump times $\{\tau_n\}$, subject to the following constraint: The distribution of the pair (X_{n+1}, τ_{n+1}), of next state and inter-jump time, depends only on the current state X_n, but not on previous states nor on previous inter-jump times. Again, if we would interpret

transitions of $\{X_n\}$ as signalling arrivals, we would have dependence in the arrival process. Also, unlike the DTMC and CTMC case, the interarrival times can be arbitrarily distributed, and these distributions depend on the state of the Markov process.

4.5.4 Semi-Markov process

The semi-Markov process (SMP) is a continuous-time discrete-state Markov process with generally distributed state sojourn times. SMPs generalize DTMCs and CTMCs. The SMP may be shown to be equivalent to the family of MRP [19].

4.6 Markov-modulated processes

Markov-modulated models constitute an important class of stochastic models.

Let $\{X(t)\}_{t=0}^{\infty}$ be a continuous time Markov process, with state space of $1, 2, ..., m$ (more complicated state spaces are possible). Now assume that while X is in state k, the probability law of customer arrivals is completely determined by k, and this holds for every $1 \leq k \leq m$. Note that when X undergoes transition to, say, state j, then a new probability law of arrivals is modulated by the state of X.

4.7 Markov-modulated Poisson processes

The most commonly used Markov-modulated model is the MMPP (Markov-Modulated Poisson Process) model, which combines the simplicity of the modulating (Markov) process with that of the modulated (Poisson) process [30]. In this case, the modulation mechanism simply stipulates that in state k of X, arrivals occur according to Poisson process at rate λ_k. As the state changes, so does the rate.

MMPP models can be used in a number of ways. Consider first a single traffic source with variable state. A simple traffic model would quantize the rate into a finite number of rates and each rate would give rise to a state in some Markov modulating process. Certainly, it remains to verify that exponential holding times are an appropriate description, but the Markov transition matrix $Q = [Q_{ij}]$ of the putative X can be easily estimated from empirical data. Simply quantize the empirical data, and the estimate Q_{kj} by calculating the fraction of states that X switched from state k to state j.

As an example, consider a two-state MMPP model, where one state is an "on" state with an associated positive Poisson rate, and the other state is an "off" state with associated rate zero (such models are known as interrupted Poisson for obvious reasons). These models have been widely used to model voice sources; the "on" state corresponds to a talk spurt (when the speaker emits a sound), and the "off" state corresponds to a silence (when the speaker takes a break).

4.8 Markov modulated fluid process

The fluid paradigm dispenses with individual arriving units. Instead, it views arriving units as a stream of fluid, characterized by a flow rate (e.g. bits per second), so that a arrival count is replaced by a arrival volume [30].

Fluid models are appropriate to cases where individual units are numerous relative to a chosen time scale. Put differently, an individual unit is by itself of vanishingly little significance, just as one molecule more or less in a water pipeline has but an infinitesimal effect on the flow.

Typical fluid models assume the sources are bursty – of the "on-off" type. While in the "off" state, traffic is switched off, whereas in the "on" state traffic arrives deterministically at constant rate λ. For analytical tractability, the durations of the "on" and "off" periods are assumed to be exponentially distributed and mutually independent.

4.9 Characteristics of self-similar processes

4.9.1 Discrete-time self-similar process

The following definition of self-similarity for discrete-time stochastic processes is widely adopted. Assume X_k to be a discrete-time wide-sense stationary process with mean $E[X_k] = \overline{X}$ and autocorrelation function $r(k)$.

Consider next the process $X_k^{(m)}(m = 1, 2, ...)$ that are constructed out of X_k as $X_k^{(m)} = \frac{1}{m}\sum_{n=0}^{m-1} X_{km+n}$, i.e. by averaging over non overlapping blocks of size m. The process $X_k^{(m)}$ are also wide-sense stationary, with mean $\overline{X}$ and autocorrelation function $r^{(m)}(k)$. The process X_k is called *exactly self-similar*, with self-similarity parameter H, $(0 < H < 1)$, if it satisfies

$$X_k \stackrel{d}{=} m^{1-H} X_k^{(m)} \tag{4.16}$$

That is, the processes X_k and $m^{1-H}X_k^{(m)}$ should have the same finite-dimensional distributions for all aggregation levels m.

The discrete-time stochastic process X_k is called *asymptotically second-order self-similar* if it satisfies

$$r^{(m)}(k) = r(k) \text{ for } m, k \to \infty. \tag{4.17}$$

In teletraffic modeling the discrete-time stochastic process X_k represents the number of traffic entity (packets, connections/calls) arrivals at time k.

4.9.2 Continuous-time self-similar process

In case we have a continuous-time stochastic process, $X(t)$, the following definition is widely adopted. The continuous-time stochastic process $\{X(t)\}$ is *strictly self-similar* with self-similarity parameter H, $(0 < H < 1)$, if for any positive stretching factor m, the following condition holds:

$$X(t) \stackrel{d}{=} m^{-H}X(mt) \tag{4.18}$$

That is, the process $m^{-H}X(mt)$, is equal in distribution to the original process $\{X(t)\}$ for all stretching factors m.

In teletraffic modeling the continuous-time stochastic process $X(t)$ represents the number of traffic entity (packets, connections/calls) arrivals in the time interval $[0, t]$.

4.9.3 Manifestation of self-similarity

Self-similarity manifests itself in a variety of ways:

- slowly (hyperbolically) decaying variances: *Var*$[X_k^{(m)}] \sim m^{-\beta}$ if $m \to \infty$, whereby $0 < \beta < 1$;
- a slowly decaying autocorrelation function: $r(k) \sim k^{-\beta}$ if $k \to \infty$;
- a power spectral density $S(f)$ that behaves like that of $1/f$ noise around the origin: $S(f) \sim f^{-(1-\beta)}$ if $f \to 0$

Self-similarity also implies *long-range dependence* (LRD), i.e., $\sum_{k=-\infty}^{+\infty} r(k) = \infty$. A process for which $\sum_{k=-\infty}^{+\infty} r(k) < \infty$ is said to be *short-range dependent* (SRD). Such a process differs from a LRD process in the sense that

- the variances *Var*$[X_k^{(m)}]$ decay as m^{-1};
- $r(k)$ decays exponentially fast: $r(k) \sim \sum_{n=1}^{p} c_n \rho_n^k$ for large k;
- the power spectral density remains finite (and approximately constant) around the origin;
- the process behaves like (second-order) pure noise for large m.

An important parameter of a LRD process is the so-called self-similarity of Hurst parameter $H = 1 - \beta/2$. It is named after H. Hurst, who observed the following fact. Given a set of experimental data a_k ($k = 1, ..., n$), with sample mean $\overline{a}(n) = \frac{1}{n}\sum_{k=1}^{n} a_k$ and sample variance $S^2 = \frac{1}{n-1}(\sum_{k=1}^{n}[a_k - \overline{a}(n)])$, define the rescaled adjusted range (R/S) statistic as

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)}[\text{Max}(0, W_1, W_2, ..., W_n) - \text{Min}(0, W_1, W_2, ..., W_n)] \tag{4.19}$$

whereby $W_k = (a_1 + a_2 + ... + a_k) - k\overline{a}(n)$. The quantities W_k measure the deviation of the process a_k from its "expected value". $R(n)$ then measures the "record" values of this deviation. For many "naturally" occurring processes, one has $E[R(n)/S(n)] \sim n^H$ when $n \to \infty$ with H "typically" around 0.7. If a_k is short-range- dependent process, i.e. with a correlation structure over only small time scales, one would have $H = 0.5$. The larger experimental values can be explained by assuming that a_k is self-similar.

Some "tools" to asses the self-similar nature of a given trace are:

- the R/S plot – plotting $\log[R(n)/S(n)]$ versus $\log(n)$ for various subsets of the available data allows one to determine, by linear regression, the Hurst parameter H;
- the variance-time plot – plotting log *Var*$[X_k^{(m)}]$ versus $\log(m)$ allows one to estimate $\beta = 2(1 - H)$;
- the spectral density, which can be estimated by means of the periodogram explained previously – plotting $\log[S(f)]$ versus $\log(f)$ allows an estimate for $1 - \beta$.

In general, self-similarity and LRD are not equivalent. As an example, the increments of a standard Brownian motion are self-similar with Hurst parameter $H = 1/2$, but clearly not LRD (the increments are independent). However, under the restriction $1/2 < H < 1$, LRD is equivalent to asymptotic second-order self-similarity for stationary processes.

4.10 Self-similar traffic models

Recent studies of high-quality, high-resolution traffic measurements have revealed a new phenomenon with potentially important ramifications to the modeling, design, and control of multi-service networks. These include an analysis of hundreds of millions observed packets over an Ethernet LAN in a R & D environment at Bellcore [54], an analysis of few millions of observed frame data generated by VBR video services [8], and analysis of tens of thousands of TCP connection arrivals on the Internet [28]. In these studies, packet and connection traffic appears to be statistically self-similar.

A self-similar (or fractal) phenomenon exhibits structural similarities across a wide range of time scales. In the case of packet traffic, self-similarity is manifested in the absence of natural length of a burst: at every time scale ranging from a few milliseconds to minutes to hours, similar-looking traffic bursts are evident [59].

Models of self-similar processes are needed for simulation and performance analysis. Three important examples of self-similar traffic models are:

- superposition of many renewal arrival process with heavy-tailed interarrival time distributions,
- superposition of arrival processes from many ON/OFF sources with heavy-tailed ON and/or OFF period durations,
- fractional Brownian motion (FBM).

FBM is defined as a stochastic process Z_t with the following properties [63]:

- Z_t has stationary increments;
- $Z_0 = 0$ and $E[Z_t] = 0$ for all $t \geq 0$;
- $Var(Z_t) = |t|^{2H}$ for all $t \geq 0$
- Z_t has a Gaussian distribution for $t > 0$.

From the first three properties it follows that the covariance function is given by:

$$\rho(s,t) = E[Z_s Z_t] = \frac{1}{2}\{t^{2H} + s^{2H} - (t-s)^{2H}\} \tag{4.20}$$

for $0 < s \leq t$. For Gaussian processes, the mean and variance structure determine the finite-dimensional distributions uniquely. Therefore we conclude from (4.20) that $\{Z_t : 0 \leq t < \infty\}$ and $\{m^{-H} Z_t(mt) : 0 \leq t < \infty\}$ has the same finite-dimensional distributions. That is, fractional Brownian motion is strictly self-similar with Hurst parameter H. In fact, FBM is the only Gaussian process with stationary increments that is self-similar.

Norros defines *fractional Brownian traffic* as

$$A_t = Mt + \sqrt{aM} Z_t, t \geq 0, \tag{4.21}$$

where M is the mean rate, a is a variance coefficient and Z_t a FBM. Here, A_t represents the number of traffic entities (e.g. data packets) that is offered in the time interval $[0,t]$. Norros motivates the scaling factor $\sqrt{M}$ in (4.21) by the superposition property: the sum $A_t = \sum_{i=1}^{K} A_t^{(i)}$ of K independent FBM traffics with common parameters a and H but individual mean rates M_i can be written $A(t) = M + \sqrt{aM} Z_t$, where $M = \sum_{i=1}^{K} M_i$ and Z_t is a FBM with parameter H. This shows that the roles of the three parameters of the traffic model (4.21) can be separated so that H and a characterize the "quality" of the traffic in contrast to the long run mean rate M which characterizes the "quantity" alone.

Kaj showed in [45] that the superposition of independent renewal counting processes associated with a heavy-tailed interarrival time distribution converges weakly after rescaling in time and space to FBM, as the number of renewal processes tend to infinity.

Taqqu, Willinger and Sherman showed in [76] that the superposition of many ON/OFF sources whose ON periods and OFF periods exhibit the Noah effect (i.e. have high variability or infinite variance) produces aggregate network traffic that features the Joseph effect (i.e. is self-similar or long-range dependent). The superposition converges after scaling to FBM, as first the number of users n tend to infinity and then, in a second step, a time rescaling parameter tends to infinity.

Taqqu, Teverovsky, and Willinger conclude that normal mono-fractal models seem to suffice in a LAN setting, but the WAN environment may require more complex structures such as multi-fractals.

A non-negative process $X(t)$ is called *multi-fractal* if the logarithm of the absolute moments scale linearly with the logarithm of the time scale m.

4.11 Modeling of call traffic for real applications

The network is offered traffic from K call classes. Each call class is associated with one of P origin-destination (OD) node pairs. Each OD pair is offered traffic from G call categories, meaning that $K = PG$.

4.11.1 OD pair call arrival process

Since the days of Erlang the Poisson model has commonly been used to describe the random arrivals of call requests to the OD pairs of a telephone network. Although the Poisson model serves its purpose in telephone networks, it lacks descriptive power in the case of Internet where a substantial portion of traffic is World Wide Web (WWW), network news (NNTP) and email (SMTP) connections transported by TCP. The WWW, NNTP and SMTP services produce connection arrivals which are different in nature from the telephone service; For example, a person using the WWW service is more likely to initiate additional downloads after the first download. A person using the telephone service is more likely to initiate independent calls.

Measurements on real WWW, NNTP and SMTP connection arrivals in the Internet have revealed that the arrival process shows burstiness over many time scales, ranging from seconds to hours. Paxson and Floyd found that actual WAN traffic is consistent with statistical self-similarity for sufficiently large time scales [67]. These findings have been verified by Feldmann *et al.* [29].

Crovella has proposed an ON/OFF model for the downloading of web documents [20]. A single TCP connection is invoked in each ON period. The duration of the TCP connection follows a heavy-tailed distribution since the distribution of WWW document sizes on Internet is heavy-tailed. The OFF period corresponds to the user thinking time. Crovella argues that also the OFF period is heavy-tailed.

Deng also developed an ON/OFF model for the web service [21]. During the ON period, the user makes multiple web requests each resulting in a new TCP connection. The OFF period is the time between two ON periods while the user views the page. Deng proposed distributions for three parameters: the duration of the ON period was found to be Pareto distributed, the duration of the OFF period was found to be Weibull distributed as well as the interarrival time of web requests during the

ON period.

Feldmann proposed to model the arrivals of WWW connections by a Weibull interarrival time distribution [28]. The Weibull distribution has finite moments, including finite mean and variance of the interarrival time. For this reason the Weibull distribution is considered to have a light tail. Since its variance is finite the Weibull distribution does not give rise to a self-similar arrival process.

4.11.2 OD pair call service time distribution

The traditional model of call service times B_k is the (negative) exponential distribution with rate parameter μ. The exponential distribution matches the actual service times in case of telephony among other services. However, for the WWW and FTP service, the connection service time is more heavy tailed [20, 21, 67].

4.11.3 Link call arrival process

The arrival process offered to a given link is a result of splitting and merging of component arrival processes. First, the per-class arrival process is *splitted* over many alternative paths. Second, at each link the per-path arrival processes are *merged* to form a superposed arrival process to the link.

The model of the superposition range from an exact Markov Renewal Process (MRP) model [17] to an approximate renewal process model [79]. The Palm-Khintchine theorem states that when the number of normalized renewal processes goes to infinity and the interarrival distribution is light-tailed, the limit process becomes Poisson. Hence, we expect the renewal model to become more accurate as the number of OD pairs P increases. The MMPP model can be used to model the superposition of renewal processes before convergence to the Poisson process is achieved [36].

4.12 Modeling of packet traffic for real applications

4.12.1 Voice

A packetized voice source can be accurately modeled as an ON/OFF Markov source with exponentially distributed ON and OFF period durations [64]. The generation of packets in the ON state can be modeled by a Poisson process or a constant (fluid) rate.

4.12.2 Video

Recent measurements have shown that compressed video (e.g. MPEG) traffic exhibit self-similar behavior on a wide range of time scales [31]. It have been shown [26] that there is a knee-point of the buffer size below which the buffer distribution is dominated by the short-range correlations and above which by long-range correlations. The conclusion is that if the queue operates under conditions of low utilization and practical buffer size, the input traffic can be modeled by short-range dependent model which is generally Markovian although the input traffic has long-range dependence.

For queueing analysis purposes, the video source can be modeled by a superposition of ON/OFF Markov fluid sources or by a superposition of two-state MMPP sources. With proper matching of parameters of the autocorrelation function, the set of MMPP sources is able to produce self-similar traffic over a certain range of time scales [2]. The FBM model is another self-similar traffic model that has been suggested for compressed video.

4.12.3 Audio

From a traffic modeling perspective compressed audio (e.g. MP3) is similar to compressed video. That is, a superposition of Markov modulated fluid or Poisson sources can be used for queueing analysis [64]. However, measurements on audio traffic have failed to show self-similarity.

4.12.4 LAN

Measurements on LAN Ethernet traffic at Bellcore in the early 90s showed that Ethernet packet traffic is self-similar [54]. The aggregate Ethernet traffic can be modeled by a FBM process or by a set of ON/OFF sources with heavy tailed ON/OFF period durations.

4.12.5 WAN

Measurements on WWW and TELNET packet traffic in the Internet conducted in the mid 90s showed that such traffic is self-similar [67]. Plausible traffic models are the FBM model or a superposition of ON/OFF sources with heavy tailed ON/OFF period durations. Similar measurements on FTP data bursts within FTP sessions has shown that the FTP data burst sizes are heavy tailed.

Chapter 5

Queueing theory

5.1 Introduction

The Kendall classification of queueing systems (1953) exists in several modifications. The most comprehensive classification uses 6 symbols:

A/B/C/q/K/p

where

- **A** is the interarrival time distribution,
- **B** is the service time distribution,
- **C** is the number of servers,
- **q** is the queueing discipline (FCFS, LCFS, . . .),
- **K** is the system capacity (number of customers in service + in queue),
- **p** is the population size (numbers of possible customers). Omitted for open systems.

The following symbols are used for arrival and service distributions:

- **M** is the exponential distribution associated with the Poisson/Markov process,
- $\mathbf{E}_k$ is the Erlang distribution with parameter k,
- $\mathbf{H}_k$ is the Hyper-exponential distribution with parameter k,

- **D** is the deterministic distribution,
- **G** is a general (any) distribution,
- **GI** is a general (any) distribution with independent random values.

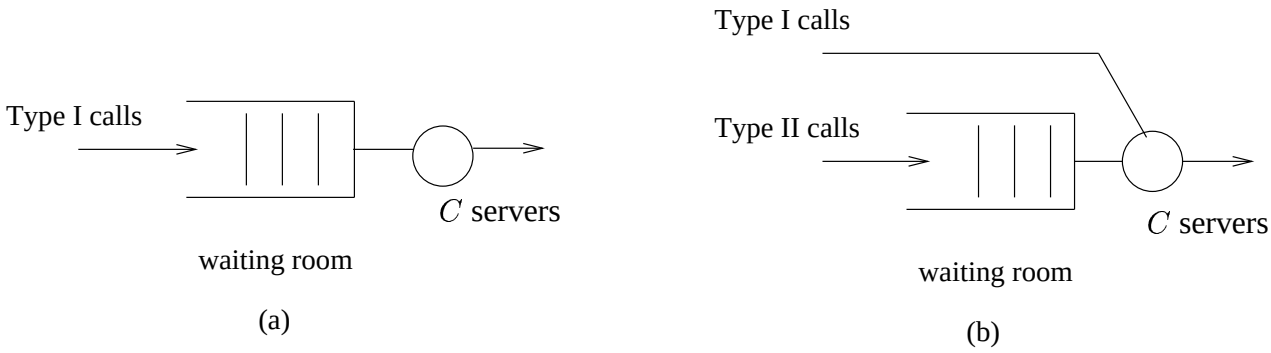


Figure 5.1: (a): $A/B/C$ delay system, and (b): $\sum A_i/B/C$ mixed loss-delay system.

The queueing systems of interest in this compendium can be classified into pure loss, pure delay and mixed loss-delay systems. We assume the following characteristics of these systems. In a loss system, customers are normally accepted in order of arrival (first come, first served) and customers are lost when no free server is available. In a delay system customers are normally served on a FCFS basis and when no free server is available, the customer is delayed in a finite or infinite waiting room. In a mixed loss-delay system we have two types of customers. Customers of type I have access the link with no restriction, but will be blocked if all servers are busy. Customers of type II have restricted access to the service facility; the cutoff parameter r_0 specifies the maximum number of type II customers that can be in service at the same time. Therefore, a queue of type II customers forms as soon as a type II arrival finds r_0 type II customers already in service or otherwise if there are not enough free servers.

The performance measures of interest in a queueing systems include:

- probability of loss (blocking),
- probability of delay,
- utilization of servers,

- average number of customers in system and in queue,
- waiting time distribution in the system and in queue,
- mean waiting time in system and in queue,

The teletraffic pioneer A.K. Erlang studied the $M/M/C$ loss system and the $M/M/C$ delay system with infinite waiting room [13]. Erlang derived the Erlang B formula for the blocking probability in a loss system, and the Erlang C formula for the delay probability in a delay system.

The superposition of the renewal processes was modeled by Cherry in [17]. His early result (1972) states that the superposition of two renewal processes is a Markov Renewal Process (MRP), which has an equivalent semi Markov Process (SMP) representation. Note that a superposition of renewal processes is autocorrelated and is therefore not renewal.

In case of a Poisson arrival process, and exponential service process, the state represents the number of customers present in the system. The state transition probabilities, which are part of the queueing model, become easy to formulate. This is due to the memoryless property of the exponential distribution: the probability of the next event being an arrival/departure is independent of the time offset between the latest arrival and the latest departure. This is not the case if we replace the Poisson process with a non-Poisson process such as the Weibull process: the probability of the next event being an arrival/departure now becomes dependent on the time between the latest arrival and the latest departure.

The queueing systems $GI/M/C$ and $M/G/1$ can be analyzed in at least two ways. In the first way, the SMP for the superposition of arrival and service process is constructed using the method of supplementary variables [18]. By introducing new state variables, which contain some temporal information, the Markov property can be preserved, resulting in a SMP. For the $GI/M/C$ system the temporal information represents the type of the latest event, and the time offset between the latest event and the latest complementary event. For the $M/G/1$ system the temporal information represents the elapsed service time. The second way is based on the embedded Markov chain method [46, 47]. In this method the arrival or departure instants are used as regeneration epochs of the Markov chain. Its main advantage is that the state space is significantly smaller than for the method of supplementary variables. The embedded Markov chain method is restricted to the case with one customer category.

The general $GI/G/C$ queue is complex to analyze, and is still subject to active research. The results achieved so far are based on approximative analytical methods or exact numerical methods.

The reference [11] contains a comprehensive summary of state-of-the-art results for the $GI/G/C$ queue. The analytical methods include the interpolation approximations by Whitt [80] and Kimura [48]. A recent analytical method is the diffusion approximation by Whitt [81]. Exact numerical methods include the work by Takahashi and Takami [75] and Bertsimas [9, 10].

5.2 Notation and structure for basic queueing systems

We consider a queueing system **A/B/C/q/K/p** according to Kendall's notation. We focus attention on the flow of customers as they arrive, pass through, and eventually leave this system; as such, we choose to number the customers with the subscript n and define C_n as follows:

C_n denotes the nth customer to enter the system

We are interested in the stochastic process $N(t)$ where

$$N(t) \stackrel{\Delta}{=} \text{ number of customers in the system at time } t \tag{5.1}$$

Another stochastic process of interest is the unfinished work $U(t)$ that exists in the system at time t, that is,

$U(t) \stackrel{\Delta}{=}$ the unfinished work in the system at time t
$\stackrel{\Delta}{=}$ the remaining time required to empty the system of all customers present at time t

Whenever $U(t) > 0$, the the system is said to be busy, and only when $U(t) = 0$ is the system said to be idle. The duration and location of these busy and idle periods are also quantities of interest.

The details of these stochastic processes may be observed by defining the following variables and then by displaying these variables on an appropriate time diagram to be discussed below. We begin to define the arrival time of customer C_n:

$$\tau_n \stackrel{\Delta}{=} \text{arrival time for } C_n \tag{5.2}$$

and further define the interarrival time between C_{n-1} and C_n as

$$t_n \stackrel{\Delta}{=} \text{interarrival time between } C_{n-1} \text{ and } C_n = \tau_n - \tau_{n-1}. \tag{5.3}$$

Since we have assumed that all interarrival times are drawn from the distribution $A(t)$, we have that

$$P[t_n \leq t] = A(t) \tag{5.4}$$

which is independent of n. Similarly, define the service time for C_n as

$$x_n \stackrel{\Delta}{=} \text{service time for } C_n \tag{5.5}$$

and from our assumptions we have

$$P[x_n \leq x] = B(x). \tag{5.6}$$

The sequences $\{t_n\}$ and $\{x_n\}$ may be thought of as input variables for our queueing system; the way in which the system handles these customers gives rise to queues and waiting times that we must now define. Thus,

$$w_n \stackrel{\Delta}{=} \text{waiting time (in queue) for } C_n. \tag{5.7}$$

The total time spent in the system by C_n is the sum of his waiting time and service time, which we define by

$$s_n \stackrel{\Delta}{=} \text{system time (queue plus service) for } C_n = w_n + x_n. \tag{5.8}$$

Finally, we introduce the concept of limiting random variable $\tilde{t}$ defined by

$$\tilde{t} \stackrel{\Delta}{=} \lim_{n \to \infty} t_n \tag{5.9}$$

which we refer to by $t_n \to \tilde{t}$. The typical notation for the probability distribution function (PDF) will be

$$P[t_n \leq t] = A_n(t) \tag{5.10}$$

and for the limiting PDF

$$P[\tilde{t} \leq t] = A(t) \tag{5.11}$$

This we denote by $A_n(t) \rightarrow A(t)$. Similarly the probability density function (pdf) for t_n and $\tilde{t}$ will be $a_n(t)$ and $a(t)$, respectively, and will be denoted $a_n(t) \rightarrow a(t)$. Finally, the Laplace transform of these pdf's will be denoted by $A_n^*(s)$ and $A^*(s)$, respectively, with the obvious notation that $A_n^*(s) \rightarrow A^*(s)$. Of course, the moments of the interarrival time are of interest and they will be denoted as follows:

$$E[\overline{t}_k] \stackrel{\Delta}{=} \overline{t^k}, k = 0, 1, 2, \ldots \tag{5.12}$$

For the first moment $\overline{t}$ a special notation has been adopted

$$\overline{t} = \frac{1}{\lambda} \tag{5.13}$$

where λ denotes the average arrival rate of customers to our queueing system.

Summarizing the information with regard to the interarrival time we have the following shorthand glossary:

$$\begin{aligned} &t_n = \text{interarrival time between } C_{n-1} \text{ and } C_n \\ &t_n \rightarrow \tilde{t}, \quad A_n(t) \rightarrow A(t), \quad a_n(t) \rightarrow a(t), \quad A_n^*(s) \rightarrow A^*(s) \\ &\overline{t}_n \rightarrow \overline{t} = \tfrac{1}{\lambda}, \quad \overline{t_n^k} \rightarrow \overline{t^k} \end{aligned} \tag{5.14}$$

In a similar manner we identify the notation associated with x_n,w_n and s_n as follows:

$$\begin{aligned} &x_n = \text{service time for } C_n \\ &x_n \rightarrow \tilde{x}, \quad B_n(t) \rightarrow B(t), \quad b_n(t) \rightarrow b(t), \quad B_n^*(s) \rightarrow B^*(s) \\ &\overline{x}_n \rightarrow \overline{x} = \tfrac{1}{\mu}, \quad \overline{x_n^k} \rightarrow \overline{x^k} \end{aligned} \tag{5.15}$$

$$\begin{aligned} &w_n = \text{waiting time for } C_n \\ &w_n \rightarrow \tilde{w}, \quad W_n(y) \rightarrow W(y), \quad w_n(y) \rightarrow w(y), \quad W_n^*(s) \rightarrow W^*(s) \\ &\overline{w}_n \rightarrow \overline{w} = W, \quad \overline{w_n^k} \rightarrow \overline{w^k} \end{aligned} \tag{5.16}$$

$$\begin{aligned} &s_n = \text{system time for } C_n \\ &s_n \rightarrow \tilde{s}, \quad S_n(y) \rightarrow S(y), \quad s_n(y) \rightarrow s(y), \quad S_n^*(s) \rightarrow S^*(s) \\ &\overline{s}_n \rightarrow \overline{s} = T, \quad \overline{s_n^k} \rightarrow \overline{s^k} \end{aligned} \tag{5.17}$$

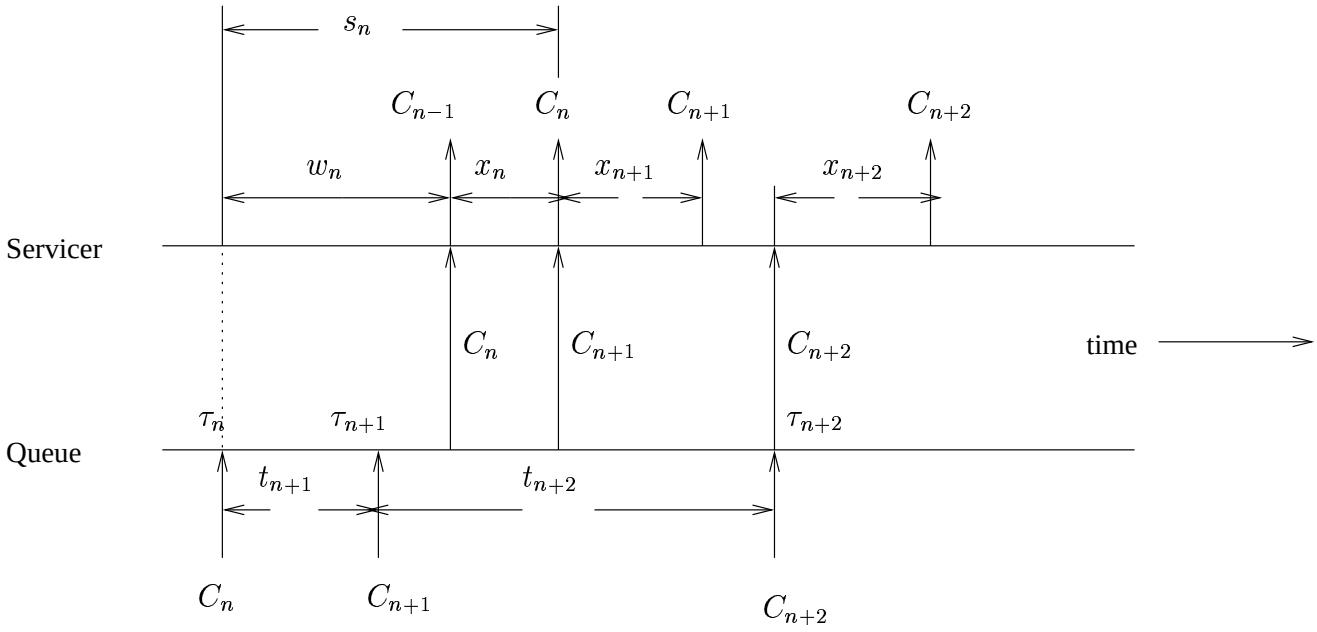


Figure 5.2: Time-diagram notation for queues.

The behavior of a queueing system random is further explained in Figure 5.2. In this time diagram the lower horizontal line represents the queue and the upper horizontal line represents the service facility; moreover the diagram shown is for the case of single server, although this too is easily generalized. An arrow approaching the queue (or service) line from below indicates that an arrival has occurred to the queue (or service facility). Arrows emanating from the line indicate the departure of a customer from the queue (or service facility). In this figure we see that the customer C_{n+1} arrives before customer C_n enters service; only when C_n departs from service may C_{n+1} enter service and, of course, these two events occur simultaneously. Notice that when C_{n+2} enters the system he finds it empty and so immediately proceeds through an empty queue directly into the service facility. In this digram we have also shown the waiting time and the system time for C_n. Thus, as time proceeds we can identify the number of customers in the system $N(t)$, the unfinished work $U(t)$, and also idle and busy periods.

Finally, we discuss a basic parameter ρ, which is commonly referred to as the *utilization factor*. The utilization factor is the ratio which "work" enters the system to the maximum rate (capacity) at which the system can perform this work; the work an arriving customer brings into the system equals

the number of seconds of service he requires. So, in the case of a single server system, the definition for ρ becomes

$$\rho \stackrel{\Delta}{=} \text{(average arrival rate of customers)} \times \text{(average service time)} = \lambda\overline{x} \tag{5.18}$$

In the case of multiple servers (say, C servers) we have

$$\rho \stackrel{\Delta}{=} \frac{\lambda\overline{x}}{C} \tag{5.19}$$

The rate at which work enters the system is sometimes referred to as the *traffic intensity* of the system and is usually expressed in *Erlangs*; in a single-server system the utilization factor is equal to the traffic intensity whereas for (C) multiple servers, the traffic intensity equals $C\rho$. So long as $0 \leq \rho < 1$, the ρ may be interpreted as

$$\rho = E[\text{fraction of busy serves}] \tag{5.20}$$

5.3 Derivation of Little's result

In a general queueing system one expects that when the number of customers is large then so is the waiting time. In this section we derive a very simple relationship between the mean number in the queueing system, the mean arrival rate of customers to that system, and the mean system time for customers.

We introduce some notation:

$$\alpha(t) \stackrel{\Delta}{=} \text{number of arrivals in } (0, t) \tag{5.21}$$

$$\delta(t) \stackrel{\Delta}{=} \text{number of departures in } (0, t) \tag{5.22}$$

A sample path of the stochastic variables $\alpha(t)$ and $\delta(t)$ is shown in Figure 5.3.

Clearly $N(t)$, the number in the system at time t, must be given by

$$N(t) = \alpha(t) - \delta(t) \tag{5.23}$$

On the other hand, the total area between these two curves up to some point, say t, represents the total time all customers have spent in the system (measured in units customer-seconds) during the interval

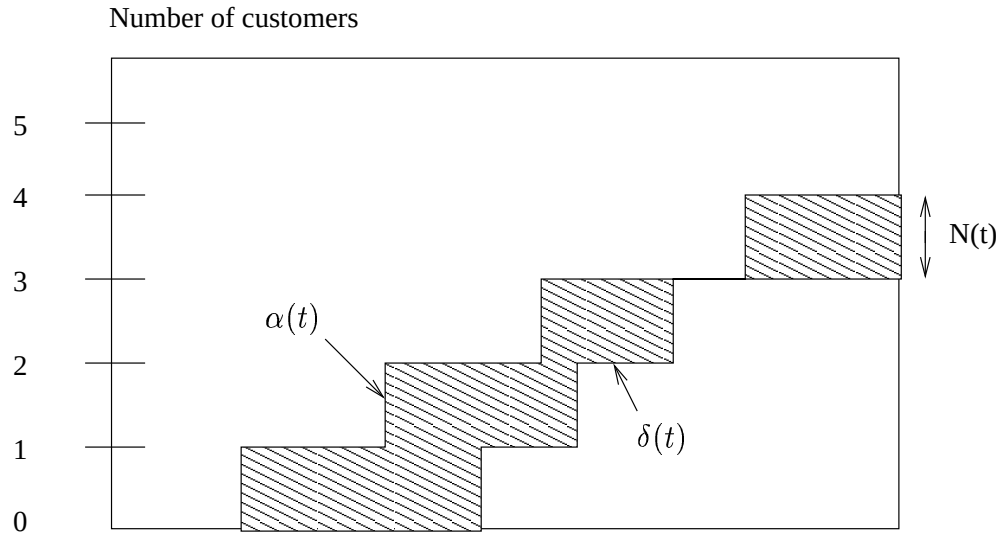


Figure 5.3: Arrivals and departures.

$(0, t)$; let us denote this cumulative area by $\gamma(t)$. Moreover, let λ_t be defined as the average arrival rate (customers per second) during the interval $(0, t)$; that is;

$$\lambda_t \triangleq \frac{\alpha(t)}{t} \tag{5.24}$$

We may define T_t as the system time per customer averaged over all customers in the interval $(0, t)$; since $\gamma(t)$ represents the accumulated customer-seconds up to time t, we may divide by the number of arrivals up to that point to obtain:

$$T_t = \frac{\gamma(t)}{\alpha(t)} \tag{5.25}$$

Lastly, let us define N_t as the average number of customers in the queueing system during the interval $(0, t)$; this may be obtained by dividing the accumulated number of customer-seconds by the total interval length t:

$$N_t = \frac{\gamma(t)}{t} \tag{5.26}$$

From these last three equations we see

$$N_t = \lambda_t T_t \tag{5.27}$$

Let us now assume that our queueing system is such that the following limits exists as $t \to \infty$:

$$\lambda = \lim_{t\to\infty} \lambda_t \tag{5.28}$$

$$T = \lim_{t\to\infty} T_t \tag{5.29}$$

These two limits corresponds to our former definitions for λ and T representing the average arrival rate and the average system time. If the limits exists, so must the limit for N_t, so we have

$$N = \lambda T \tag{5.30}$$

This is the result we are seeking and it is known as *Little's result*.

Little's result also holds for the average number of customers in the queue N_q, average arrival rate λ and average waiting time in the queue W:

$$N_q = \lambda W \tag{5.31}$$

Finally, note that its always true that

$$T = \overline{x} + W. \tag{5.32}$$

5.4 M/M/C queueing system

In this section we evaluate the performance of queueing system with C parallel servers and infinite waiting room operating under the first-come-first-served queueing discipline. We assume the system is offered customers from a single customer category with Poisson customer arrival process and exponentially distributed service times. The interarrival time have a exponential distribution according with mean arrival rate λ [s^{-1}] and the mean service time is $1/\mu$ [s]. Each customer is assumed to require 1 unit of capacity. The state of the system is denoted by n and represents the number of customers present in the system (in service and in queue).

With the assumptions on exponential service time distribution and Poisson customer arrivals we observe that

- we only need to look at the transition between time t and $t + \Delta t$ for $\Delta t \to 0$.

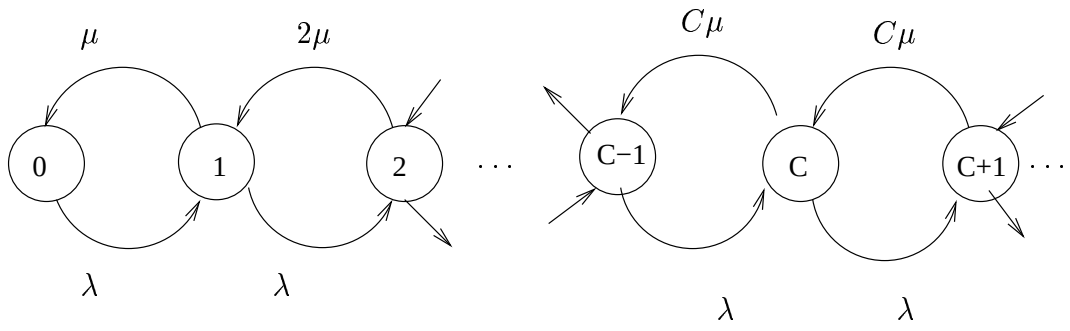


Figure 5.4: State transition diagram for infinite customer queueing system.

- if there are n customers in the queue at time $t + \Delta t$, then there could have be only $n - 1, n$ or $n + 1$ customers at time t

With these assumptions we can write:

$$
\begin{aligned}
p_0(t+\Delta t) &= p_0(t)[(1-\lambda\Delta t) + \mu\Delta t\lambda\Delta t] + p_1(t)[\mu\Delta t(1-\lambda\Delta t)] \\
p_n(t+\Delta t) &= p_n[(1-n\mu\Delta t)(1-\lambda\Delta t) + n\mu\Delta t\lambda\Delta t] + \\
&\quad + p_{n+1}[(n+1)\mu\Delta t(1-\lambda\Delta t)] + p_{n-1}[\lambda\Delta t(1-(n-1)\mu\Delta t)], \quad 1 \le n \le C, \\
p_n(t+\Delta t) &= p_n[(1-C\mu\Delta t)(1-\lambda\Delta t) + C\mu\Delta t\lambda\Delta t] + \\
&\quad + p_{n+1}[(C+1)\mu\Delta t(1-\lambda\Delta t)] + p_{n-1}[\lambda\Delta t(1-C\mu\Delta t)], \qquad n > C.
\end{aligned}
$$

According to Taylor series we have

$$
p_n(t+\Delta t) = p_n(t) + \frac{\mathrm{d}}{\mathrm{dt}} p_n(t)\Delta t \tag{5.33}
$$

Combining the two equations above and letting $\Delta t \to 0$ we have

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{dt}} p_0(t) &= -\lambda p_0(t) + \mu p_1(t) \\
\frac{\mathrm{d}}{\mathrm{dt}} p_n(t) &= -(\lambda + n\mu) p_n(t) + \lambda p_{n-1}(t) + (n+1)\mu p_{n+1}(t), \quad 1 \le n \le C, \\
\frac{\mathrm{d}}{\mathrm{dt}} p_n(t) &= -(\lambda + C\mu) p_n(t) + \lambda p_{n-1}(t) + (C+1)\mu p_{n+1}(t), \quad n > C.
\end{aligned}
$$

These are the differential-difference equations which describe the state of the system as function of time. We are interested in steady state behavior so we set $\frac{\mathrm{d}}{\mathrm{dt}} p_n(t) = 0$ for all n, which yields

$$
\begin{aligned}
0 &= -\lambda\pi_0 + \mu\pi_1 \\
0 &= -(\lambda + n\mu)\pi_n + \lambda\pi_{n-1} + (n+1)\mu\pi_{n+1}, \quad 1 \le n \le C, \\
0 &= -(\lambda + C\mu)\pi_n + \lambda\pi_{n-1} + (C+1)\mu\pi_{n+1}, \quad n > C
\end{aligned}
$$

From these equations we have

$$
\begin{aligned}
\lambda\pi_0 &= \mu\pi_1 \\
\lambda\pi_1 &= 2\mu\pi_2 \\
\vdots \quad &\quad \vdots \quad \vdots \\
\lambda\pi_{C-1} &= C\mu\pi_C \\
\lambda\pi_C &= C\mu\pi_{C+1} \\
\vdots \quad &\quad \vdots \quad \vdots
\end{aligned}
\tag{5.34}
$$

As $\rho = \frac{\lambda}{C\mu}$ is the offered traffic, we get:

$$
\pi_n = \begin{cases} \pi_0 \frac{(C\rho)^n}{n!} & n \le C, \\ \pi_0 \frac{C^C \rho^n}{C!} & n > C. \end{cases}
$$

All probabilities have to sum up to one:

$$
\sum_{n=0}^{\infty} \pi_n = 1 \tag{5.35}
$$

Therefore:

$$
\pi_0 = \left[\sum_{n=0}^{C-1} \frac{(C\rho)^n}{n!} + \frac{(C\rho)^C}{C!(1-\rho)} \right]^{-1} \tag{5.36}
$$

Inserting the expression for π_0 in (5.35) yields Erlang's C formula:

$$
P(\text{delay}) = \sum_{n=C}^{\infty} \pi_n = \frac{(C\rho)^C}{C!(1-\rho)} \times \left[\sum_{n=0}^{C-1} \frac{(C\rho)^n}{n!} + \frac{(C\rho)^C}{C!(1-\rho)} \right]^{-1} \tag{5.37}
$$

The average number of customers in the queue, N_q, is given by:

$$
N_q = \frac{\pi_0 (C\rho)^C \rho}{C!(1-\rho)^2} \tag{5.38}
$$

The average time in the queue, W, is given by Little's formula:

$$W = \frac{N_q}{\lambda} \tag{5.39}$$

The average time in the system, T, is obtained from:

$$T = W + \frac{1}{\mu} \tag{5.40}$$

The average number of customers in the system, N, is given by Little's formula:

$$N = \lambda T = N_q + \frac{\lambda}{\mu} \tag{5.41}$$

5.5 M/M/C/* loss system

In this section we evaluate the performance of loss system with C parallel servers and no waiting room operating under the first-come-first-served queueing discipline. We assume the system is offered calls from a single call category with Poisson call arrival process and exponentially distributed service times. The interarrival time have a exponential distribution according with mean arrival rate λ [s^{-1}] and the mean service time is $1/\mu$ [s]. Each call is assumed to require 1 unit of capacity. The state of the system is denoted by n and represents the number of busy servers.

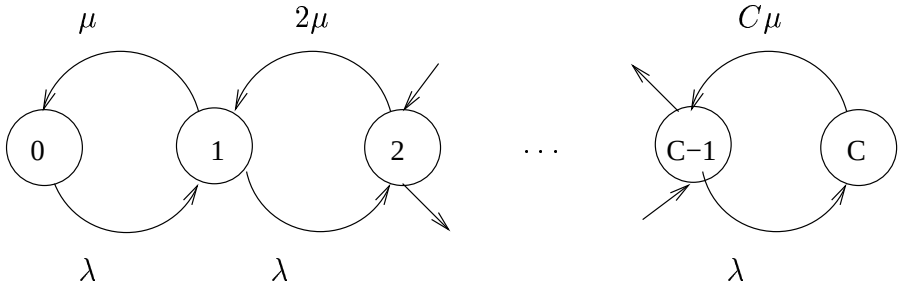


Figure 5.5: State transition diagram for customer loss system.

By the same way as in the previous section we obtain the differential equations:

$$\frac{\mathrm{d}}{\mathrm{dt}}p_n(t) = -(\lambda + n\mu)p_n(t) + \lambda p_{n-1}(t) + (n+1)\mu p_{n+1}(t) \tag{5.42}$$

$$\frac{\mathrm{d}}{\mathrm{dt}}p_0(t) = -\lambda p_0(t) + \mu p_1(t) \tag{5.43}$$

We are interested in steady state behavior so we set $\frac{\mathrm{d}}{\mathrm{dt}}p_n(t) = 0$ for all n, which yields

$$0 = -(\lambda + n\mu)\pi_n + \lambda\pi_{n-1} + (n+1)\mu\pi_{n+1} \tag{5.44}$$

$$0 = -\lambda\pi_0 + \mu\pi_1 \tag{5.45}$$

Hence, the probabilities π_n are

$$\pi_1 = \frac{\lambda}{\mu}\pi_0 \tag{5.46}$$

$$\pi_2 = \frac{(\lambda+\mu)\frac{\lambda}{\mu}\pi_0 - \lambda\pi_0}{2\mu} = \frac{\lambda^2}{2\mu}\pi_0 \tag{5.47}$$

$$\pi_3 = \frac{(\lambda+2\mu)\frac{\lambda^2}{2\mu^2}\pi_0 - \lambda\frac{\lambda}{\mu}\pi_0}{3\mu} = \frac{\lambda^3}{6\mu^3}\pi_0 \tag{5.48}$$

$$\pi_4 = \frac{(\lambda+3\mu)\frac{\lambda^3}{6\mu^3}\pi_0 - \lambda\frac{\lambda^2}{\mu^2}\pi_0}{4\mu} = \frac{\lambda^4}{4!\mu^4}\pi_0 \tag{5.49}$$

$$\vdots \tag{5.50}$$

$$\pi_n = \left[\prod_{i=1}^{n}\frac{\lambda}{i\mu}\right]\pi_0 = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n\pi_0 \tag{5.51}$$

Normalization of the probabilities, $\sum_{n=0}^{C}\pi_n = 1$, yields

$$\sum_{n=0}^{C}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n\pi_0 = 1 \tag{5.52}$$

resulting in the empty system probability π_0

$$\pi_0 = \frac{1}{\sum_{n=0}^{C}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n} \tag{5.53}$$

A customer which find a system with C busy servers must be rejected. Hence, the customer blocking probability, $E(C,\rho)$, where $\rho = \lambda/\mu$, is

$$E(C,\rho) = \frac{\frac{1}{C!}\rho^C}{\sum_{n=0}^{C}\frac{1}{n!}\rho^n} \tag{5.54}$$

which also is known as the Erlang blocking formula.

5.6 GI/M/C queueing system

5.6.1 Introduction

In this section we evaluate the performance of queueing system with C parallel servers and an infinite waiting room operating under the first-come-first-served queueing discipline. The presentation is based on Kleinrock [49]. We assume the system is offered calls from a single call category with renewal call arrival process and exponentially distributed service times. The interarrival time have a general distribution according to $A(t)$ and the mean service time is $1/\mu$ [s]. Each call is assumed to require 1 unit of capacity throughout its holding time.

5.6.2 The embedded Markov chain method

A one-dimensional embedded Markov chain can be defined at the arrival instants. Let X_n be the number of customers found upon arrival of the nth customer. Then X_n is a one-dimensional embedded Markov chain with infinite countable state space

$$X = \{x : x = 0, 1, \ldots\}. \tag{5.55}$$

Let v_n be a stochastic process of discrete time $n = 1, 2, \ldots$ representing the number of customers served between the arrival of the $(n-1)$th and the nth customer. Then

$$X_{n+1} = X_n + 1 - v_{n+1}. \tag{5.56}$$

We must now calculate the state transition probabilities associated with this Markov chain, and so we define

$$p_{ij} = P[X_{n+1} = j | X_n = i] \tag{5.57}$$

It is clear that p_{ij} is merely the probability that $i + 1 - j$ customers are served during an interarrival time. It is further clear that

$$p_{ij} = 0 \quad \text{for } j > i + 1 \tag{5.58}$$

since there are at most $i+1$ customers present between the arrival of the nth and the $(n+1)$th customer. The Markov chain has transitions such as shown in Figure 5.6.

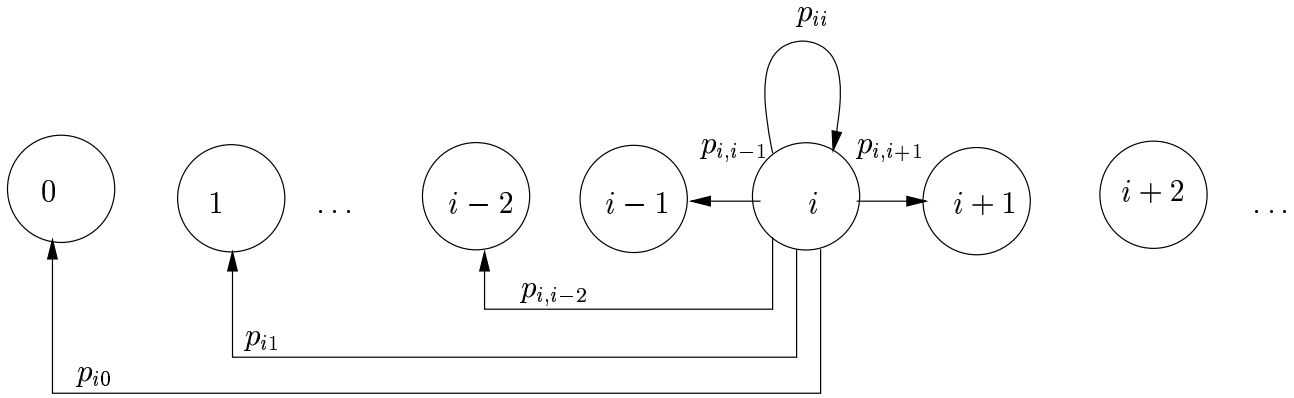


Figure 5.6: State-transition-probability diagram for the $GI/M/C$ embedded Markov chain.

For the system to be ergodic we require that the utilization factor $\rho < 1$, where

$$\rho = \frac{\lambda}{C\mu} \tag{5.59}$$

In the ergodic case we are assured that an equilibrium probability distribution will exist describing the number present at the arrival instants. Thus we define

$$r_k = \lim_{n\to\infty} P[X_n = k] \tag{5.60}$$

and it is this probability we seek for the $GI/M/C$ system.

Transition probabilities

Our first task is to find the one-step transition probabilities. We consider four regions in the (i, j) plane:

1. $i + 1 < j$,
2. $j \leq i + 1 \leq C$,
3. $C \leq j \leq i + 1, i \geq C$,

4. $j < C < i + 1$.

For region 1 we already know that $p_{ij} = 0$ for $i + 1 < j$. For region 2 we have

$$p_{ij} = \int_0^\infty \begin{pmatrix} i+1 \\ j \end{pmatrix} [1 - e^{-\mu t}]^{i+1-j} e^{-j\mu t} dA(t), \quad j \le i + 1 \le C. \tag{5.61}$$

For region 3 we have

$$p_{ij} = \int_0^\infty \frac{(C\mu t)^{i+1-j}}{(i+1-j)!} e^{-C\mu t} dA(t), \quad C \le j \le i + 1, i \ge C. \tag{5.62}$$

For region 4 we have

$$p_{ij} = \int_0^\infty \begin{pmatrix} C \\ j \end{pmatrix} e^{-j\mu t} \times \left[\int_0^t \frac{(C\mu y)^{i-C}}{(i-C)!} (e^{-\mu y} - e^{-\mu t})^{C-j} C\mu dy \right] dA(t), \quad j < C < i + 1. \tag{5.63}$$

Having established the form for our one-step transition probabilities we may place them in the transition matrix

$$\mathbf{P} = \begin{pmatrix}
p_{00} & p_{01} & 0 & 0 & & & & & \\
p_{10} & p_{11} & p_{12} & 0 & \dots & & & & \\
p_{20} & p_{21} & p_{22} & p_{23} & & & & & \\
\vdots & & & \vdots & & & & & \\
p_{C-2,0} & p_{C-2,1} & & \dots & p_{C-2,C-1} & 0 & 0 & 0 & \dots \\
p_{C-1,0} & p_{C-1,1} & & \dots & p_{C-1,C-1} & \beta_0 & 0 & 0 & \dots \\
p_{C,0} & p_{C,1} & & \dots & p_{C,C-1} & \beta_1 & \beta_0 & 0 & \dots \\
 & \vdots & & & & & & & \\
p_{C+n,0} & p_{C+n,1} & & \dots & p_{C+n,C-1} & \beta_{n+1} & \beta_n & \dots & \beta_0 \\
 & \vdots & & & & & & & \vdots
\end{pmatrix} \tag{5.64}$$

Calculation of equilibrium distribution

Define for $k \ge C - 1$:

$N_k(t)$= number of arrival instants in the interval $(0,t)$ in which the arriving customer finds the system in state E_k, given 0 customers at time $t=0$.

σ_k = E[number of times state E_{k+1} is reached between two successive visits to state E_k].

It can be shown that σ_k is independent of k and that σ must be the limit of the ratio of the number of times we find ourselves in state E_{k+1} to the number of times we find ourselves in state E_k; thus we may write

$$\sigma = \lim_{t\to\infty} \frac{N_{k+1}(t)}{N_k(t)}, \qquad k \geq C-1. \tag{5.65}$$

It can be shown that the limit (5.65) exists. However, this limit must be the ratio of the steady state probability of finding the system in state E_{k+1} to the probability of finding it in state E_k. Consequently, we have established

$$\frac{r_{k+1}}{r_k} = \sigma, \qquad k \geq C-1. \tag{5.66}$$

The solution to this last set of equations is clearly

$$r_k = K\sigma^k, \qquad k \geq C-1, \tag{5.67}$$

for some constant K. This is a basic result, which says that the distribution of the number of customers found at the arrival instants is *geometric* for the case $k \geq C-1$. It remains for us to find σ and K, as well as r_k for $k < C-1$.

We have for $k \geq C$:

$$r_k = K\sigma^k = \sum_{i=0}^{\infty} r_i p_{ik} = \sum_{i=k-1}^{\infty} r_i p_{ik} = \sum_{i=k-1}^{\infty} K\sigma^i \beta_{i+1-k}. \tag{5.68}$$

where

$$\beta_n = p_{i,i+1-n} = \int_0^{\infty} \frac{(C\mu t)^n}{n!} e^{-C\mu t} dA(t), \quad 0 \leq n \leq i+1-C, C \leq i. \tag{5.69}$$

Canceling the constant K as well as the common factors we have

$$\sigma = \sum_{i=k-1}^{\infty} \sigma^{i+1-k} \beta_{i+1-k}. \tag{5.70}$$

Changing the index of summation we finally have

$$\sigma = \sum_{n=0}^{\infty} \sigma^n \beta_n. \tag{5.71}$$

Inserting the expression for β_n gives:

$$\sigma = \sum_{n=0}^{\infty} \sigma^n \int_{t=0}^{\infty} \frac{(C\mu t)^n}{n!} e^{-C\mu t} dA(t) = \int_{t=0}^{\infty} e^{-(C\mu - C\mu\sigma)t} dA(t). \tag{5.72}$$

It can be shown that so long as $\rho < 1$ then there is a unique real solution for σ in the range $0 < \sigma < 1$.

The equilibrium distribution is of the form

$$\mathbf{r} = K\sigma^{C-1}[R_0, R_1, \ldots, R_{C-2}, 1, \sigma, \sigma^2, \ldots] \tag{5.73}$$

where

$$R_k = \frac{r_k \sigma^{1-C}}{K}, \quad k = 0, 1, \ldots, C-2 \tag{5.74}$$

Furthermore, for convenience we define

$$J = K\sigma^{C-1} \tag{5.75}$$

In terms of our one-step transition probabilities we have

$$R_k = \sum_{t=k-1}^{\infty} R_i p_{ik}, \quad k = 0, 1, \ldots, C-2, \tag{5.76}$$

where we extend the definition of R_k such that $R_i = \sigma^{i-C+1}$ for $i \geq C-1$. The tail of the sum above may be evaluated to give

$$R_k = \sum_{i=k-1}^{C-2} R_i p_{ik} + \sum_{i=C-1}^{\infty} \sigma^{i+1-C} p_{ik} \tag{5.77}$$

Solving for R_{k-1}, the lower-order term present, we have

$$R_{k-1} = \frac{1}{p_{k-1,k}} \left[R_k - \sum_{i=k}^{C-2} R_i p_{ik} - \sum_{i=C-1}^{\infty} \sigma^{i+1-C} p_{ik} \right] \quad k = 1, 2, \ldots, C-1. \tag{5.78}$$

The set of equations (5.78) is a triangular set in the unknowns R_k; in particular we may start with the fact that $R_{C-1} = 1$ and then solve recursively over the range $k = C-1, C-2, \ldots, 1, 0$ in order.

Finally we may use the conservation of probability to evaluate the constant J as

$$J = \sum_{k=0}^{C-2} R_k + J \sum_{k=C-1}^{\infty} \sigma^{k-C+1} = 1 \tag{5.79}$$

$$J = 1/[\frac{1}{1-\sigma} + \sum_{k=0}^{C-2} R_k] \tag{5.80}$$

Calculation of performance measures

The probability that an arriving will be delayed the $GI/M/C$ queue is

$$P(\text{delay}) = \frac{K\sigma^C}{1-\sigma} \tag{5.81}$$

The average waiting time for the $GI/M/C$ queue is

$$W = \frac{K\sigma^C}{C\mu(1-\sigma)^2} = \frac{J\sigma}{C\mu(1-\sigma)^2} \tag{5.82}$$

The delay distribution for the $GI/M/C$ queue is

$$W(y) = 1 - \sigma e^{-C\mu(1-\sigma)y}/[1 + (1-\sigma)\sum_{k=0}^{C-2} R_k], \quad y \geq 0. \tag{5.83}$$

5.7 M/G/1 queueing system

5.7.1 Introduction

In this section we evaluate the performance of queueing system with one server and an infinite waiting room operating under the first-come-first-served queueing discipline. The presentation is based on Kleinrock [49]. We assume the system is offered customers from a single customer category with Poisson customer arrival process and generally distributed service times. The interarrival time have an exponential distribution with mean interarrival time $1/\lambda$ [s] and the mean service time is $1/\mu$ [s]. Each customer is assumed to require 1 unit of capacity throughout its service time.

5.7.2 The embedded Markov chain method

A one-dimensional embedded Markov chain can be defined at the departure instants. Let X_n be the number of customers found left behind the departure of the nth customer. Then X_n is a one-dimensional embedded Markov chain with infinite countable state space

$$X = \{x : x = 0, 1, \ldots\}, \tag{5.84}$$

Let v_n be a stochastic process of discrete time $n = 1, 2, \ldots$ representing the number of customers arriving during service of the nth customer. Then

$$X_{n+1} = \begin{cases} X_n - 1 + v_{n+1}, & X_n > 0 \\ v_{n+1}, & X_n = 0 \end{cases} \tag{5.85}$$

Let us introduce the shifted discrete step function

$$\Delta_{X_k} = \begin{cases} 1, & k = 1, 2, \ldots \\ 0, & k \leq 0 \end{cases} \tag{5.86}$$

Hence, we may write the system equation as

$$X_{n+1} = X_n - \Delta_{X_n} + v_{n+1}, \tag{5.87}$$

We define the one-step transition probabilities

$$p_{ij} = P[X_{n+1} = j | X_n = i] \tag{5.88}$$

Since these transitions are observed only at departures, it is clear that $X_{n+1} < X_n - 1$ is an impossible situation; on the other hand, $X_{n+1} \geq X_n - 1$ is possible for all values due to the arrivals v_{n+1}. It is easy to see that the matrix of transition probabilities $\mathbf{P} = [p_{ij}], (i, j = 0, 1, 2, \ldots)$, takes the form:

$$\mathbf{P} = \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \ldots \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \ldots \\ 0 & \alpha_0 & \alpha_1 & \alpha_2 & \ldots \\ 0 & 0 & \alpha_0 & \alpha_1 & \ldots \\ 0 & 0 & 0 & \alpha_0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{5.89}$$

where $\alpha_k = P[v_{n+1} = k]$.

Let us now calculate α_k. The service time for the nth customer is independent of n and distributed according to $B(x)$. Therefore, v_n, the number of arrivals during the service time x_n depends only upon the duration of x_n and not upon n at all. We may therefore dispense with the subscripts on v_n and x_n, replacing them with random variables $\tilde{v}$ and $\tilde{x}$ so that we may write $P[x_n \leq x] = P[\tilde{x} \leq x] = B(x)$ and $P[v_n = k] = P[\tilde{v} = k] = \alpha_k$. We have by the total law of probability

$$\alpha_k = P[\tilde{v} = k] = \int_0^\infty P[\tilde{v} = k, x < \tilde{x} \leq x + dx]dx \tag{5.90}$$

By conditional probabilities we further have

$$\alpha_k = \int_0^\infty P[\tilde{v} = k | \tilde{x} = x]b(x)dx \tag{5.91}$$

where $b(x) = dB(x)/dx$ is the pdf for the service time. Since we have a Poisson arrival process, we have

$$\alpha_k = \int_0^\infty \frac{(\lambda x)^k}{k!} e^{-\lambda x} b(x)dx \tag{5.92}$$

Our Markov chain is ergodic if the utilization factor ρ fulfills

$$\rho = \lambda \overline{x} < 1, \tag{5.93}$$

where $\overline{x}$ denote the average service time.

The mean queue length

Im this section we derive the Pollaczek-Khinchin formula for the mean value of the limiting queue length. In particular, we define

$$\tilde{X} = \lim_{n \to \infty} X_n \tag{5.94}$$

Assume the jth moment of X_n exists in the limit as n goes to infinity independent of n:

$$\lim_{n \to \infty} E[X_n^j] = E[\tilde{X}_n^j] \tag{5.95}$$

Previously we defined the system equation as

$$X_{n+1} = X_n - \Delta_{X_n} + v_{n+1}, \tag{5.96}$$

Taking the expectation we get

$$E[X_{n+1}] = E[X_n] - E[\Delta_{X_n}] + E[v_{n+1}] \tag{5.97}$$

Using (5.95) we have, in the limit as $n \to \infty$,

$$E[\tilde{X}] = E[\tilde{X}] - E[\Delta_{\tilde{X}}] + E[\tilde{v}] \tag{5.98}$$

Alas, the expectation we are seeking drops out of the equation, which instead yields:

$$E[\Delta_{\tilde{X}}] = E[\tilde{v}] \tag{5.99}$$

What insight does this last equation provide us? We have by definition that

$$E[\tilde{v}] = \text{average number of arrivals in a service time} \tag{5.100}$$

Let us now interpret the left-hand side of (5.99). By definition we have

$$E[\Delta_{\tilde{X}}] = \sum_{k=0}^{\infty} \Delta_k P[\tilde{X} = k] = 0P[\tilde{X} = 0] + 1P[\tilde{X} > 0] = P[\tilde{X} > 0] \tag{5.101}$$

Equation (5.101) may also be written as

$$E[\Delta_{\tilde{X}}] = P[\text{busy system}] = P[\text{delay}] \tag{5.102}$$

And from our definition of utilization factor we further have

$$P[\text{busy system}] = \rho \tag{5.103}$$

Thus from the above equations we conclude that

$$E[\tilde{v}] = \rho \tag{5.104}$$

We now return to the task of solving for the expected value of $\tilde{X}$. Forming the first moment of the system equation yielded interesting results but failed to give the desired expectation. Let us now attempt to find this value by first squaring the system equation and the taking the expectations as follows:

$$X_{n+1}^2 = X_n^2 + \Delta_{X_n}^2 + v_{n+1}^2 - 2X_n\Delta_{X_n} + 2X_n v_{n+1} - 2\Delta_{X_n} v_{n+1} \tag{5.105}$$

From the definition of the shifted discrete step function we have $(\Delta_{X_n})^2 = \Delta_{X_n}$ and also $X_n\Delta_{X_n} = X_n$. Applying this to (5.105) and taking expectations, we have

$$E[X_{n+1}^2] = E[X_n^2] + E[\Delta_{X_n}] + E[v_{n+1}^2] - 2E[X_n] + 2E[X_n v_{n+1}] - 2E[\Delta_{X_n} v_{n+1}] \tag{5.106}$$

Taking the limit as n goes to infinity, and using our limit assumption (5.95) we have

$$0 = E[\Delta_{\tilde{X}}] + E[\tilde{v}^2] - 2E[\tilde{X}] + 2E[\tilde{X}]E[\tilde{v}] - 2E[\Delta_{\tilde{X}}]E[\tilde{v}] \tag{5.107}$$

Using equations (5.99) and (5.104) we obtain as an intermediate result for the expectation of $\tilde{X}$:

$$E[\tilde{X}] = \rho + \frac{E[\tilde{v}^2] - E[\tilde{v}]}{2(1-\rho)} \tag{5.108}$$

The only unknown here is $E[\tilde{v}^2]$.

Let us solve not only for the second moment of $\tilde{v}$ but, in fact, let us describe a method for obtaining *all* the moments. Equation (5.92) gives an expression for $\alpha_k = P[\tilde{v} = k]$. Using this equation we calculate the moments as follows. Let us define the z-transform for the random variable $\tilde{v}$ as

$$V(z) \stackrel{\Delta}{=} E[z^{\tilde{v}}] = \sum_{k=0}^{\infty} P[\tilde{v} = k] z^k \tag{5.109}$$

Forming $V(z)$ from equations (5.92) and (5.109) we have

$$V(z) = \sum_{k=0}^{\infty} \int_0^{\infty} \frac{(\lambda x)^k}{k!} e^{-\lambda x} b(x) dx z^k \tag{5.110}$$

We may interchange the order of the summation and integration:

$$V(z) = \int_0^{\infty} e^{-\lambda x} \left(\sum_{k=0}^{\infty} \frac{(\lambda x z)^k}{k!} \right) b(x) dx = \int_0^{\infty} e^{-(\lambda - \lambda z)x} b(x) dx \tag{5.111}$$

At this point we define the Laplace transform $B^*(s)$ for the service time pdf as

$$B^*(s) \triangleq \int_0^\infty e^{-sx} b(x) dx \tag{5.112}$$

We note that equation (5.111) is of this form, with the complex variable s replaced by $\lambda - \lambda z$, and so we recognize the important result that

$$V(z) = B^*(\lambda - \lambda z) \tag{5.113}$$

Various derivatives of z-transforms evaluated at $z = 1$ give the various moments of the random variable under consideration. Similarly, the appropriate derivative of the Laplace transform evaluated at its argument $s = 0$ also give rise to moments. In particular,

$$B^{*(k)}(0) \triangleq \left.\frac{d^k B^*(s)}{ds^k}\right|_{s=0} = (-1)^k E[\tilde{x}^k] \tag{5.114}$$

$$V^{(1)}(1) \triangleq \left.\frac{dV(z)}{dz}\right|_{z=1} = E[\tilde{v}] \tag{5.115}$$

$$V^{(2)}(1) \triangleq \left.\frac{d^2V(z)}{dz^2}\right|_{z=1} = E[\tilde{v}^2] - E[\tilde{v}] \tag{5.116}$$

In order to simplify the notation for these limiting derivative operations, we have used the more usual supscript notation with the argument replace by its limit. Furhermore, we now resort to the overbar notation to denote expected value of the random variable below the bar. Thus equation (5.114) - (5.116) become

$$B^{*(k)}(0) = (-1)^k \overline{x^k} \tag{5.117}$$

$$V^{(1)}(1) = \overline{v} \tag{5.118}$$

$$V^{(2)}(1) = \overline{v^2} - \overline{v} \tag{5.119}$$

Form the conservation of probability we have

$$B^*(0) = V(1) = 1 \tag{5.120}$$

Taking the appropriate derivative of (5.113) we have

$$\frac{dV(z)}{dz} = \frac{dB^*(\lambda - \lambda z)}{dz} \tag{5.121}$$

The right-hand side can be written

$$\frac{dB^*(\lambda - \lambda z)}{dz} = \frac{dB^*(\lambda - \lambda z)}{d(\lambda - \lambda z)} \frac{d(\lambda - \lambda z)}{dz} = -\lambda \frac{dB^*(y)}{dy} \tag{5.122}$$

where

$$y = \lambda - \lambda z \tag{5.123}$$

Setting $z = 1$ in (5.121) we have

$$V^{(1)}(1) = -\lambda \left. \frac{dB^*(y)}{dy} \right|_{z=1} \tag{5.124}$$

But from equation (5.123) the case $z = 1$ is the case $y = 0$, and so we have

$$V^{(1)}(1) = -\lambda B^{*(1)}(0) \tag{5.125}$$

Combining equations (5.114) and (5.124) we finally have

$$\overline{v} = \lambda \overline{x} \tag{5.126}$$

But $\lambda \overline{x} = \rho$ and we have once again established that $\overline{v} = \rho$. We may continue to pick up higher moments by differentiating equation (5.121) once again to obtain

$$\frac{d^2V(z)}{dz^2} = \frac{d^2B^*(\lambda - \lambda z)}{dz^2} \tag{5.127}$$

Using the first derivative of $B^*(y)$ we now form the second derivative as follows:

$$\frac{d^2B^*(\lambda - \lambda z)}{dz^2} = \frac{d}{dz}\left[-\lambda \frac{dB^*(y)}{dy}\right] = -\lambda \frac{d^2B^*(y)}{dy^2} \frac{dy}{dz} \tag{5.128}$$

or

$$\frac{d^2B^*(\lambda - \lambda z)}{dz^2} = \lambda^2 \frac{d^2B^*(y)}{dy} \tag{5.129}$$

Setting z equal to 1 in equation (5.121) we have

$$V^{(2)}(1) = \lambda^2 B^{*(2)}(0) \tag{5.130}$$

Thus from earlier results in equation (5.114) we obtain

$$\overline{v^2} - \overline{v} = \lambda^2 \overline{x^2} \tag{5.131}$$

Returning to our intermediate result (5.108) we apply equation (5.131) to obtain

$$\overline{X} = \rho + \rho^2 \frac{\lambda^2 \overline{x^2}}{2(1-\rho)} \tag{5.132}$$

Let us rewrite this result in terms of $C_b^2 = \sigma_b^2/\overline{x}^2$, the squared coefficient of variation for service time:

$$\overline{X} = \rho + \rho^2 \frac{(1+C_b^2)}{2(1-\rho)} \tag{5.133}$$

This is the well-known formula for the average number of customers in a $M/G/1$ system and is commonly referred to as the *Pollaczek-Khinchin (P-K) mean-value formula*.

According to Little's result:

$$N = \lambda T \tag{5.134}$$

where T denotes the average time the customer spends in the the system and $N = \overline{X}$. Thus we have

$$N = \rho + \rho^2 \frac{(1+C_b^2)}{2(1-\rho)} = \lambda T \tag{5.135}$$

Solving for T we have

$$T = \overline{x} + \frac{\rho \overline{x}(1+C_b^2)}{2(1-\rho)} \tag{5.136}$$

This is easily interpreted. The average total time in the system is clearly the average time spent in service plus the average time spent in the queue. Hence, the average waiting time in the queue is

$$W = \frac{\rho \overline{x}(1+C_b^2)}{2(1-\rho)} \tag{5.137}$$

5.8 Fluid flow queueing system

5.8.1 Assumptions and notation

In this section we evaluate the performance of a fluid flow queueing system. The presentation is based on Jacobsen and Dittman [39]. We assume the FCFS multiplexer is offered traffic from c classes

of ON/OFF sources. An ON/OFF source alternates between an active ON state, when it transmits information at a peak rate, and a silent OFF state. The durations of the ON and OFF periods are assumed to be exponentially distributed.

A traffic class i is described by four parameters:

- number of sources: N_i
- average duration of the ON state: $1/b_i$ [s]
- average duration of the OFF state: $1/a_i$ [s]
- peak rate in the ON state: p_i [bps]

The intensity (rate) of transitions from the ON (OFF) state to the OFF (ON) state for a class i source is $b_i(a_i)[\text{s}^{-1}]$. The mean rate for class i is given by $m_i = p_i \frac{a_i}{a_i+b_i}$ [bps].

The traffic descriptor (peak rate, mean rate, maximum burst size) can be enforced by two leaky buckets or two token buckets, one supervising the peak rate and one supervising the mean rate and the maximum burst size. It is widely believed, though the author have seen no formal mathematical proof, that the worst case output from a leaky bucket or token bucket with respect to queue build up is a periodic ON/OFF process in which the source is transmitting at peak as long as allowed and then turns silent until it is able to transmit a maximum burst at peak again. From numerical experience it is known that more variable ON and OFF period durations lead to larger queueing build up. Therefore the cell loss ratio in the exponential distributed ON/OFF queue will be an upper bound on the cell loss probability in the periodic ON/OFF queue. Hence, the ON $\rightarrow$ OFF intensity b_i should be set to $b_i = p_i/MBS$. The OFF $\rightarrow$ ON intensity a_i should be set to $a_i = \frac{b_i m_i}{p_i - m_i}$.

The data transmitted by the $\sum N_i$ sources is received by a finite buffer of size B Mbits with a maximum output rate of C bps. The buffer is modelled as a fluid reservoir with a hole in the bottom and arriving information is modelled as a fluid running into the reservoir. If p_{in} is the input rate, and x is the buffer content, then the change in buffer content is given by

$$dx/dt = \begin{cases} 0 & \text{when } x = 0 \text{ and } p_{in} < C \\ p_{in} - C & \text{when } \begin{cases} x = 0 \text{ and } p_{in} > C \\ 0 < x < B \\ x = B \text{ and } p_{in} < C \end{cases} \\ 0 & \text{when } x = B \text{ and } p_{in} > C \end{cases} \tag{5.138}$$

The average input rate is $\sum_{i=1}^{c} N_i p_i a_i/(a_i + b_i)$ and the load on the output is then given by

$$\rho = \frac{\sum_{i=1}^{c} N_i p_i \frac{a_i}{a_i+b_i}}{C} \tag{5.139}$$

We assume the average input rate to be smaller than the output capacity, i.e. $\rho < 1$.

Let k_i denote the number of active sources in class i, and $\mathbf{k} = (k_1, ..., k_c)$ the state vector which the sources are in. Let

$$S := \{\mathbf{k} = (k_1, ..., k_c) : 0 \leq k_i \leq N_i, i = 1, ..., c\} \tag{5.140}$$

denote the state space for the sources. S is of cardinality $(N_1 + 1) \cdots (N_c + 1)$.

Vectors and matrices will appear in bold letter such that they can be distinguished from numbers, functions etc.

Let $\mathbf{p} = (p_1, ..., p_c)$ denotes the peak rate vector for the sources and by the scalar product of $\mathbf{k}$ and $\mathbf{f}$ we mean:

$$p_{in} = \mathbf{k} \cdot \mathbf{p} = \sum_{i=1}^{c} k_i p_i \tag{5.141}$$

which is the total input rate in state $\mathbf{k}$.

The assumption that the duration of the ON (OFF) state is exponential ensures that the transitions between state in S are determined by a Markov chain, and this together with the fluid assumption enables us to find the equilibrium buffer distribution as solution to a set of first order differential equations.

A set of differential equations

For $\mathbf{k}$ in S, $t > 0, 0 < x < B$, let $P_{\mathbf{k}}(t, x)$ denote the probability that at time t, the sources are in state $\mathbf{k}$ and the buffer content does not exceed x. Due to the exponential assumption, at the next small time interval $(t, t + \Delta t)$ a source from class i will switch from the OFF state to the ON state with probability $(N_j - k_j)a_j\Delta t + o(\Delta t)$, and a source from class i will switch from the ON state to the OFF state with probability $k_j b_j \Delta t + o(\Delta t)$. The probability of two or more changes is $o(\Delta t)$. These considerations yields the following expression:

$$P_{\mathbf{k}}(t+\Delta t,x) = \sum_{i=1}^{c}(N_i-k_i+1)a_i\Delta t P_{(k_1,\ldots,k_i-1,\ldots,k_c)}(t,x-\Delta x)$$
$$\sum_{i=1}^{c}(k_i+1)b_i\Delta t P_{(k_1,\ldots,k_i+1,\ldots,k_c)}(t,x-\Delta x)$$
$$(1-\sum_{i=1}^{c}(N_i-k_i)a_i\Delta t + k_i b_i\Delta t)P_{\mathbf{k}}(t,x-\Delta x) \tag{5.142}$$

where $\Delta x = (\sum_{i=1}^{c} p_i k_i - C)\Delta t$ and where a function $g(t)$ is written $o(t)$ whenever $g(t)/t \to 0$ for $t \to 0$. Isolating terms containing $P_{\mathbf{k}}$ on the left side, dividing by Δt and taking the limit $\Delta t \to 0$, yields

$$\frac{\partial P_{\mathbf{k}}}{\partial t}(t,x) + (\sum_{i=1}^{c} p_i k_i - C)\frac{\partial P_{\mathbf{k}}}{\partial x}(t,x) = -\sum_{i=1}^{c}((N_i-k_i)a_i + k_i b_i)P_{\mathbf{k}}(t,x)$$
$$+\sum_{i=1}^{c}(N_i-k_i+1)a_i P_{(k_1,\ldots,k_i-1,\ldots,k_c)}(t,x)$$
$$+\sum_{i=1}^{c}(k_i+1)b_i P_{(k_1,\ldots,k_i+1,\ldots,k_c)}(t,x) \tag{5.143}$$

We are only interested in the time independent equilibrium probabilities

$F_{\mathbf{k}}(x)$:= equilibrium probability that the sources are in state $\mathbf{k}$ and the buffer content does not exceed x and we therefore put $\partial P_{\mathbf{k}}/\partial t = 0$ and obtain

$$(\sum_{i=1}^{c} p_i k_i - C)\frac{\partial F_{\mathbf{k}}}{\partial x}(x) = -\sum_{i=1}^{c}(N_i-k_i)a_i + k_i b_i)F_{\mathbf{k}}(x)$$
$$+\sum_{i=1}^{c}(N_i-k_i+1)a_i F_{(k_1,\ldots,k_i-1,\ldots,k_c)}(x)$$
$$+\sum_{i=1}^{c}(k_i+1)b_i F_{(k_1,\ldots,k_i+1,\ldots,k_c)}(x) \tag{5.144}$$

where $F_{\mathbf{k}} := 0$ when k_i is not in $\{0,1,...,N_i\}$. Equation (5.144) can be written as a $(N_1+1)\cdots(N_c+1)$ dimensional matrix differential equation

$$\mathbf{D}\frac{d\mathbf{F}}{dx} = \mathbf{M}\mathbf{F}(x) \tag{5.145}$$

where $\mathbf{D}$ is a diagonal matrix with entry $(\mathbf{k},\mathbf{k})$ equal to

$$d_{\mathbf{k}} = (\sum_{i=1}^{c} p_i k_i - C) \tag{5.146}$$

and where entry $(\mathbf{k}, \mathbf{n})$ in $\mathbf{M}$ looks as follows:

$$\begin{aligned} m_{(\mathbf{k},\mathbf{k})} &= -\sum_{i=1}^{c}(N_i - k_i)a_i + k_i b_i, \quad \text{for } \mathbf{k} \in S \\ m_{(\mathbf{k},k_1,\ldots,k_i-1,\ldots,k_c)} &= (N_i - k_i + 1)a_i, \quad \text{for } \mathbf{k} \in S \\ m_{(\mathbf{k},k_1,\ldots,k_i+1,\ldots,k_c)} &= (k_i + 1)b_i, \quad \text{for } \mathbf{k} \in S \end{aligned} \tag{5.147}$$

In both approaches to be presented it is necessary to be able to invert the matrix $\mathbf{D}$. To ensure this, the following technical assumption is needed:

$$C \in \left\{ \sum_{i=1}^{c} p_i k_i | k_i \text{ is an integer} \right\} \tag{5.148}$$

Equation (5.148) is a mild assumption and in the case where all the p_i's are multiples of p_1 the condition (5.148) is fulfilled whenever C is not a multiple integer of p_1.

The (vector) solution to the matrix differential equation is:

$$\mathbf{F}(x) := \exp(\mathbf{D}^{-1}\mathbf{M}x)\mathbf{f}_0 \tag{5.149}$$

where $\mathbf{F}(x) = \{F_{\mathbf{k}}(x)\}_{\mathbf{k}\in S}$ and where the initial vector $\mathbf{f}_0$ is found from initial conditions.

The initial condition

Let $u_{\mathbf{k}}$ denote the probability of the buffer being held at its maximum B, and the sources being in state $\mathbf{k}$. If $q_{\mathbf{k}}$ denotes the overall probability of the sources being in state $\mathbf{k}$, then the following expression is valid for $u_{\mathbf{k}}$

$$u_{\mathbf{k}} = q_{\mathbf{k}} - \lim_{x\to B} F_{\mathbf{k}}(x) \tag{5.150}$$

where $q_{\mathbf{k}}$is to be calculated as:

$$q_{\mathbf{k}} = \prod_{i=1}^{c} \frac{\binom{N_i}{k_i}(\frac{a_i}{b_i})^{k_i}}{(1+\frac{a_i}{b_i})^{N_i}} \tag{5.151}$$

Using this notation the initial condition is easy to formulate. When the input rate is larger than the output rate the buffer cannot stay empty. Therefore:

$$F_{\mathbf{k}}(0) = 0 \text{ when } \sum_{i=1}^{c} k_i p_i > C \tag{5.152}$$

When the input rate is smaller than the output rate the buffer cannot stay at its maximum. Therefore:

$$0 = u_{\mathbf{k}} = q_{\mathbf{k}} - \lim_{x \to B} F_{\mathbf{k}}(x) \text{ when } \sum_{i=1}^{c} k_i p_i < C \tag{5.153}$$

5.8.2 The solution

We seek fundamental solutions of the form

$$F(x) = \exp(z(\mathbf{k})x)\phi_{\mathbf{k}} \tag{5.154}$$

Inserting in the vector-matrix differential equation (5.145) yields:

$$\begin{aligned} \mathbf{D}z(\mathbf{k})\exp(z(\mathbf{k})x)\phi_{\mathbf{k}} = &\quad \mathbf{M}\exp(z(\mathbf{k})x)\phi_{\mathbf{k}} \\ \left[\mathbf{D}^{-1}\mathbf{M} - z(\mathbf{k})I\right]\phi_{\mathbf{k}} = &\quad 0 \\ \mathbf{D}^{-1}\mathbf{M}\phi_{\mathbf{k}} = &\quad z(\mathbf{k})\phi_{\mathbf{k}} \end{aligned} \tag{5.155}$$

Hence, the matrix $\mathbf{D}^{-1}\mathbf{M}$ has eigenvectors $\phi_{\mathbf{k}}$ and eigenvalues $z(\mathbf{k})$, for $\mathbf{k} \in S$. Since the dimension of $\mathbf{D}^{-1}\mathbf{M}$ is equal to $(N_1 + 1) \cdots (N_c + 1)$, it is possible to use S as index set. With this notation the solution will be a linear combination of fundamental solutions:

$$\mathbf{F}(x) = \sum_{\mathbf{k} \in S} a_{\mathbf{k}} \exp(z(\mathbf{k})x)\phi_{\mathbf{k}} \tag{5.156}$$

where the coefficients $a_{\mathbf{k}}$ must be found by means of the initial condition which in the eigenvalue context reads:

$$\begin{aligned} 0 = F_{\mathbf{n}}(0) = \sum_{i=1}^{c} a_{\mathbf{k}}(\phi_{\mathbf{k}})_{\mathbf{n}} &\quad \text{when } \mathbf{n} \cdot \mathbf{p} > C \\ q_{\mathbf{n}} = \lim_{x \to B} F_{\mathbf{n}}(x) = \sum_{i=1}^{c} a_{\mathbf{k}} \exp(z(\mathbf{k})x)(\phi_{\mathbf{k}})_{\mathbf{n}} &\quad \text{when } \mathbf{n} \cdot \mathbf{p} < C \end{aligned} \tag{5.157}$$

where $(\phi_{\mathbf{k}})_{\mathbf{n}}$ denotes the $\mathbf{n}$-th component of the $\mathbf{k}$-th eigenvector.

In the infinite buffer case with identical sources, the initial condition is different, and it can be formulated such that the corresponding coefficient matrix is a Van der Monde matrix, and an analytical solution exist. However, in the finite buffer case with a more general input stream, no such approach seems possible, and the equation must be solved numerically.

The eigenvalue for state $\mathbf{k}$ can be found by solving the algebraic equation $f(z(\mathbf{k})) = g(z(\mathbf{k}))$ where

$$f(z(\mathbf{k})) = z(\mathbf{k})(C - \sum_{i=1}^{c} \frac{N_i}{2} p_i) - \sum_{i=1}^{c} \frac{N_i}{2}(a_i + b_i) \tag{5.158}$$

$$g(z(\mathbf{k})) = \sum_{i=1}^{c} (k_i - \frac{N_i}{2}) \sqrt{(z(\mathbf{k})p_i + b_i - a_i)^2 + 4a_i b_i} \tag{5.159}$$

The eigenvector $\phi_{\mathbf{k}}$ corresponding to the eigenvalue $z(\mathbf{k})$ is given as the coefficients in the following polynomial in c variables:

$$p_{\mathbf{k}} = \prod_{i=1}^{c} (x_i - r_i(z))^{k_i} (x_i - s_i(z))^{N_i - k_i} \tag{5.160}$$

where

$$r_i(z) = \frac{-(zp_i + b_i - a_i) + \sqrt{(zp_i + b_i - a_i)^2 + 4a_i b_i}}{2a_i} \tag{5.161}$$

$$s_i(z) = \frac{-(zp_i + b_i - a_i) - \sqrt{(zp_i + b_i - a_i)^2 + 4a_i b_i}}{2a_i} \tag{5.162}$$

The size of the state space for the fluid flow model is $N_s = \prod_{i=1}^{c} (N_i + 1)$. As far as computational complexity is concerned, is is dominated by the calculation of the coefficients $a_{\mathbf{k}}$ from the boundary conditions. The coefficients are found by solving a set of N_s linear equations. This system of equations can be solved by Gauss elimination or, which can be more efficient, by some iterative method. In any case, the associated complexity is in the order of $O(N_s^3)$. Hence, the complexity of the fluid flow model increases very fast with increasing number of classes c and increasing class sizes N_i. Therefore, the fluid flow model is only considered to be useful as a reference model, and not as a basis for implementation of call admission control in real multi-service networks.

5.8.3 Performance formulas

In this subsection we present formulas for the buffer overflow probability, packet loss ratio, and mean waiting time in the queue.

The equilibrium distribution can be written as

$$\mathbf{F}(x) = \sum_{\{\mathbf{k}|\mathbf{k}\cdot\mathbf{p}>C\}} a_{\mathbf{k}} \exp(z(\mathbf{k})x)\phi_{\mathbf{k}} + \mathbf{q} \tag{5.163}$$

where $\mathbf{q}$ is the equilibrium probability distribution.

The buffer overflow probability $G(x) = \mathrm{P}(Q > x)$ can be written as

$$\begin{aligned} G(x) = 1 - \sum_{\mathbf{k}\in S} F_{\mathbf{k}}(x) &= 1 - \sum_{\mathbf{k}\in S} q_{\mathbf{k}} - \sum_{\mathbf{k}\in S} \sum_{\{\mathbf{n}|\mathbf{n}\cdot\mathbf{p}>C\}} a_{\mathbf{n}} \exp(z(\mathbf{n})x)(\phi_{\mathbf{n}})_{\mathbf{k}} \\ &= - \sum_{\mathbf{k}\in S} \sum_{\{\mathbf{n}|\mathbf{n}\cdot\mathbf{p}>C\}} a_{\mathbf{n}} \exp(z(\mathbf{n})x)(\phi_{\mathbf{n}})_{\mathbf{k}} \end{aligned} \tag{5.164}$$

For large buffers the following asymptotic approximation is valid:

$$G(x) \sim -a_{\mathbf{0}} \exp(z(\mathbf{0})x) \sum_{\mathbf{k}\in S} (\phi_{\mathbf{0}})_{\mathbf{k}} \tag{5.165}$$

where $z(\mathbf{0})$ is the largest negative eigenvalue and $\phi_{\mathbf{0}}$ its associated eigenvector.

The overall loss ratio p_{loss} is defined as the fraction lost information to offered information. Loss can only take place when the buffer is at maximum and the input rate is larger than the output rate. Therefore we get:

$$p_{loss} = \frac{\sum_{\{\mathbf{k}|\mathbf{k}\cdot\mathbf{p}>C\}} (\mathbf{k}\cdot\mathbf{p} - C) u_{\mathbf{k}}}{\sum_{i=1}^{c} N_i p_i \frac{a_i}{a_i + b_i}} \tag{5.166}$$

The buffer overflow probability is an upper bound of the packet loss ratio.

The distribution of queueing delay, $H(t) = \mathrm{P}(T \le t)$, is given by:

$$H(t) = \sum_{\mathbf{k}\in S} F_{\mathbf{k}}(Ct) \sum_{i=1}^{c} \frac{k_i}{\rho_i N_i} \frac{N_i}{N} = \sum_{\mathbf{k}\in S} F_{\mathbf{k}}(Ct) \sum_{i=1}^{c} \frac{k_i}{\rho_i N} \tag{5.167}$$

where $N = \sum_{i=1}^{c} N_i$ denotes the total number of sources, and $\rho_i = \frac{a_i}{a_i + b_i}$ denotes the probability that an class i source is in the active state.

This expression (5.167) can be used to determine the the $1-\alpha$ quantile, $D_{queueing}(\alpha)$, of the queueing delay distribution. The delay quantile can be found by searching for the value of t which yields $H(t) = 1-\alpha$.

The mean waiting time in the queue, μ, is obtained as

$$\mu = \int_0^{B/C} t \mathrm{d}H(t) \tag{5.168}$$

The variance of the waiting time in the queue, σ^2, is obtained as

$$\sigma^2 = \int_0^{B/C} (t-\mu)^2 \mathrm{d}H(t) \tag{5.169}$$

5.9 Bufferless fluid flow model

5.9.1 Exact solution

The exact packet loss ratio for a superposition of heterogeneous ON/OFF Markov fluid sources offered to a bufferless multiplexer is:

$$p_{loss} = \frac{\sum_{\{\mathbf{k}|\mathbf{k}\cdot\mathbf{p}>C\}} (\mathbf{k}\cdot\mathbf{p} - C) q_{\mathbf{k}}}{\sum_{i=1}^{c} N_i p_i \frac{a_i}{a_i+b_i}} \tag{5.170}$$

where $q_{\mathbf{k}}$ is the equilibrium probability of being in state :

$$q_{\mathbf{k}} = \prod_{i=1}^{c} \frac{\binom{N_i}{k_i} (\frac{a_i}{b_i})^{k_i}}{(1+\frac{a_i}{b_i})^{N_i}} \tag{5.171}$$

The computational complexity of the exact method may be estimated by [55]

$$O\left[(c+N)\prod_{i=1}^{c} N_i\right] \text{ where } N = \sum_{i=1}^{c} N_i \tag{5.172}$$

Hence, the computational complexity increases geometrically with the size of the classes. Also this method is considered to be too complex to serve as a basis for call admission control in real multi-service networks.

5.9.2 Integrated Chernoff bound approximation

The cell loss ratio is defined as the mean loss rate, L, to mean offered rate, M: $p_{loss} = L/M$. The mean rate M is given by

$$M = \sum_{i=1}^{c} p_i \rho_i \tag{5.173}$$

where $\rho_i = \frac{a_i}{a_i + b_i}$ denotes the activity factor for class i.

In the following, we will consider an efficient approximative method for computing L [55]. Let T be a sum of the peak rate p_i of individual sources:

$$T = \sum_{i=1}^{c} N_i p_i \tag{5.174}$$

If $T \leq C$, L is immediately zero by definition. If not, we may transform L into an integral from C to T of the complementary distribution function, $F_c(x) = \mathrm{P}(X > x)$, for a stochastic variable X representing the input rate.

$$N = -\int_C^T (x - C) \mathrm{d}F_c(x) = \int_C^T F_c(x) \mathrm{d}x \tag{5.175}$$

where $F_c(x)$ obeys the Chernoff bound:

$$F_c(x) = P(X > C) = \int_C^\infty f(x) \mathrm{d}x \leq \int_0^\infty e^{s(x-C)} f(x) \mathrm{d}x = e^{-sC} \int_0^\infty e^{sx} f(x) \mathrm{d}x = \frac{E[e^{sX}]}{e^{sC}} \tag{5.176}$$

Hence, we have

$$L < E[e^{sX}] \int_C^T e^{-sx} \mathrm{d}x < \frac{E[e^{sX}]}{se^{sC}} \tag{5.177}$$

where we have replaced T with infinity to obtain the final inequality. Let $f(s)$ denote the right hand side of Equation (5.177), which is a parametric upper bound of L.

$$f(s) = \frac{E[e^{sX}]}{se^{sC}} = \frac{E[e^{s(X-C)}]}{s} \tag{5.178}$$

If the input rate is larger than C, then $f(s)$ has the minimum value at $s = s^*$ that is the root of the equation

$$\frac{\mathrm{d} \log E[e^{sX}]}{\mathrm{d}s} - \frac{1}{s} - C = 0 \tag{5.179}$$

Proof: The second derivative of $f(s)$ is always positive for $s > 0$, and hence $f(s)$ is a convex function for $s > 0$. From Equation (5.178), $\lim_{s \to 0} f(s) = \infty$, and if $T > C$, then $\lim_{s \to \infty} f(s) =$

∞. Function $f(s)$ thus has the minimum value with respect to s. s^* minimizing $f(s)$ is the root of $\mathrm{d}f(s)/\mathrm{d}s = 0$. Applying Equation (5.178) yields Equation (5.179).

If $T > C$, then we use the minimum value of $f(s)$ as the best approximation of L. Thus we have

$$p_{loss} = \frac{\mathrm{Min}_{s>0} f(s)}{M} = \frac{f(s^*)}{M} \tag{5.180}$$

This approximation is referred to as the Integrated Chernoff Bound (ICB).

The moment generating function $E[e^{sX}]$ is obtained as

$$E[e^{sX}] = \prod_{i=1}^{c} E[e^{sX_i}] = \prod_{i=1}^{c} [(1-\rho_i) + \rho_i e^{sp_i}]^{N_i} \tag{5.181}$$

In conclusion, we have

$$L = \frac{\prod_{i=1}^{c} [(1-\rho_i) + \rho_i e^{s^* p_i}]^{N_i}}{s^* e^{s^* C}} \tag{5.182}$$

where s^* is the root of the following equation:

$$\sum_{i=1}^{c} \frac{N_i \rho_i p_i e^{sp_i}}{\rho_i (e^{sp_i} - 1) + 1} - \frac{1}{s} - C = 0 \tag{5.183}$$

This nonlinear equation can be solved by Newton's method or the Interval halfing method. The above two formulas reduces the p_{loss} computational complexity to order $O(c)$. The ICB method has sufficiently low complexity to serve as the basis for call admission control in real multi-service networks.

An alternative to the ICB method is the Integrated Large Deviation (ILD) method, which provides slightly tighter cell loss ratio bounds while having the same computational complexity.

Chapter 6

Queueing networks

6.1 Introduction

We have so far considered Markovian systems in which each customer was demanding a single service operation from the system. We may refer to this as a "single-node" system. In this chapter we are concerned with multiple-node systems in which a customer requires service at more than one station (node). Thus we may think of a *network of nodes*, each of which is a service center (perhaps with multiple servers at some of the nodes) and each with a storage room for queues to form. Customers enter the system at various points, queue for service, and upon departure from a given node then proceed to some other node, there to receive additional service. A packet-switched network with packet queues in the routers is an example of a queueing network.

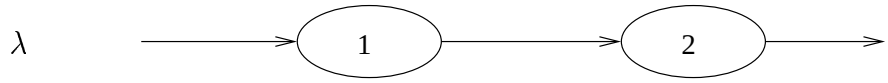


Figure 6.1: A two-node tandem network.

A number of new considerations emerge when one considers networks. For example, the topological structure of the network is important since it describes the permissible transitions between nodes. Also the paths taken by individual customers must somehow be described. Of great significance is the nature of stochastic flow in terms of the basic stochastic processes describing that flow; for example, in the case of a tandem queue, customers departing from node i immediately enter node $i + 1$, we see that the inter-departure times from the former generate the interarrival times to the latter. Let us

for the moment consider the simple two-node tandem network shown in Figure 6.1. Each oval in that figure describes a queueing system consisting of a queue and server(s); within each oval is given the node number. (It is important not to confuse these physical *network* diagrams with the abstract *state-transition-rate* diagrams we have seen earlier.) For the moment let us assume that a Poisson process generates the arrivals to the system at rate λ, all of which enter node one; further assume that node one consists of a single exponential server a rate μ. Thus node one is exactly an M/M/1 queueing system. As we will assume that node two has a single exponential server also of rate μ. The basic question is to solve for the interarrival time distribution feeding node two; this certainly will be equivalent to the interdeparture time distribution of node one. Let $d(t)$ be the pdf describing the interdeparture process from node one and as usual let its Laplace transform be denoted by $D^*(s)$. Let us now calculate $D^*(s)$. When a customer departs from node one either a second customer is available in the queue and ready to be taken into service immediately or the queue is empty. In the first case, the time until this next customer departs from node one will be distributed exactly as the service time and in that case we will have

$$D^*(s)|_{\text{node one nonempty}} = B^*(s) \tag{6.1}$$

On the other hand, if the node is empty this first customer's departure then we must wait for the sum of two intervals, the first being the time until the second customer arrives and the next being his service time; since these two intervals are independently distributed then the PDP of the sum must be the convolution of the pdf's for each. Certainly then the transform of the sum pdf will be the product of the transforms of the individual pdfs and so we have

$$D^*(s)|_{\text{node one empty}} = \frac{\lambda}{s+\lambda} B^*(s) \tag{6.2}$$

where we have given the explicit expression for the transform of the interarrival time density. Since we have an exponential server we may also write $B^*(s) = \mu/(s+\mu)$; furthermore the probability of a departure leaving behind an empty system is the same as the probability of an arrival finding an empty system, namely $1-\rho$. This permits is to write down the unconditional transform for the interdeparture time density as

$$D^*(s) = (1-\rho)D^*(s)|_{\text{node one empty}} + \rho D^*(s)|_{\text{node one nonempty}} \tag{6.3}$$

Using our above calculations we have

$$D^*(s) = (1-\rho)\left(\frac{\lambda}{s+\lambda}\right)\left(\frac{\mu}{s+\mu}\right) + \rho\left(\frac{\mu}{s+\mu}\right) \tag{6.4}$$

Little algebra gives

$$D^*(s) = \frac{\lambda}{s+\lambda} \tag{6.5}$$

and so the interdeparture distribution is given by

$$D(t) = 1 - e^{-\lambda t}, \qquad t \geq 0. \tag{6.6}$$

Thus we find the remarkable conclusion that the interdeparture times are exponentially distributed with the same parameter as the interarrival times! In other words (in the case of a stable stationary queueing system), a Poisson process driving an exponential server generates a Poisson process for departures. This startling result is usually referred to as *Burke's* theorem [14]. In fact, Burke's theorem says more, namely that the steady-state output of a stable M/M/m queue with input parameter λ and service-time parameter μ for each of the m channels is in fact a Poisson process at the same rate λ. Burke also established that the output process was independent of the other processes in the system,

6.2 Open queueing networks

Jackson considered an arbitrary network of FCFS queues [40]. The system he studied consists of M nodes where the ith node consists of m_i exponential servers each with parameter μ_i; further the ith node receives arrivals from the outside system in the form of a Poisson process at rate λr_{si}. Thus, if $M = 1$ then we have an M/M/m system. Upon leaving the ith node a customer then proceeds to the jth node with probability r_{ij}: this formulation permits the case where $r_{ii} \geq 0$. On the other hand, after completing service in the ith node the probability that the customer departs from the network (never to return again) is given by $1 - \sum_{j=1}^{M} r_{ij}$. The notation is described in Figure 6.2.

In each queue, the service time of the customer is drawn independent of the service times in other queues. Note that in a packet network the sending time of a packet, in reality, is the same in all queues (or differs by a constant factor, the inverse of the line speed). The dependence, however, does not markedly affect the behavior of the system (so called Kleinrock's independence assumption.)

We must calculate the total average rate of customers to a given node. To do so, we must sum the (Poisson) arrivals from outside the system plus arrivals (not necessarily Poisson) from all internal

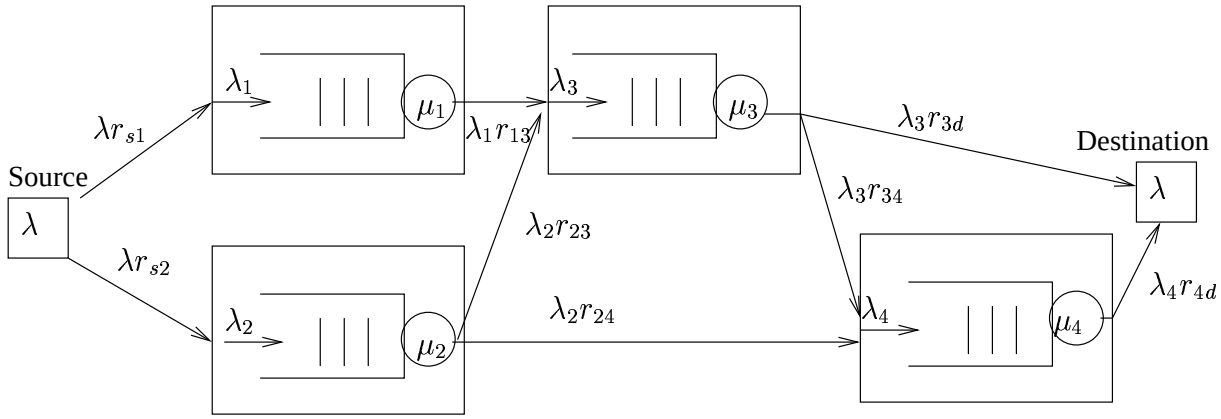


Figure 6.2: Example of an open queueing network.

nodes; that is, denoting the total average arrival rate to node i by λ_i we easily find that this set of parameters must satisfy the following equations:

$$\lambda_i = \lambda r_{si} + \sum_{j=1}^{M} \lambda_j r_{ji}, \qquad i = 1, \ldots, M. \tag{6.7}$$

In order to for all nodes in the system to represent ergodic Markov chains we require that $\lambda_i < m_i \mu_i$ for all i. What is amazing is that Jackson was able to show that each node (say the ith) in the network behaves as if it were an independent M/M/m system with Poisson input rate λ_i. In general, the total input will *not* be a Poisson process. The state variable for this M-node system consists of the vector $\mathbf{k} = (k_1, k_2, \ldots, k_M)$, where k_i is the number of customers in the i node (including the customer(s) in service). Let the equilibrium probability associated with this state be denoted by $\pi(k_1, k_2, \ldots, k_M)$. Similarly we denote the marginal distribution of finding k_i customers at the i node by $\pi_i(k_i)$. Jackson was able to show that the joint distribution for all nodes factored into the product of each of the marginal distributions. The following is known as *Jackson's theorem*:

$$\pi(\mathbf{k}) = \prod_{i=1}^{M} \pi_i(k_i) \tag{6.8}$$

where $\pi_i(k_i)$ is given as the solution to the classical $M/M/m_i$ system:

$$\pi(k_i) = \begin{cases} \pi(0) \frac{(C\rho)^{k_i}}{k_i!} & k_i \leq m_i, \\ \pi(0) \frac{C^C \rho^{k_i}}{C!} & k_i > m_i. \end{cases} \tag{6.9}$$

where $\rho = \frac{\lambda}{m_i \mu}$ is the utilization factor, and

$$\pi(0) = \left[\sum_{k_i=0}^{m_i-1} \frac{(m_i\rho)^{k_i}}{k_i!} + \frac{(m_i\rho)^{m_i}}{m_i!(1-\rho)} \right]^{-1} \tag{6.10}$$

Jackson's result is obtained from the flow balance equations:

$$\begin{aligned} \lambda\pi(\mathbf{k}) + \sum_{i=1}^{M} \mu_i 1_{k_i>0} \pi(\mathbf{k}) = \lambda \sum_{i=1}^{M} r_{si} \pi(\mathbf{k} - \mathbf{e}_i) \\ + \sum_{i=1}^{M} r_{id} \mu_i \pi(\mathbf{k} + \mathbf{e}_i) \\ + \sum_{i=1}^{M} \sum_{j=0}^{M} r_{ji} \mu_j \pi(\mathbf{k} + \mathbf{e}_j - \mathbf{e}_i) \end{aligned} \tag{6.11}$$

where the left-hand side represents the probability flow out of state $\mathbf{k}$, and the right-hand side the probability flow to state $\mathbf{k}$. The quantity $1_{k_i>0}$ is one when the condition $k_i > 0$ is true and zero otherwise.

Rewrite the factor λr_{si} in the first term on the right-hand side by means of the flow conservation equation $\lambda r_{si} = \lambda_i - \sum_{j=1}^{M} \lambda_j r_{ji}$ The equation becomes:

$$\begin{aligned} \lambda\pi(\mathbf{k}) + \sum_{i=1}^{M} \mu_i 1_{k_i>0} \pi(\mathbf{k}) = \sum_{i=1}^{M} \lambda_i \pi(\mathbf{k} - \mathbf{e}_i) - \sum_{i=1}^{M} \sum_{j=1}^{M} \lambda_j r_{ji} \pi(\mathbf{k} - \mathbf{e}_i) \\ + \sum_{i=1}^{M} r_{id} \mu_i \pi(\mathbf{k} + \mathbf{e}_i) \\ + \sum_{i=1}^{M} \sum_{j=0}^{M} r_{ji} \mu_j \pi(\mathbf{k} + \mathbf{e}_j - \mathbf{e}_i) \end{aligned} \tag{6.12}$$

We insert the product form solution of Jackson's theorem as a trial and show how the equation indeed is satisfied. The following relations hold for the product form trial

$$\begin{cases} \lambda_i \pi(\mathbf{k} - \mathbf{e}_i) = \mu_i 1_{n_i>0} \pi(\mathbf{k}) \\ \lambda_j \pi(\mathbf{k} - \mathbf{e}_i) = \mu_j \pi(\mathbf{k} - \mathbf{e}_i + \mathbf{e}_j) \\ \lambda_i \pi(\mathbf{k}) = \mu_i \pi(\mathbf{k} + \mathbf{e}_i) \end{cases} \tag{6.13}$$

Substitute these relations, in this order, into the first three terms of the right-hand side. The equation now takes the form:

$$\lambda\pi(\mathbf{k}) + \sum_{i=1}^{M} \mu_i 1_{k_i>0}\pi(\mathbf{k}) = \sum_{i=1}^{M} \mu_i 1_{k_i>0}\pi(\mathbf{k}) - \sum_{i=1}^{M}\sum_{j=1}^{M} r_{ji}\mu_j\pi(\mathbf{k}+\mathbf{e}_j-\mathbf{e}_i) + \sum_{i=1}^{M} r_{id}\lambda_i\pi(\mathbf{k}) + \sum_{i=1}^{M}\sum_{j=0}^{M} r_{ji}\mu_j\pi(\mathbf{k}+\mathbf{e}_j-\mathbf{e}_i) \tag{6.14}$$

The second term on the left-hand side and the first term on the right-hand side cancel; so do the second and forth term on the right-hand side. What remains is

$$\lambda\pi(\mathbf{k}) = \sum_{i=1}^{M} r_{id}\lambda_i\pi(\mathbf{k}) = \pi(\mathbf{k})\sum_{i=1}^{M} r_{id}\lambda_i \tag{6.15}$$

which is satisfied, because the streams into and out from the network are equal, $\lambda = \sum_{i=1}^{M} \lambda_i r_{id}$. Thus we have shown that the product form solution stated in Jackson's theorem indeed satisfy the global balance equations of the Markov process describing the network.

6.3 Closed queueing networks

A modification of Jackson's network of queues was considered by Gordon and Newell [34]. The modification they investigated was that of a *closed* Markovian network in the sense that a fixed and finite number of customers, say K, are considered to be in in the system and are trapped in that system in the sense that no other customers may enter and none of these may leave; this corresponds to Jackson's case in which $r_{si} = 0$ and $r_{id} = 0$ for all i.

The customer streams through the different nodes satisfy the conservation law:

$$\lambda_i = \sum_{j=1}^{M} \lambda_j r_{ji}, \qquad i = 1, \ldots, M. \tag{6.16}$$

These constitutes a homogeneous linear set of equations. One equation is linearly dependent on the others, and the solution is determined uniquely up to a constant factor. Let $(\hat{\lambda}_1, \ldots, \hat{\lambda}_M)$ be a solution. The general solution is of the form $\alpha(\hat{\lambda}_1, \ldots, \hat{\lambda}_M)$, where α is a constant that can be determined from mean value analysis.

The equilibrium probabilities of a closed queueing network are:

$$\pi(\mathbf{k}) = \begin{cases} \frac{1}{G(K,M)} \prod_{i=1}^{M} \hat{\rho}_i^{k_i} & \text{when } \sum_i k_i = K \\ 0 & \text{when } \sum_i k_i \neq K \end{cases}$$

where $\hat{\rho}_i = \hat{\lambda}_i/\mu_i$, and the the normalization constant $G(K, M)$ is given by

$$G(K, M) = \sum_{\mathbf{k}:\sum_i k_i = K} \prod_{i=1}^{M} \hat{\rho}_i^{k_i} \tag{6.18}$$

This is the solution to the closed finite queueing network problem, and we observe once again that it has the product form. The proof of is similar to that in open networks. The details are omitted.

6.4 BCMP queueing networks

The theorems of Jacksson and Gordon/Newell have been extended by Baskett, Chandy, Muntz and Palacios [7] to networks with several job classes and different service strategies and interarrival/service time distributions, and also to mixed networks that contain open and closed classes. The networks considered by BCMP must fulfill the following assumptions:

- Type of network: The network is either open, closed or mixed.
- Number of nodes: The network consists of M nodes.
- Number of service classes: An open network is offered traffic from R service classes.
- Queueing discipline: The following disciplines are allowed: FCFS, PS (processor sharing), IS (infinite server), LCFS-PR (Last come first served preemptive resume).
- Description of the service times: The service times of an FCFS node must be exponentially distrivuted and class-independent, while PS, LCFS-PR and IS nodes can have any kind of service time distribution with rational Laplace transform. For the latter three queueing disciplines, the mean service time for different job classes can be different.
- Load-dependent service rate: The service rate of an FCFS node is only allowed to depend on the number of jobs at this node, whereas in a PS, LCFS-PR amd IS node the service rate for a particular job class can also depend on the number of jobs of that class but not on the number of jobs in another class.

- Arrval process: Arriving jobs to an open network are distributed over the nodes in the network in accordance to the probability $r_{si}(c)$ where c denotes the service class.

These assumptions lead to four product-form node types and the local balance conditions for BCMP networks, that is:

- Type-1: -/M/m - FCFS
- Type-2: -/G/1 - PS
- Type-3: -/G/∞ - IS
- Type-4: -/G/1 - LCFS PR

Note that we use the -/M/m notation since we know that, in general, the arrival process to a node in a BCMP network will not be Poisson.

The BCMP theorem says that networks with the characteristics just decribed have product form solution:

$$\pi(\mathbf{k}_1, \ldots, \mathbf{k}_M) = \prod_{i=1}^{M} \pi_i(\mathbf{k}_i) \tag{6.19}$$

where $\mathbf{k}_i = (k_{i1}, \ldots, k_{iR})$ denotes the state of node i.

6.4.1 Closed BCMP queueing networks

For a closed queueing network fulfilling the assumptions of the BCMP theorem, the equilibrium state probabilities have the form:

$$\pi(\mathbf{k}_1, \ldots, \mathbf{k}_M) = \frac{1}{G(\mathbf{K})} \prod_{i=1}^{M} F_i(\mathbf{k}_i) \tag{6.20}$$

where the normalization constant is defined as:

$$G(\mathbf{K}) = \sum_{\sum_{i=1}^{M} \mathbf{k}_i = \mathbf{K}} \prod_{i=1}^{M} F_i(\mathbf{k}_i) \tag{6.21}$$

where $\mathbf{K} = (K_1, \ldots, K_R)$ and K_c denotes the number of jobs in the network from class c, and the function $F_i(\mathbf{k}_i)$ is given by

$$F_i(\mathbf{k}_i) = \begin{cases} k_i! \frac{1}{\beta_i(k_i)} \left(\frac{1}{\mu_i}\right)^{k_i} \prod_{c=1}^{R} \frac{1}{k_{ic}!} e_{ic}^{k_{ic}}, & \text{Type-1} \\ k_i! \prod_{c=1}^{R} \frac{1}{k_{ic}!} \left(\frac{e_{ic}}{\mu_{ic}}\right)^{k_{ic}}, & \text{Type-2,4} \\ \prod_{c=1}^{R} \frac{1}{k_{ic}!} \left(\frac{e_{ic}}{\mu_{ic}}\right)^{k_{ic}}, & \text{Type 3.} \end{cases} \tag{6.22}$$

where $k_i = \sum_{c=1}^{R} k_{ic}$ and $\beta_i(k_i)$ is given by

$$\beta_i(k_i) = \begin{cases} k_i!, & k_i \leq m_i, \\ m_i! m_i^{k_i - m_i}, & k_i \geq m_i, \\ 1, & m_i = 1 \end{cases} \tag{6.23}$$

The mean number of visits e_{ic} of a job of the cth class at the ith node of a closed network are the solution to R sets of linear equations:

$$e_{ic} = \sum_{j=1}^{M} e_{jc} r_{ji}(c), \qquad i = 1, \ldots M, c = 1, \ldots R. \tag{6.24}$$

Each of the R sets contains M linear equations. These equations balance the flows into and flows out of node by each of the R customer classes.

6.4.2 Open BCMP queueing networks

For an open queueing network fulfilling the assumptions of the BCMP theorem and load-independent arrival and service rates, we have

$$\pi(k_1, \ldots, k_M) = \prod_{i=1}^{M} \pi_i(k_i) \tag{6.25}$$

where

$$\pi_i(k_i) = \begin{cases} (1 - \rho_i)\rho_i^{k_i}, & \text{Type-1,2,4}(m_i = 1) \\ e^{-\rho_i} \frac{\rho_i^{k_i}}{k_i!}, & \text{Type-3} \end{cases} \tag{6.26}$$

and

$$\begin{aligned} k_i &= \textstyle\sum_{c=1}^{R} k_{ic} \\ \rho_i &= \textstyle\sum_{c=1}^{R} \rho_{ic} \end{aligned} \tag{6.27}$$

with

$$\rho_{ic} = \begin{cases} \lambda_c \frac{e_{ic}}{\mu_i}, & \text{Type-1}(m_i = 1) \\ \lambda_c \frac{e_{ic}}{\mu_{ic}}, & \text{Type-2,3,4} \end{cases} \tag{6.28}$$

For Type-1 with more that one server ($m_i > 1$) Equation (6.9) can be used to compute the probababilities $\pi_i(k_i)$.

The mean number of visits e_{ic} of a job of the cth class at the ith node of a open network are the solution to R sets of linear equations:

$$e_{ic} = r_{si}(c) + \sum_{j=1}^{M} e_{jc} r_{ji}(c), \qquad i = 1, \ldots M, c = 1, \ldots R. \tag{6.29}$$

6.5 Performance evaluation of queueing networks

6.5.1 Mean value analysis

Although product-form solutions can be expressed very easily as formulae, the computation of state probabilities in a closed queueing network is very time consuming if a straightforward computation of the normalization constant (6.18) is carried out. The Mean Value Analysis (MVA) is an iterative technique, due to Reiser and Lavenberg [71], where the mean values of the performance measures are computed directly without computing the normalization constant.

MVA for closed networks is described as follow. Our objective is to find the mean number of customers $N_i[K]$ and sojourn time $T_i[K]$ as well as the absolute values of customer streams λ_i through different queues. The MVA method is based on two simple laws:

1. *Little's theorem*, which express the relation between the number of jobs, the thoughput, and the mean response time of a node or the overall system: $N = \lambda T$

2. *Arrival theorem*, proven by [71] for all networks that have a product form solution. The arrival theorem says that in a closed network with K customers, the state probabilities seen by a customer entering any node are the same as the equilibrium probabilities $\pi[K-1](\mathbf{k})$ in a network with $K-1$ customers.

The MVA calculation proceeds recursively, incrementing the customer population in the network step by step. Therefore, the total customer population is explicitly indicated in the brackets. The MVA results can be collected as the following recursive method. At the start of recursion, set $N_i[0] = 0$.

The recursion step is:

$$\begin{aligned} T_i[K] &= (1 + N_i[K-1])\tfrac{1}{\mu_i} \\ N_i[K] &= K \tfrac{\hat{\lambda}_i T_i[K]}{\sum_{j=1}^{M} \hat{\lambda}_j T_j[K]} \\ \lambda_i &= \tfrac{N_i[K]}{T[K]} \end{aligned} \tag{6.30}$$

In the second equation, the $\hat{\lambda}_i$'s are any solution to the flow equations $\lambda_i = \sum_j \lambda_j r_{ji}$.

6.5.2 Performance measures

The performance of queueing networks can be evaluated from the equilibrium probabilities $\pi_i(k)$ as follows. The mean number of customers in node i is:

$$N_i = \sum_{k=1}^{\infty} k \pi_i(k) \tag{6.31}$$

The mean sojourn time in node i is obtained from Little's law:

$$T_i = \frac{N_i}{\lambda_i} \tag{6.32}$$

The mean waiting time in node i is:

$$W_i = T_i - \frac{1}{\mu_i} \tag{6.33}$$

The mean time in the network of a customer entering node i is:

$$T_{id} = T_i + \sum_{j=1}^{M} r_{ij} T_{jd}, \qquad i = 1, \ldots, M. \tag{6.34}$$

From this set of equations, the T_{id} can be solved. The mean time in the network of a customer is:

$$T = \frac{N}{\lambda} = \frac{\sum_{i=1}^{M} N_i}{\lambda} \tag{6.35}$$

Chapter 7

Dynamic programming

7.1 Finite planning horizon

Dynamic programming (DP) over a finite control horizon was pioneered by Bellman in the 50's. Today, DP is used in many applications to achieve optimal control of deterministic or stochastic systems over a finite control horizon. Applications include inventory control, machine maintenance, game problems, optimal stopping problems, consumption problems, investment problems, and more.

7.1.1 Notation

We consider a basic optimal control problem, which is amenable to a dynamic solution. Our basic model has two principal features: (1) an underlying *discrete-time dynamic system*, and (2) a *reward function that is additive over time*. The dynamic system takes the form

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots \tag{7.1}$$

where

- k indexes discrete time
- x_k is the state of the system and summarizes past information that is relevant for future optimization
- u_k is the control or decision variable to be selected at time k
- w_k is a random parameter (also called disturbance or noise depending on the context)

- N is the horizon or number of times control is applied

In case the disturbance w_k only can take on single value we have a *deterministic system*, otherwise we have a *stochastic system*.

7.1.2 The basic problem

We are given a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots \tag{7.2}$$

where the state x_k is an element of a state space S_k, the control u_k is an element of a space C_k, and the random "disturbance" w_k is an element of space D_k. The control u_k is constrained to take values in a given nonempty subset $U_k(x_k) \subset C_k$, which depends on the current state x_k; that is, $u_k \in U_k(x_k)$ for all $x_k \in S_k$ and k. The random disturbance w_k is characterized by a probability distribution $P_k(\cdot, x_k, u_k)$ that may depend explicitly on x_k and u_k but no on values of prior disturbances $w_{k-1}, \ldots, w_0$.

We consider the class of policies (also called control laws) that consist of a sequence of functions

$$\pi = \{\mu_0, \ldots, \mu_{N-1}\} \tag{7.3}$$

where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Such policies will be called *admissible*.

Given an initial state x_0 and an admissible policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, the system equation

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots \tag{7.4}$$

makes x_k and w_k random variables with well-defined distributions. Thus for given functions $r_k, k = 0, 1, \ldots, N$, the expected reward

$$J_\pi(x_0) = \underset{\substack{w \\ k=0,1,\ldots,N-1}}{E} \left\{ r_N(x_N) + \sum_{k=0}^{N-1} r_k(x_k, u_k(x_k), w_k) \right\} \tag{7.5}$$

is a well-defined quantity. For a given initial state x_0, and optimal policy π^* is one that maximizes the reward, that is,

$$J_{\pi^*}(x_0) = \max_{\pi \in \Pi} J_\pi(x_0) \tag{7.6}$$

where Π is the set of all admissible policies.

Note that the optimal policy π^* is associated we a fixed initial state x_0. However, an interesting aspect of the basic problem and of dynamic programming is that it is typically possible to find a policy π^* that is simultaneously optimal for all initial states.

The optimal reward depends on x_0 and is denoted $J^*(x_0)$; that is,

$$J^*(x_0) = \max_{\pi \in \Pi} J_\pi(x_0) \tag{7.7}$$

It is useful to view J^* as a function that assigns to each initial state x_0 the optimal reward $J^*(x_0)$ and call it the *optimal reward function* or *optimal value function.*

7.1.3 Principle of optimality

The DP technique rests on a very simple idea, the *principle of optimality*. The name is due to Bellman, who contributed a great deal to the popularization of DP and to its transformation into a systematic tool. Roughly, the principle of optimality states the following rather obvious fact:

> Let $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ be an optimal policy for the basic problem, and assume that when using π^*, a given state x_i occurs at time i with positive probability. Consider the subproblem whereby we are at x_i at time i and wish to maximize the "reward-to-go" from time i to time N
>
> $$E\left\{ r_N(x_N) + \sum_{k=i}^{N-1} r_k(x_k, u_k, w_k) \right\} \tag{7.8}$$
>
> Then the truncated policy $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ is optimal for this subproblem.

7.1.4 DP algorithm

We now state the DP algorithm for the basic problem:

> For every initial state x_0, the optimal reward $J^*(x_0)$ of the basic problem is equal to $J_0(x_0)$, where the function J_0 is given by the last step of the following algorithm, which proceeds backwards from period $N-1$ to period 0:

$$J_N(x_N) = r_N(x_N) \tag{7.9}$$

$$J_k(x_k) = \max_{u_k \in U_k(x_k)} \underset{w_k}{E} \left\{ r_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\}, \quad k = 0, 1, \ldots, N-1, \tag{7.10}$$

where the expectation is taken with respect to the probability distribution of w_k, which depends on x_k and u_k. Furthermore, if $u_k^* = \mu_k^*(x_k)$ maximizes the right side of equation (7.10) for each x_k and k, then policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is optimal.

7.2 Infinite planning horizon

Dynamic programming (DP) for stochastic control with an infinite planning horizon was pioneered by Bellman, Howard and others in the 50's and 60's. Today, DP is used in many applications to achieve optimal control of deterministic and stochastic systems over an infinite planning horizon. Applications are found within machine maintenance, structural inspection, elevator control policies, autonomous robots, behavioral ecology, medical diagnosis, and telecommunications, and more.

DP is used for stochastic control of a Markov decision process (MDP) or a semi-Markov decision process (SMDP). A MDP (SMDP) is a discrete-time (continuous-time) Markov process where transitions from the current Markov state to other Markov states are influenced by a decision or action taken by the controller in the current state. At each state transition a reward is delivered to the controller. The goal of the controller is to select actions that maximizes the accumulated reward.

7.2.1 Markov decision process

Let x_k and u_k denote the state and action at discrete time k, respectively, where x_k is a member of the state space S, and the set of feasible actions at state x is denoted $U(x)$. After selecting action u_k at state x_k, the controller receives an immediate reward r_{k+1}. The environment is specified by the state transition probabilities,

$$p_{xy}(u) = P[x_{k+1} = y | x_k = x, u_k = u], \quad x, y \in S, u \in U(x) \tag{7.11}$$

and expected immediate rewards

$$R(x,u) = E[r_{k+1}|x_k = x, u_k = u], \quad x \in S, u \in U(x) \tag{7.12}$$

For infinite horizon problems, two alternative formulations of the reward maximization objective are common: *average reward* and *discounted reward*. Introducing the (stationary) *policy function* $\pi : S \to U$, the average reward $\overline{R}(\pi)$ under policy π is defined:

$$\overline{R}(\pi) = \lim_{N\to\infty} E\left\{\frac{1}{N}\sum_{k=0}^{N-1} r_{k+1}(x_k, u_k)\right\} \tag{7.13}$$

where $u_k = \mu(x_k)$. The average reward does not depend on the initial state x_0 provided (for example) that at least one state is recurrent. An optimal policy π^* is one that maximizes the average reward, that is,

$$\overline{R}(\pi^*) = \max_{\pi\in\Pi} \overline{R}(\pi) \tag{7.14}$$

where Π is the set of all feasible policies. The optimal average reward is denoted $\overline{R}^*$.

An optimal policy for an MDP can be obtained, in principle, from *Bellman's optimality equation*. This is actually a system of nonlinear equations (using the average-reward formulation):

$$\begin{cases} v^*(x) = \max_{u\in U(x)} \left\{R(x,u) - \overline{R}^* + \sum_{y\in S} p_{xy}(u)v^*(y)\right\}, x \in S \setminus \{x_r\} \\ v^*(x_r) = 0 \end{cases} \tag{7.15}$$

where x_r is an arbitrary chosen reference state, and $v^*(x)$ are *relative values* (also referred to as *differential rewards*) of state $x \in S$ under an optimal policy π^*. The relative values have the following interpretation: The relative value $v(x,\pi)$ under a policy π, is defined as the difference in future reward earnings when starting in state x, compared to starting in the reference state, x_r.

The *discounted reward* has the following definition:

$$V(x,\pi) = \lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1} \gamma^k r_{k+1}(x_k, u_k)|x_0 = x\right\} \tag{7.16}$$

where $V(x,\pi)$ is the *value* of state x under policy π, $u_k = \mu(x_k)$, and $\gamma \in (0,1)$ is a discount factor that makes the sum bounded. An optimal policy π^* maximizes the value function for all states. By selecting γ sufficiently close to 1, a policy that is optimal with respect to average reward is also optimal with respect to discounted reward (and is said to be *Blackwell optimal*).

Bellman's optimality equation for the discounted-reward formulation takes the following form:

$$V^*(x) = \max_{u \in U} \left\{ R(x,u) + \gamma \sum_{y \in S} p_{xy}(u) V^*(y) \right\}, x \in S \tag{7.17}$$

where $V^*(x)$ is the value of state x under an optimal policy π^*.

7.2.2 Semi-Markov decision process

Recalling that an SMDP is a continuous-time version of the MDP, let the expectancy of the time spent in a state x after performing action u be denoted $\tau(x,u)$. The average-reward formulation of Bellman's optimality equation takes the following form:

$$\begin{cases} v^*(x) = \max_{u \in U(x)} \left\{ R(x,u) - \overline{R}^* \tau(x,u) + \sum_{y \in S} p_{xy}(u) v^*(y) \right\}, x \in S \setminus \{x_r\} \\ v^*(x_r) = 0 \end{cases} \tag{7.18}$$

In case of discounted reward, the continuous-time discounting is effectuated by a negative-exponential time window with parameter β ($\beta > 0$). The expected, discounted, immediate reward obtained after selecting action u at state x is

$$R_\beta(x,u) = E \left\{ \int_0^{t_y - t_x} e^{-\beta t} q(t) dt \right\} \tag{7.19}$$

where $q(t)$ is the reward rate at time t, t_x and t_y denote the time of entering state x and the next state y, respectively, and the expectations of $(t_x - t_y)$ and $q(t)$ depend on x and u. In practice $q(t)$ is often constant between state transitions. The Bellman optimality equation is

$$V^*(x) = \max_{u \in U} \left\{ R_\beta(x,u) + e^{-\beta \tau(x,u)} \sum_{y \in S} p_{xy}(u) V^*(y) \right\}, x \in S \tag{7.20}$$

7.2.3 DP algorithms

In this section we outline three DP algorithms known as *Policy iteration*, *Value iteration* and *Linear Programming*. These algorithms apply to both the average reward and discounted reward formulation. However, we choose to describe them for the average reward formulation for SMDPs. In this case the policy iteration algorithm finds the optimal policy along with the optimal relative values and the optimal average reward. Similarly, the value iteration algorithm find the optimal policy along with the optimal values. The linear programming algorithm finds just the optimal policy.

Policy iteration

The policy iteration algorithm contains four steps:

1. Model identification
2. Policy evaluation
3. Policy improvement
4. Convergence test

Before the policy iteration can start, the policy iteration counter k is set to 0, and an initial policy π_0 is arbitrarily selected. Then the first step is to identify the SMDP; that is, the state transition probabilities and the expected immediate rewards under the current policy.

The second step, the relative value function and the average reward are computed. An efficient way of doing this is to solve a system of linear equations:

$$\begin{cases} v(x,\pi_k) = R(x,u) - \overline{R}(\pi_k)\tau(x,u) + \sum_{y\in S} p_{xy}(u)v(y,\pi_k), x \in S \setminus \{x_r\} \\ v(x_r,\pi_k) = 0 \end{cases} \tag{7.21}$$

where $u = \mu(x)$. Finally, the policy is improved by traversing the actions for each state, and setting the policy to the actions that results in the highest relative values:

$$u = \arg\max_{a\in U(x)} \left\{ R(x,a) - \overline{R}(\pi_k)\tau(x,a) + \sum_{y\in S} p_{xy}(a)v(y,\pi_k) \right\}, x \in S \setminus \{x_r\} \tag{7.22}$$

The forth step is the convergence step: stop the iteration if $\pi_{k+1} = \pi_k$ where π_{k+1} is the new improved policy, and π_k is the current policy. Otherwise, let $k := k + 1$ and go to step 1.

An assumption for the policy iteration algorithm to work is that in each iteration the underlying Markov chain is irreducible.

Value iteration

The value iteration algorithm contains three steps:

1. Model identification

2. Recursion step

3. Convergence test

Before the value iteration can start, an initial policy π_0 and initial values $V_0(x, \pi_0), x \in S$ are arbitrarily selected. Then the first step is to identify the SMDP.

The second step is the recursion step:

$$V_n(x, \pi_k) = \tilde{R}(x, u) - \overline{R}(\pi_k)\tau(x, u) + \sum_{y \in S} \tilde{p}_{xy}(u) V_{n-1}(y, \pi_k), x \in S \tag{7.23}$$

where $u = \mu(x)$, $V_n(x, \pi)$ denotes the n-step value function, and the immediate reward $\tilde{R}(x, u)$ and transition probabilities $\tilde{p}_{xy}(u)$ are computed by *uniformization* which transforms the continuous-time SMDP to a discrete-time MDP:

$$\begin{aligned} \tilde{R}(x, u) &= \tfrac{1}{\tau(x,u)} R(x, u), && x \in S, u \in U(x) \\ \tilde{p}_{xy}(u) &= \begin{cases} \tfrac{\tau}{\tau(x,u)} p_{xy}(u), & y \neq x, x \in S, u \in U(x) \\ \tfrac{\tau}{\tau(x,u)} p_{xy}(u) + [1 - \tfrac{\tau}{\tau(x,u)}], & y = x, x \in S, u \in U(x) \end{cases} \end{aligned} \tag{7.24}$$

where τ is the size of the discrete time step, chosen such that $0 < \tau \leq \min_{x,u} \tau(x, u)$.

The third step is the convergence test. The iteration is stopped if $0 \leq M_n - m_n \leq \epsilon m_n$, where ϵ is a predetermined error. Otherwise, let $n := n + 1$ and go to 1. The convergence test makes use of the following variables:

$$\begin{aligned} m_n &= \min_{x \in S} \Delta V_n(x, \pi_k) \\ M_n &= \max_{x \in S} \Delta V_n(x, \pi_k) \\ \Delta V_n(x, \pi_k) &= V_n(x, \pi_k) - V_{n-1}(x, \pi_k) \end{aligned} \tag{7.25}$$

An assumption for the value iteration algorithm to work is that in exach iteration the underlying Markov chain is irreducible.

Linear programming

The linear programming algorithm is:

$$\begin{array}{ll} \max & \sum_{x\in S}\sum_{u\in U(x)} R(x,u)z_{xu} \\ \text{s.t.} & \sum_{x\in S}\sum_{u\in U(x)} \tau(x,u)z_{xu} = 1 \\ & \sum_{u\in U(y)} z_{yu} = \sum_{x\in S}\sum_{u\in U(x)} p_{xy}(u)z_{xu}, y \in S, \\ & z_{xu} \geq 0, x \in S, u \in U(x). \end{array} \tag{7.26}$$

Let z^* be an optimal extreme point for the above LP (obtained, for example, by the simplex method). Then for each $x \in S$, there will be at most one $u \in U(x)$ such that $z^*_{xu} > 0$; call this action $g(x)$. If $z^*_{xu} = 0$ for all $u \in U(x)$, then set $g(x)$ to an arbitrary element of $U(x)$. The policy $g = [g(x) : x \in S]$, which chooses action $g(x)$ when ever the state is x, maximizes over all policies the long-run average reward.

7.3 Applications

7.3.1 Call admission control

Call admission control (CAC) is an important function in multi-service communication networks. In this section we describe the semi-Markov decision process (SMDP) model for CAC on a single link. The single link CAC problem is part of the end-to-end CAC and routing problem. The objective for end-to-end CAC and routing is to maintain the QoS and GoS for all existing calls and call classes, and to maximize the revenue for the network operator. To obtain a manageable complexity, the network Markov process is decomposed into a set of independent link Markov processes. The state on each link is assumed to evolve independently of other links. However, in a real network, the presence of multi-link calls introduces positive correlation between states on successive links.

We formulate the CAC task as a reward maximization problem as follows. The link is offered call requests from K traffic classes. Class $j \in J = \{1, \ldots, K\}$ is characterized by:

- bit rate requirement b_j [bandwidth units],
- Poissonian call arrival process with rate λ_j [s^{-1}],
- exponentially distributed call holding time with mean $1/\mu_j$ [s],
- reward parameter $r_j \in (0, \infty)$.

A SMDP can be described as a tuple $< S, U, p, R, \tau >$, where

- S is a finite set of states,
- U is a finite set of actions,
- $p : S \times U \rightarrow \Pi(S)$ is the state-transition function, where a member of $\Pi(S)$ is a probability distribution over the set S,
- $R : S \times U \rightarrow \mathbb{R}$ is the reward function, giving the expected immediate reward gained by the agent,
- $\tau : S \times U \rightarrow \mathbb{R}$ is the mean state sojourn time function, giving the mean time duration in each state until a state transition.

The state space S is given by:

$$S = \left\{ \mathbf{x} = \{x_j\} : \sum_{j \in J} b_j x_j \leq C \right\}, \tag{7.27}$$

where C denotes the capacity of the link.

The Markov decision action u is represented by a vector $\mathbf{u} = \{u_j\}, j \in J$, corresponding to admission decisions for presumptive call requests. The action space is given by:

$$U = \{\mathbf{u} = \{u_j\} : u_j \in \{0, 1\}, j \in J\}, \tag{7.28}$$

where $u_j = 0$ denotes call rejection and $u_j = 1$ denotes call acceptance. The permissible action space is a state-dependent subset of U:

$$U(\mathbf{x}) = \{\mathbf{u} \in U : u_j = 0 \text{ if } \mathbf{x} + \delta_j \notin S, j \in J\} \tag{7.29}$$

where δ_j denotes a vector of zeros except a one in position $j \in J$.

The Markov chain is characterized by state transition probabilities $p_{\mathbf{xy}}(\mathbf{u})$ which express the probability that the next state is $\mathbf{y}$, given that action $\mathbf{u}$ is taken in state $\mathbf{x}$. In our case, the state transition probabilities become:

$$p_{\mathbf{xy}}(\mathbf{u}) = \begin{cases} \lambda_j u_j \tau(\mathbf{x}, \mathbf{u}), & \mathbf{y} = \mathbf{x} + \delta_j \in S, j \in J \\ x_j \mu_j \tau(\mathbf{x}, \mathbf{u}), & \mathbf{y} = \mathbf{x} - \delta_j \in S, j \in J \\ 0, & \text{otherwise}, \end{cases} \tag{7.30}$$

where $\tau(\mathbf{x}, \mathbf{u})$ denotes the average sojourn time in state $\mathbf{x}$.

The expected reward in state $\mathbf{x}$ is given by $R(\mathbf{x}, \mathbf{u}) = q(\mathbf{x})\tau(\mathbf{x}, \mathbf{u})$, where $q(\mathbf{x})$ is obtained from

$$q(\mathbf{x}) = \sum_{j \in J} r_j x_j \mu_j. \tag{7.31}$$

The average sojourn time in state $\mathbf{x}$ is given by

$$\tau(\mathbf{x}, \mathbf{u}) = \left\{ \sum_{j \in J} x_j \mu_j + u_j \lambda_j \right\}^{-1}. \tag{7.32}$$

We have tested the above SMDP model on the following example. $K = 2, C = 24, (b_1, b_2) = (1, 6), (\mu_1, \mu_2) = (1, 1), (r_1, r_2) = (1, 6)$; λ_1 and λ_2 were varied so that

$$\rho = \frac{b_1 \lambda_1}{C \mu_1} + \frac{b_2 \lambda_2}{C \mu_2} = 1.5. \tag{7.33}$$

Figure 7.1 compares the average reward rate of the optimal policy to that of Complete Sharing (CS). The CS policy always accepts a new call provided there is sufficient free capacity on the link. Figure 7.2 shows the per-class blocking probabilities. The following observation were made:

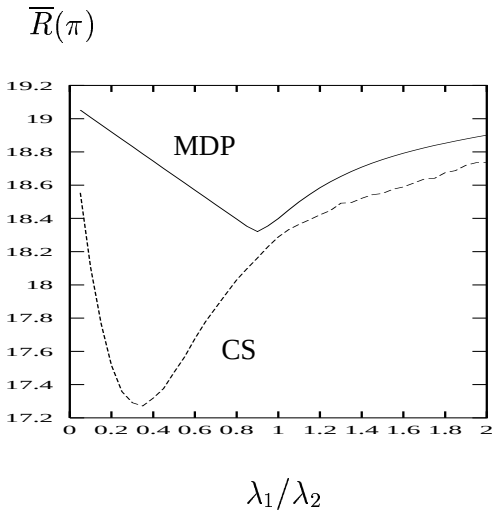


Figure 7.1: Average reward versus traffic ratio

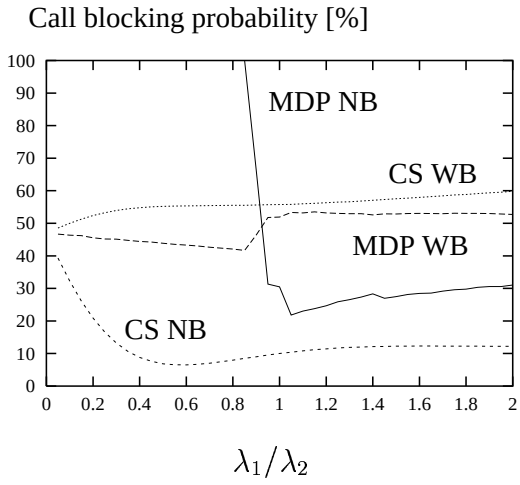


Figure 7.2: Per-class call blocking probability versus traffic ratio

1. The optimal policy is not in general of the threshold type, where a narrow-band (class-1) call is accepted if and only if the free capacity is below a fixed threshold. A specific counterexample

is given by $C = 24, (b_1, b_2) = (1, 6), (r_1, r_2) = (1, 6), \lambda_1 = \lambda_2 = 4, \mu_1 = \mu_2 = 1$ where, under the optimal policy, a narrow-band call is accepted in state (18,0) and (13,1), but is blocked in (6,2) and (0,3). Note that in all of these states the free capacity is 6 units. An explanation of this is as follows. In states (19,0) and (13,1), the time until next departure (which would give sufficient room for a wide-band (class-2) call) is exponentially distributed with rate 19 and 14, respectively. But in state (7,2) and (1,3) the rates are 9 and 4, respectively. Thus, the risk of blocking a future wide-band call in state (7,2) and (1,3) is higher than in states (19,0) and (13,1). Therefore, the optimal policy blocks a narrow-band call in states (6,2) and (0,3) thereby preventing the system from moving into a high-risk state.

2. For small and large values of the arrival rate ratio, there is little difference in performance between the optimal policy and CS; for moderate values of λ_1/λ_2, the improvement can be close to 10 percent.

3. The MDP policy depends on the arrival rate ratio.

4. The CS policy has lower NB blocking probability than the MDP policy.

5. The MDP policy blocks NB calls in all link states when $\lambda_1/\lambda_2 < 0.8$.

6. The CS policy has higher WB blocking probability than the MDP policy.

Chapter 8

Game theory

8.1 Introduction

8.1.1 What is game theory?

Game theory is the formal study of conflict and cooperation [60]. Game theoretic concepts apply whenever the actions of several agents are interdependent. These agents may be individuals, groups, firms, or any combinations of these. The concepts of game theory provide a language to formulate, structure, analyze, and understand strategic scenarios.

8.1.2 History and impact of game theory

The mathematician Emile Borel suggested a formal theory of games in 1921, which was furthered by the mathematician John von Neuman in 1928. Game theory was established as a field in its own right after the 1944 publication of the monumental volume *Theory of Games and Economic Behavior* by von Neuman and the economist Oskar Morgenstern. The book provided much of the basic terminology and the problem setup that is still in use today.

In 1950, John Nash demonstrated that finite games always have an equilibrium point, at which all players chose action which are best for them given their opponents' choices. The central concept of non-cooperative game theory has been the focal point of analysis since then. In the 1950s and 1960s, game theory was broadened theoretically and applied to problems of war and politics. Since the 1970s, it has driven a revolution in economic theory. Additionally, it has found applications in sociology and psychology, and established links with evolution and biology. Game theory received special attention

in 1994 with the awarding of the Nobel prize in economics to Nash, John Harsanyi, and Reinhard Selten.

At the end of the 1990s, a high-profile application of game theory has been the design of auctions. Prominent game theorists have been involved in the design of auctions for allocating rights to use the bands of the electromagnetic spectrum to the mobile telecommunications industry. Most of these auctions were designed with the goal of allocating these resources more efficiently than traditional governmental practices.

8.1.3 Definition of games

The object of study in game theory is the *game*, which is a formal model of an interactive situation. It typically involves several *players*; a game with only one player is usually called a *decision problem*. The formal definition lays out the players, their preferences, their information, the strategic actions available to them, and how these influence the outcome.

In game theory, we generally assume players to be rational and intelligent. A decision maker is *rational* if he makes decisions consistently in pursuit of his own objectives. In game theory, building on the fundamental results of decision theory, we assume that each player's objective is to maximize the expected value of his own payoff, which is measured some *utility scale*. A player is *intelligent* if he knows every that we know about the game and that he can make any inferences about the situation that we can make.

The goal of game-theoretic analysis is to predict how the game will be played, or, relatedly, to give advise on how best to play the game against opponents who are rational and intelligent.

All games can be classified as complete information games or incomplete information games:

- *Complete information game*: the player whose turn it is to move knows at least as much as those who moved before him/her.
- *Incomplete information game*: at some node in the game the player whose turn it is to make a choice knows less than a player who has already moved.

All games can be classified into perfect information games or imperfect information games:

- *Perfect information game*: players know the full history of the game, all moves made by all players etc and all payoffs.

- *Imperfect information game*: games involving simultaneous move where players know all the possible outcomes/payoffs, but not the actions chosen by other players.

In other words in a game with imperfect information, players are simply unaware of the actions chosen by other players. However, they know who the other players are, what their strategies/actions are, and the preferences/payoffs of the other players. Hence, the information about these other players in imperfect information is complete. On the other hand, in a game with incomplete information, players may or may not know some information about other players, e.g. their 'type', their strategies, payoffs or preferences.

Games can be classified into simultaneous or sequential games:

- *simultaneous game*: all the players move together/or play their strategy at the same time.
- *sequential game*: all players move one after another; the order of movement is important.

Games can be classified into one-shot or repeated games:

- *one-shot game*: the game is only played only one time,
- *repeated game*: the game is repeated finitely or infinitely many times.

Finally, games can be classified into cooperative and non-cooperative games:

- *cooperative games*: players enter coalitions who act selfishly in a rational and intelligent way. The members of a coalition cooperate to increase the members' payoffs.
- *non-cooperative games*: players act selfishly in a rational and intelligent way.

8.1.4 Equilibrium concepts

Table 8.1 shows the various equilibrium notions that have been proposed to characterize different kinds of games. Selten provided a strengthening of the Nash equilibrium concept that would rule out equilibria sustained by incredible threats [73]. Games of incomplete information are also called *Bayesian Games* since some of the nodes in these games are owned by *Chance* or *Nature*. The actions made by chance are not directly observed by the players and hence the name. A mixed strategy Nash equilibrium of such games is called Bayes-Nash equilibrium, and was introduced by Harsani [35]. A second, stronger refinement of the equilibrium was introduced by Keeps and Wilson based on the concept of sequential rationality [51].

	Information Type	
Game Type	Complete	Incomplete
Static	Nash Equilibrium (Nash, 1951)	Bayes-Nash Equilibrium (Harsani, 1968)
Dynamic	Subgame Perfect Equilibrium (Selten, 1975)	Perfect Bayesian Equilibrium Sequential Equilibrium (Kreps and Wilson, 1982)

Table 8.1: Equilibrium concepts

8.2 Games in extensive form

The extensive form is a complete description of how the game is played over time. This includes the order which the players take actions, the information that the players have at the time they must take those actions, and the times at which any uncertainty in the situation is resolved. A game in extensive form may be analyzed directly, or can be converted into an equivalent strategic form.

To introduce the extensive form, let us consider a simple card game that is played by two people, whom we call "player 1" and "player 2". At the beginning of this game, players 1 and 2 each put a dollar in the pot. Next, player 1 draws a card from a shuffled deck in which half the cards are red (diamonds and hearts) and half are black (clubs and spades). Player 1 looks at his card privately and decides whether to raise or fold. If player 1 folds he shows his card to player 2 and game ends; in this case player 1 takes the money from the pot if the card is red, but player 2 takes the money from the pot if the card is black. If player 1 raises he adds another dollar to the pot and player 2 must decide whether to meet or pass. If player 2 passes, then the game ends and player 1 takes the money in the pot. If player 2 meets, then she must also put another dollar to the pot, and then player 1 shows the card to player 2 and the game ends; in this case, again player 1 takes the money in the pot if the card is red and player 2 takes the money in the pot if the card is black. Figure 8.1 shows a tree diagram for this simple card game in extensive form. The meaning of the notation in the tree digram will be explained by the following definition.

Formally, for any positive integer n, an *n-person extensive-form game* Γ^e is a rooted tree together with functions that assign labels to every node and branch satisfying the following properties:

1. Each nonterminal node has a *player label* that is in the set $\{0, 1, 2, \ldots, n\}$. Nodes that are

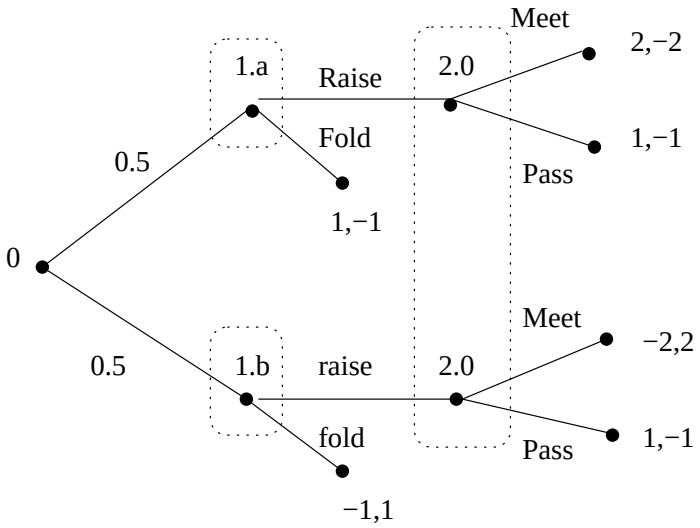


Figure 8.1: Simple card game in extensive form.

assigned a player label 0 are called *chance nodes*. The set $\{1, 2, \ldots, n\}$ represents the set of *players* in the game, and, for each i in this set, the nodes with the player-label i are *decision nodes* that are controlled by player i.

2. Every alternative at a chance node has a label that specifies its probability. At each chance node, these *chance probabilities* of the alternatives are nonnegative numbers that sum to 1.

3. Every node that is controlled by a player has a second label that specifies the *information state* that the player would have if the path of play reached this node. When the path of play reaches a node controlled by a player, he knows only the information state of the current node. That is, two nodes that belong to the same player should have the same information state if and only if the player would be unable to distinguish between situations represented by these nodes when either occurs in the play of the game. In our notation, the player label and the information label at any node are separated by a decimal point, with the player label to the left and the information label to the right, so "$i.k$" would indicate a node where player i moves with information state k.

4. Each alternative at a node that is controlled by a player has a *move label*. Furthermore, for any two nodes x and y that have the same player label and the same information label, and for any alternative at node x, there must be exactly one alternative at node y that has the same move label.

5. Each terminal node has a label that specifies a vector of n numbers $(u_1, \ldots, u_n) = (u_i)_{i\in\{1,\ldots,n\}}$. For each player i, the number u_i is interpreted as the *payoff* to player i, measured in some utility scale, when this node is the outcome of the game.

6. Whenever a player moves, he remembers all the information that he knew earlier in the game, including all of his own past moves (known as *perfect recall*).

   A *strategy* for a player in an extensive-form game is any rule for determining a move at every possible information state in the game. In our simple card game, player 1 has four possible strategies. We denote the set of strategies for player 1 in this game as { Rr,Rf,Fr,Ff }, where we write first in upper case the initial letter of the designated move at 1.a node (with a red card), and the second in lower case the initial letter of the designated move at the 1.b node (with a black card). For example, Rf denotes the strategy "Raise if the card is red, but fold if the card is black," and Rr the strategy "Raise no matter what the color of the card may be." Notice that a strategy for player 1 is a complete rule that specifies a move for player 1 in all possible contingencies, even though only one contingency will actually arise. Player 2 has only two possible strategies M and P because player 2 has only one possible information state. Here M denotes "Meet if player 1 raises" and P denotes "Pass if player 1 raises"

8.3 Games in strategic form

The strategic form (also called normal form) is the basic type of game studied in non-cooperative game theory. To define a game in strategic form, we need only to specify the set of players in the game, the set of options available to each player, and the way the players payoff depend on the options they choose. Formally, a *strategic form* game is any Γ of the form

$$\Gamma = (N, (C_i)_{i\in N}, (u_i)_{i\in N}) \tag{8.1}$$

where N is a nonempty set, and, for each i in N, C_i is a nonempty set and u_i is a function from $\times_{j\in N} C_j$ into the set of real numbers $\mathbb{R}$. Here, N is the set of players in the game Γ. For each player i, C_i is the set of *strategies* available to player i. When the strategic-form game Γ is played, each player i must choose one of the strategies in the set C_i. A *strategy profile* is a combination of strategies that the players in N might choose. We let C denote the set of all possible strategy profiles, so that

$$C = \times_{j \in N} C_j \tag{8.2}$$

For any strategy profile $c = (c_j)_{j \in N}$ in C, the number $u_i(c)$ represents the expected utility payoff that player i would get in this game if c were the the combination of strategies implemented by the players. When we study a strategic form game, we usually assume that the players all choose their strategies simultaneously, so there is no element of time in the analysis of strategic-form games.

A strategic form game is *finite* if the set of players N and all the strategy sets C_i are finite. In developing the basic ideas of game theory, we generally assume finiteness.

Von Neuman and Morgenstern suggested a procedure for constructing a game in strategic form, given any extensive-form game Γ^e. To illustrate this procedure, consider again the simple card game, shown in Figure 8.1. Now suppose the players 1 and 2 know that they are going to play this game tomorrow, but today each player is planning his or her moves in advance. Player 1 does not know today what color his card will be, but he can plan now what he would do with a red card, and what he would do with a black card. That is, as we have seen, the set of possible strategies for player 1 in this extensive-form game is C_1={ Rr,Rf,Fr,Ff }, where the first letter designates his move if his card is red (at the node labeled 1.a) and the second letter designates his move if the card is black (at the node labeled 1.b). Player 2 does not know today whether player 1 will raise or fold, but she can plan today whether to meet or pass if 1 raises. So the set of strategies player 2 can choose among is C_2={ M,P }, where M denotes the strategy "meet if player 1 raises" and P denotes the strategy "pass is player 1 raises."

Even if we knew the strategy that each player plans to use, we still could not predict the actual outcome of the game, because we do not know whether the card will be red or black. However, we can compute the expected payoff to each player when these strategies are used in the game, because we know that the red and black cards each have probability 1/2. So when player 1 plans to use the strategy Rf and player 2 plans to use the strategy M, the expected payoff to player 1 is

$$u_1(\text{Rf,M}) = 2 \times 1/2 + (-1) \times 1/2 = 0.5 \tag{8.3}$$

Similarly, player 2's expected payoff from the strategy profile (Rf,M) is

$$u_2(\text{Rf,M}) = (-2) \times 1/2 + 1 \times 1/2 = -0.5 \tag{8.4}$$

	C_2	
C_1	M	P
Rr	0,0	1,-1
Rf	0.5,-0.5	0,0
Fr	-0.5, 0.5	1,-1
Ff	0,0	0,0

Table 8.2: The simple card game in strategic form, the normal representation

The strategy-form game shown in Table 8.2 is called the *normal representation* of our simple card game. It describes how, at the beginning of the game, the expected utility payoff to each player would depend on their strategic plans.

8.4 Static non-cooperative games

8.4.1 Elimination of dominated strategies

For any player i, let C_{-i} denote the set of all possible combinations of strategies for the players other than i; that is,

$$C_{-i} = \times_{j \in N \setminus \{i\}} C_j$$

(Here, $N \setminus \{i\}$ denotes the set of all players other than i.) Given any $e_{-i} = (e_j)_{j \in N \setminus \{i\}}$ in C_{-i} and any d_i in C_i, we let (e_{-i}, d_i) denote the strategy profile in C such that the i-component is d_i and all other components are as in e_{-i}.

For any set Z and any function $f : Z \to \mathbb{R}$, $\arg\max_{y \in Z} f(y)$ denotes the set of points in Z that maximize the function f, so

$$\arg\max_{y \in Z} f(y) = \{y \in Z | f(y) = \max_{z \in Z} f(z)\}$$

If player i believed that some distribution η in $\Delta(C_{-i})$ predicted the behavior of the other players in the game, so each strategy combination e_{-i} in C_{-i} would have probability $\eta(e_{-i})$ of being chosen

by the other players, then player i would want to choose his own strategy in C_i to maximize his own expected utility payoff. So in game Γ, player i's set of *best responses* to η would be

$$\arg\max_{d_i \in C_i} \sum_{e_{-i} \in C_{-i}} \eta(e_{-i}) u(e_{-i}, d_i) \tag{8.5}$$

We let $\Delta(Z)$ denote, for any finite set Z, the set of all probability distributions over the set Z.

In general, given any strategic-form game $\Gamma = (N, (C_i)_{i \in N}, (u_i)_{i \in N})$, a *randomized strategy* for player i is an probability distribution over the set of C_i. Thus $\Delta(C_i)$ denotes the set of randomized strategies for player i. To emphasize the distinction from these randomized strategies, the strategies in the set C_i may also be called *pure strategies*. For any pure strategy c_i in C_i and any randomized strategy σ_i in $\Delta(C_i)$, $\sigma_i(c_i)$ denotes the randomized strategy σ_i in $\Delta(C_i)$.

The highest expected utility that player i can get against any combination of other players' randomized strategies does not depend on whether i himself uses randomized strategies or only pure strategies. Furthermore, the optimal randomized strategies for i are just those that assign positive probabilities to his optimal pure strategies.

Any strategy d_i is *strongly dominated* for player i if and only if there exists some randomized strategy σ_i in $\Delta(C_i)$ such that

$$u_i(c_{-i}, d_i) < \sum_{e_i \in C_i} \sigma_i(e_i) u_i(c_{-i}, e_i), \forall c_{-i} \in C_{-i}$$

where $C_{-i} = \times_{j \in N \setminus \{i\}} C_j$ and $c_{-i} = (c_j)_{j \in N \setminus \{i\}}$. For example, in the strategic form of our simple card game (Table 8.2), the strategy Ff is strongly dominated for player 1 by the randomized strategy 0.5[Rr]+0.5[Rf].

Note that d_i is strongly dominated for player i if and only if d_i can never be a best response for i, no matter what he may believe about the other players' strategies. This in fact suggest that eliminating a strongly dominated strategy for any player i should not affect the analysis of the game, because player i would never use this strategy, and this fact should be evident to all the players if they are intelligent.

After one or more strongly dominated strategies have been eliminated from the game, other strategies that were not strongly dominated in original game may become strongly dominated in the game that remains. For example, consider the game in Table 8.3. In this game, z_2 is strongly dominated for

	C_2		
C_1	x_2	y_2	z_2
a_1	2,3	3,0	0,1
b_1	0,0	1,6	4,2

Table 8.3: A game in strategic form

player 2 by $0.5[x_2] + 0.5[y_2]$ (the randomized strategy that gives probabilities 0.5 to x_2 and y_2 each). None of the other strategies for either player are strongly dominated in this game, because each is a best response to at least one conjecture about how the other player may behave. (Strategy a_1 is best for 1 against x_2, b_1 is best for 1 against z_2, x_2 is best for 2 against a_1, and y_2 is best for 2 against b_1.) However, in the game that remains after eliminating z_2, b_1 is strongly dominated by a_1 for player 1 (because $0 < 2$ and $1 < 3$). Furthermore, when z_2 and b_1 both are eliminated, we are left with a game in which y_2 is strongly dominated by x_2 for player 2 (because 0 ¡ 3). Then eliminating y_2 leaves only one strategy for each player in the game: a_1 for player 1 and x_2 for player 2. Thus iterative elimination of strongly dominated strategies leads to a unique prediction as to what players should do in this game.

Any strategy d_i is *weakly dominated* for player i if and only if there exists some randomized strategy σ_i in $\Delta(C_i)$ such that

$$u_i(c_{-i}, d_i) \leq \sum_{e_i \in C_i} \sigma_i(e_i) u_i(c_{-i}, e_i), \forall c_{-i} \in C_{-i}$$

and for at least one strategy combination $\hat{c}_{-i}$ in C_{-i},

$$u_i(\hat{c}_{-i}, d_i) < \sum_{e_i \in C_i} \sigma_i(e_i) u_i(\hat{c}_{-i}, e_i). \tag{8.6}$$

It is harder to argue that eliminating a weakly dominated strategy should not affect the analysis of the game because weakly dominated strategies could be best responses for a player, if he feels confident that some strategies of other players have probability 0. Nevertheless, weak domination and iterative elimination if weakly dominated strategies are useful concepts for analysis of games. In our simple card game, the fact that Fr and Ff are both weakly dominated for player 1 is a formal expression of our natural intuition that player 1 should not fold when he has a winning card.

8.4.2 Nash equilibrium

In the previous examples, consideration of dominating strategies alone yielded precise advice on how to play the game. In many games, however, there are no dominated strategies, as so these considerations are not enough to rule out any outcomes or to provide specific advise on how to play the game.

The central concept of *Nash equilibrium* is much more general. A Nash equilibrium recommends a strategy to each player that the player cannot improve upon *unilaterally*, that is, given that the other players follow the recommendation. Since other players also are rational, it is reasonable for each player to expect his opponents to follow the recommendation as well.

We say that a pure-strategy profile c in C is a Nash equilibrium if and only iff

$$u_i(c) \geq u_i(c_{-i}, d_i), \forall i \in N, \forall d_i \in C_i.$$

The following general equilibrium existence theorem is due to Nash (1951).

Theorem

Given any finite game Γ*, there exists at least one equilibrium in* $\times_{i \in N} \Delta(C_i)$.

Consider our simple card game. It is straightforward to verify that there are no equilibrium in pure strategies in this game (Just check all eight possibilities). Thus, there must be an equilibrium that uses properly randomized strategies. To find these equilibrium, notice first that the pure strategy Ff is strongly dominated and Fr is weakly dominated in this game. It can be shown that any equilibrium of the residual game generated by iterative elimination of weakly and strongly dominated strategies is also an equilibrium of the original game. Thus, we can expect to find an equilibrium that involves randomization between Rr and Rf and between M and P.

So let q[Rr]+$(1-q)$[Rf] and s[M]+$(1-s)$[P] denote the equilibrium strategies for player 1 and 2, where q and s are some numbers between 0 and 1. That is, let q denote the probability that player 1 would raise even with a losing card, and let s denote the probability that player 2 would meet if player 1 raised. Player 1 would be willing to randomize between Rr and Rf only if Rr and Rf give him the same expected utility against s[M]+$(1-s)$[P], so

$$0s + 1(1-s) = 0.5s + 0(1-s)$$

which implies that $s = 2/3$. Thus player 2's strategy in the equilibrium must be 2/3[M]+1/3[P] to make player 1 willing to randomize between [Rr] and [Rf]. Similarly, to make player 2 willing to randomize between M and P, M and P must give he the same expected utility against q[Rr]+$(1-q)$[Rf], so

$$0q + (-0.5)(1-q) = -1q + 0(1-q)$$

which implies that $q = 1/3$. Thus the equilibrium must be

$$(1/3[\text{Rr}] + 2/3[\text{Rf}], 2/3[\text{M}] + 1/3[\text{P}])$$

That is, player 1 raises for sure when he has a winning (red) card, player 1 raises with probability 1/3 when he has a losing (black) card, and player 2 meets with probability 2/3 when she sees player 1 raise in this equilibrium. In fact, this randomized-strategy profile is the unique equilibrium of the simple card game.

Two general observations about Nash equilibria are now in order. A game may have equilibria that are *inefficient*, and a game may have *multiple equilibria*.

An outcome of a game is (*weakly*) *Pareto efficient* if and only iff there is no other outcome that would make all players better off. For an example of equilibria that are not efficient, consider the game in Table 8.4 known as *Prisoner's Dilemma*. The story behind this game (taken from Luce and Raiffa, 1957) may be described as follows. The two players are accused of conspiring in two crimes, one minor crime for which their guilt can be proved without any confession, and one major crime for which they they can be convicted only if at least on confesses. The prosecutor promises that, if exactly one confesses, the confessor will go free now but the other will go to jail for 6 years. If both confesses, then they both go to jail for 5 years. If neither confesses they will both go to jail for only 1 year. So each player i has two possible strategies: to confess (f_i) or not to confess (g_i). The payoffs, measured in the number of years of freedom that the player will enjoy over the next 6 years, are shown in Table 8.4.

In this game, $([f_1], [f_2])$ is the unique Nash equilibrium. (In fact, f_1 and f_2 are the only two strategies that are not strongly dominated.) However, the outcome resulting from $(f_1 f_2)$ is the only outcome of the game that is not Pareto efficient. If Nash equilibria can be interpreted as describing how rational players should play a game, then rational individuals should expect to all do relatively

	C_2	
C_1	g_2	f_2
g_1	5,5	0,6
f_1	6,0	1,1

Table 8.4: Prisoner's Dilemma game in strategic form

badly in this game. This example has been very influential as a simple illustration of how people's rational pursuit of their individual best interests can lead to outcomes that are bad for all of them.

For an example of game with multiple equilibria, consider the game in Table 8.5, known as *Battle of the sexes*. The story behind this game (again, taken from Luce and Raiffa, 1957) is that the players 1 and 2 are husband and wife, respectively, and that they are deciding where to go on Saturday afternoon. Each f_i strategy represents going to the football match, whereas s_i represents going to the shopping center. Neither spouse would derive any pleasure from being without the other, but the husband would prefer meeting at the football match, whereas the wife would prefer meeting at the shopping center.

There are three equilibria of this game: $([f_1], [f_2])$ which gives payoffs allocation (3,1) to players 1 and respectively, $([s_1], [s_2])$, which gives payoff allocation (1,3); and

$$(0.75[f_1] + 0.25[s_1], 0.25[f_2] + 0.75[s_2])$$

which gives each player an expected payoff of 0.75. In the first equilibrium, the players go both to the football match, which which is player 1's favorite outcome. In the second equilibrium, the players go both the shopping center, which is player 2's favorite outcome. So each player prefers a different equilibrium.

In the third equilibrium the players behave in a random and uncoordinated manner, which neither player can unilaterally improve on. Each player is uncertain about where the other player will go, and gets the same expected payoff ($0.75 \times 1 = 0.25 \times 3$) from going either way. The third equilibrium is worse for both players than either of the other two equilibria, so it is also an inefficient equilibrium.

In a Nash equilibrium, two different pure strategies of player i both have positive probability, then they most both give him the same expected payoff in the equilibrium, because otherwise he would never use the strategy that gave him a lower expected payoff. That is, in equilibrium, a player must be indifferent between any of his strategies that have positive probability in this randomized strategy.

	C_2	
C_1	f_2	s_2
f_1	3,1	0,0
s_1	0,0	1,3

Table 8.5: Battle of the sexes game in strategic form

Thus, the set of strategies that have positive probability is an important qualitative aspect of an equilibrium. In general, the *support* of a randomized-strategy profile σ in $\times_{i \in N} \Delta(C_i)$ is is the set of all pure-strategy profiles in C that would have positive probability if the payers chose their strategies according to σ. That is, the support of σ is the set

$$\times_{i \in N} \{c_i \in C_i | \sigma_i(c_i) > 0\}$$

For example, consider again the Battle of the sexes game. The support of the ([f_1],[f_2]) equilibrium is obviously $\{f_1\} \times \{f_2\}$. In this equilibrium each player strictly prefers using the f_i strategy to the s_i strategy, given that the other player is expected to go to the football match. Similarly, the support of the ([s_1],[s_2]) equilibrium is obviously $\{s_1\} \times \{s_2\}$; and in this equilibrium each player i strictly prefers s_i to f_i, given that the other player is expected to go shopping.

As we have seen, this game also has a third equilibrium with support $\{f_1, s_1\} \times \{f_2, s_2\}$. In an equilibrium with this support, each player i must be indifferent between f_i and s_i, given the anticipated behavior of the other player. Notice that player 2's anticipated payoff against player 1's randomized strategy σ_1 depends on her pure strategy choice as follows:

$$u_2(\sigma_1, [f_2]) = 1\sigma_1(f_1) + 0\sigma_1(s1) \text{ if 2 chooses} f_2$$
$$u_2(\sigma_1, [s_2]) = 1\sigma_1(f_1) + 3\sigma_1(s1) \text{ if 2 chooses} s_2$$

So for player 2 to be willing to choose either f_2 or s_2, each with positive probability, these two expected payoffs must be equal. That is, we must have $\sigma_1(f_1) = 3\sigma_1(s_1)$. This equation, together with the probability equation $\sigma_1(f_1) + \sigma_1(s_1) = 1$ implies that $\sigma_1(f_1) = 0.75$ and $\sigma_1(s_1) = 0.25$. Notice that player 2 should be willing to randomize between her pure strategies in the support. Similarly, to make player 1 willing to chose either f_1 or s_1, each with positive probability, we must have

$$3\sigma_2(f_2) + 0\sigma_2(s_2) = 0\sigma_2(f_2) + 1\sigma_2(s_2)$$

which implies that $\sigma_2(f_2) = 0.25$ and $\sigma_2(s_2) = 0.75$. Thus, the equilibrium with support $\{f_1, s_1\} \times \{f_2, s_2\}$ must be

$$(\sigma_1, \sigma_2) = (0.75[f_1] + 0.25[s_1], 0.25[f_2] + 0.75[s_2])$$

8.4.3 Significance of Nash equilibria

Nash's (1951) concept of equilibrium is probably the most important solution concept in game theory. The general argument for the importance of this concept may be summarized as follows. Suppose that we are acting either as theorists, trying to predict the players' behavior in a given game, or as social planners, trying to prescribe the players' behavior. If we specify which (possibly randomized) strategies should be used by the players and if the players understand this specification also (recall that they know everything that we know about the game), the we must either specify an equilibrium or impute irrational behavior to some players. If we do not specify an equilibrium, then some player could gain by changing his strategy to something other than we have specified for him. Thus non-equilibrium specification would be a self-denying prophecy if all players believed it.

8.4.4 The focal-point effect

As the Battle of the sexes game in Table 8.5, shows, a game may have more than one equilibrium. Indeed, it is easy to construct games in which the set of equilibria is quite large.

When a game has multiple equilibria, the constraint imposed by the concept of Nash equilibrium on social planners and theorists becomes weaker. We know that any one of these equilibria, if it were expected by all players, could become a self-fulfilling prophecy. For example, in the Battle of the sexes, suppose that (for some reason) players 1 and 2 expected each other to implement the (f_1, f_2) equilibrium. Then each player would expect to maximize his own payoff by fulfilling this expectation. Player 2 would prefer the (s_1, s_2) equilibrium, but choosing s_2 instead of f_2 would reduce her payoff from 1 to 0, given that she expects player one to choose f_1.

Thus, to understand games with multiple equilibria, we must ask what might cause the players in a game to expect each other to implement some specific equilibrium. The phenomena that tend to

focus the players' attention on one equilibrium is called the *focal-point effect*. The focal equilibrium is an equilibrium that has some property that conspicuously distinguishes it from other equilibria. According to the focal-point effect, if there is one focal point equilibrium in a game, when we should expect to observe that equilibrium.

8.4.5 Bayes-Nash equilibrium

In games with complete information, all of the players know other players' preferences, whereas they wholly or partially lack this knowledge in games with incomplete information. Since the Nash equilibrium is based on the assumption that the players know each others' preferences, no methods has been available for analyzing games with incomplete information, despite the fact that such games best reflect many strategic interactions in the real world. Analysis of games where players do not know each others' payoff or preferences was usually in the domain of decision theory and uncertainty, since game-theoretic approaches were inadequate to handle such complexities.

Harsani's work (1967-1968) [35] provided the tools required for analyzing such interesting games. He postulated that every player is one of several different 'types'. Each player in a game with incomplete information chooses a strategy for each of his types. Under certain conditions, Harsani showed that for every game with incomplete information, there is an equivalent game with complete information. In other words, he showed that games with incomplete information can be transformed into games with imperfect information; it is to these games that the machinery of game theory applies to. A mixed strategy equilibrium in such static-Bayesian games is called Bayes-Nash equilibrium. That is, a Bayes-Nash equilibrium specifies an action or randomized strategy for each type of each player, such that each type of each player would be maximizing how own expected utility when he knows his own given type but does not know the other players' types. Notice that, in a Bayes-Nash equilibrium, a player's action can depend on his own type but not on the types of other players. We must specify what every possible type of every player would do, not just the actual type, because otherwise we could not define the expected utility payoff for a player who does not know the other players' actual types.

In a static Bayesian game, player 1's strategy is a best response to player 2's if, for each of player 1's types, the action specified by 1's action rule for that type maximizes 1's expected payoff, given player 1's belief about player 2's type and player 2's strategy. It is important to note that the best responses of the players are conditional on the beliefs held by them. Thus, the idea of beliefs is central

to the analysis of incomplete information games. A classic example of static bayesian games is the sealed-bid action, where each player knows only his private valuation. The players bid based on the belief they hold about the probability distribution over all possible evaluations.

To formally state the definition of a Bayes-Nash equilibrium, let Γ^b be a Bayesian game

$$\Gamma^b = (N, (C_i)_{i \in N}, (T_i)_{i \in N}, (p_i)_{i \in N}, (u_i)_{i \in N})$$

A *randomized-strategy profile* for the Bayesian game Γ^b is any σ in the set $\times_{i \in N} \times_{t_i \in T_i} \Delta(C_i)$, that is, any σ such that

$$\sigma = ((\sigma_i(c_i|t_i))_{c_i \in C_i})_{t_i \in T_i, i \in N}$$
$$\sigma_i(c_i|t_i), \forall c_i \in C_i, \forall t_i \in T_i, \forall i \in N,$$
$$\sum_{c_i \in C_i} \sigma_i(c_i|t_i) = 1, \forall t_i \in T_i, \forall i \in N.$$

In such a randomized-strategy profile σ, the number $\sigma_i(c_i|t_i)$ represents the conditional probability that player i would use action c_i if his type were t_i. In the randomized-strategy profile σ, the randomized strategy for type t_i of player i is

$$\sigma_i(\cdot, |t_i) = (\sigma_i(c_i|t_i))_{c_i \in C_i}.$$

A Bayes-Nash equilibrium of the game Γ^b is any randomized-strategy profile σ such that, for every player i in N and every type t_i in T_i,

$$\sigma_i(\cdot|t_i) = \arg \max_{\tau_i \in \Delta(C_i)} \sum_{t_{-i} \in T_{-i}} p_i(t_{-i}|t_i) \sum_{c \in C} \left(\prod_{j \in N \setminus \{i\}} \sigma_j(c_j|t_j) \right) \tau_i(c_i) u_i(c_i|t)$$

where for any $\tau_i \in \Delta(C_i)$, we let (σ_{-i}, τ_i) denote the randomized-strategy profile in which the i-component is τ_i and all other components are as in σ.

For a simple, two-player example, suppose that $C_1 = \{x_1, y_1\}$, $C_2 = \{x_2, y_2\}$, $T_1 = \{1.0\}$ (so player 1 has only one possible type and no private information). $T_2 = \{2.1, 2.2\}$, $p_1(2.1|1.0) = 0.6$, $p_2(2.2|1.0) = 0.4$, and the utility payoffs (u_1, u_2) depend on the actions and player 2's type as in Tables 8.6 and 8.7.

In this game, y_2 is a strongly dominated action for type 2.1, and x_2 is a strongly dominated action for type 2.2, so 2.1 must choose x_2 and 2.2 must choose y_2 in a Bayes-Nash equilibrium. Player 1 wants to get either (x_1, x_2) or (y_1, y_2) to be the outcome of the game, and he thinks that 2.1 is more likely than 2.2. Thus, the unique Bayes-Nash equilibrium of this game is

$$\sigma_1(\cdot|1.0) = [x_1], \sigma_2(\cdot|2.1) = [x_2], \sigma_2(\cdot|2.2) = [y_2]$$

This example illustrates the danger of analyzing each matrix separately, as if it were a game with complete information. If it were common knowledge that player 2's type was 2.1, the the players would be in the matrix on the left in Table 8.6, in which the unique equilibrium is $x_1, x_2)$. If it were common knowledge that player 2's type was 2.2, then the players would be in matrix on the right in Table 8.7, in which the unique equilibrium is y_1, y_2. Thus, if we only looked at the full-information Nash equilibria of these two matrices, then we might make the prediction, "the outcome of the game will be (x_1, x_2) if player 2's type is 2.1 and it will be (y_1, y_2) if player 2's type is 2.2".

This prediction would be absurd, however, for the actual Bayesian game in which player 1 does not initially know player 2's type. Notice first that this prediction ascribes two different actions to player 1, depending on player 2's type (x_1 if 2.1, and y_1 if 2.2). So player 1 could not behave as predicted unless he received some information from player 2. That is, the prediction would be impossible to fulfill unless some kind of communication is added to the structure of the game. Now notice that player 2 prefers (y_1, y_2) over (x_1, x_2) if her type is 2.1, and she prefers (x_1, x_2) over (y_1, y_2) if her type is 2.2. Thus, even if communication between players were allowed, player 2 would not be willing to communicate the information that is necessary to fulfill this prediction, because it would always give her the outcome that she prefers less. She would prefer to manipulate her communication to get the outcomes (y_1, y_2) if 2.1 and (x_1, x_2) if 2.2.

	C_2	
C_1	x_2	y_2
x_1	1,2	0,1
y_1	0,4	1,3

Table 8.6: Type $t_2 = 2.1$ for player 2

	C_2	
C_1	x_2	y_2
x_1	1,3	0.4
y_1	0,1	1,2

Table 8.7: Type $t_2 = 2.2$ for player 2

8.5 Dynamic non-cooperative games

8.5.1 Subgame perfect equilibrium

Subgame perfectness refines the Nash equilibrium in extensive form games and is based on subgames of the original game. Essentially, a *subgame* of a game in extensive form is any game starting from some node of the original game. An equilibrium is 'subgame perfect' if for every subgame of the original game, the equilibrium strategies restricted to the subgame continue to be an equilibrium. For instance, consider the extensive form game in Table 8.8. The normal form reveals that the game has two equilibria, namely (L, L') and (R, R'). The second nash equilibrium exists because player 1's best response to L' by player 2 is to end the game by choosing L. But (L, L') relies on the incredible threat by player 2 to play L' rather than R', if given the move. But for player 2 choosing L' when given the move is not a best response. Hence (L, L') cannot be an equilibrium since it does not satisfy the equilibrium conditions for the subgame starting at node 2 of the game. The game has only one subgame perfect equilibrium, i.e. (R, R'). Subgame Perfectness is a fundamental refinement of the Nash equilibrium in dynamic games with complete information. It is often used to reduce the number of possible equilibria and to eliminate equilibria sustained by incredible threats.

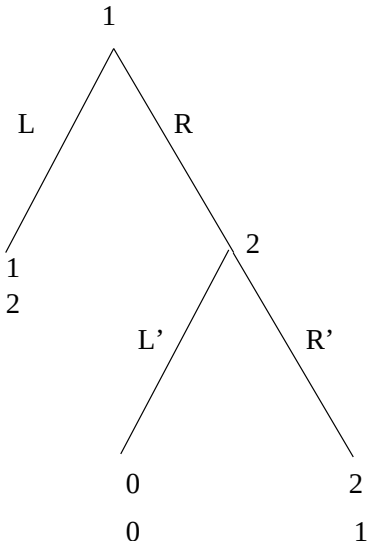


Figure 8.2: A game in extensive form

	C_2	
C_1	L'	R'
L	1,2	1,2
R	0,0	2,1

Table 8.8: A game in strategic form

8.5.2 Perfect Bayesian equilibrium

To analyze dynamic Bayesian games, we introduce the concept of perfect Bayesian equilibrium. A related concept is sequential equilibrium which is equivalent to perfect Bayesian equilibrium in many applications but in some cases is slightly stronger. Sequential equilibrium is more complicated to define and to apply than perfect Bayesian equilibrium, so most authors use the latter. The crucial new feature of perfect Bayesian equilibrium is due to Kreps and Wilsson [51]: beliefs are elevated to the level of importance of strategies in the definition of equilibrium. That is, the definition of equilibrium no longer consists of just a strategy for each player but now also includes a belief for each player whenever the player has the move but is uncertain about the history of prior play. The advantage of making the players' beliefs an explicit part of the equilibrium is that, just as we previously insisted that the players choose credible (that is, subgame-perfect) strategies, we can now also insist that they hold reasonable beliefs.

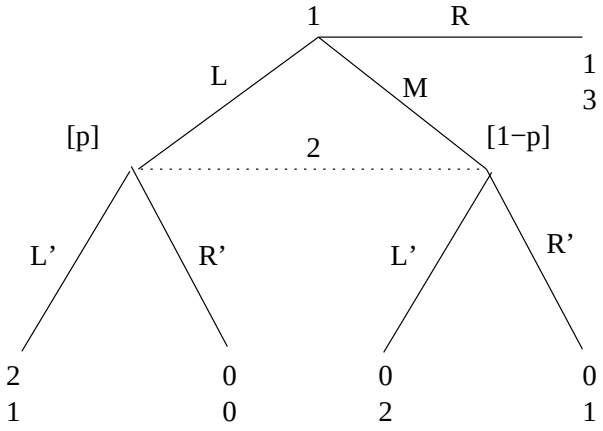


Figure 8.3: A game in extensive form

	C_2	
C_1	L'	R'
L	2,1	0,0
M	0,2	0,1
R	1,3	1,3

Table 8.9: A game in strategic form

To illustrate why the player's beliefs are as important as their strategies, consider the example in Table 8.9. First, player 1 chooses among three actions: L,M and R. If player 1 chooses R the the game ends without a move by player 2. If player 1 chooses either L or M the player 2 learns that R was not chosen (but not which of L or M was chosen) and the chooses between two actions, L' and R', after which the game ends. (The dashed line connecting player 2's two decision nodes in the game three on the left of Figure 8.9 indicates that if player 2 gets the move, player 2 does not know which node has been reached – that is, whether player 1 has chosen L or M. The probabilities p and $1-p$ attached to player 2's decision nodes will be explained below.) Payoffs are given in the game tree.

The strategic-form representation of this game on the right-hand side of Figure 8.9 reveals that there are two pure-strategy Nash equilibria: (L, L') and (R, R'). We first ask whether these Nash equilibria are subgame-perfect. Because a subgame is defined to begin when the history of prior play is common knowledge, there are no subgames in the game tree above. (After player 1's decision node at the beginning of the game, there is no point at which the complete history of play is common knowledge: the only other nodes are player 2's, and if these nodes are reached, then player 2 does not know whether the previous play was L or M.) If a game has no subgames, the the requirement of subgame-perfection – namely, that the players' strategies constitute a Nash equilibrium on every subgame – is trivially satisfied. Thus, in any game that has no subgames the definition of subgame-perfect Nash equilibrium is equivalent to the definition of Nash equilibrium, so in this example both (L, L') and (R, R') are subgame-perfect Nash equilibria. Nontheless, (R, R') clearly depends on a non credible threat: if player 2 gets the move, the playing L' dominates player R', so player 1 should not be induced to play R by 2's threat to play R' if given the move.

On way to strengthen the equilibrium concept so as to rule out the subgame-perfect Nash equilibrium (R, R') is to impose two requirements

Requirement 1: Whenever a player has the move and is uncertain about the history of prior play, the player must have a *belief* over the set of feasible histories of play.

Requirement 2: Given their beliefs, the players' strategies must be *sequentially rational*. That is, whenever a player has to move, the player's action (and the player's strategy from then on) must be optimal given the player's belief at that point (and the other player's strategies from then on).

In the example above, Requirement 1 implies that if player 2 gets the move, then player 2 must have a belief about whether player 1 has played L or M. This belief is represented by the probabilities p and $1 - p$ attached to the relevant nodes in the game tree. Given player 2's belief, the expected payoff from playing R' is $p \cdot 0 + (1 - p) \cdot 1 = 1 - p$, while the expected payoff from playing L' is $p \cdot 1 + (1 - p) \cdot 2 = 2 - p$. Since $2 - p > 1 - p$ for any value of p, Requirement 2 prevents player 2 from choosing R'. Thus simply requiring that each player have a belief and act optimally given this belief suffices to eliminate the implausible equilibrium (R, R') in this example.

What about the other subgame-perfect Nash equilibrium, (L, L') ? Requirement 1 dictates that player 2 have a belief but does not specify what it should be. In the spirit of rational expectations, however, player 2's belief in this equilibrium should be $p = 1$. We state this idea a bit more formally as

Requirement 3: Where possible, beliefs should be determined from Bayes' rule from the players equilibrium strategies.

Requirements 1 through 3 constitute the definition of *perfect Bayesian equilibrium*.

8.6 Repeated games

People may behave quite differently toward those with whom they expect no future relationship than toward those with whom they expect no future interaction. To understand how rational and intelligent behavior may be affected by the structure of a long-term relationship, we study repeated games.

In a repeated game, there is an infinite sequence of *rounds*, or points in the game, at which players may get information and choose moves. That is, a repeated game has an *infinite time horizon* unlike the finite game trees. In a repeated game, because no move is necessarily the last, a player must always consider the effect that this current move might have on the moves and information of other players in the future. Such considerations may lead players to be more cooperative, or more belligerent, in a repeated game than they would be in a finite game in which they would know when their relationship will end.

To introduce the study of repeated games, let us begin with a well known example, the repeated prisoner's dilemma. Here g_i is i's generous move, and f_i is i's selfish move. The only equilibrium in this game is (f_1, f_2). Not let us consider what happens if player 1 and 2 expect to repeatedly play this game with each other every day, for a long time.

For example, suppose that the number of times that the game will be played is a random variable, unknown to the players until the game stops, and that this random stopping time has a geometric distribution with expected value 100. That is, the probability of play continuing for exactly k rounds is $(0.99)^{k-1} \times 0.01$. Thus, after each round of play, the probability that players 1 and 2 will meet and play again is 0.99; and at any time when they are still playing, the expected number of further rounds of play is 100. In this repeated game, generous behavior can be supported in equilibrium. Consider, for example, the strategy of playing g_i every day until someone plays $f_1$or f_2 and thereafter playing f_i. If both players plan to use this strategy, then, at any time in the game each player will get an expected total future payoff of

$$\sum_{k=1}^{\infty}(0.99)^{k-1}(0.01)(5k) = 500,$$

as long as no one has deviated and chosen f_i. But if either player i deviated from this strategy and chose f_i on some particular day, then his expected total future payoff from this day onward would be

$$6 + \sum_{k=2}^{\infty} (0.99)^{k-1}(0.01)(1k) = 105,$$

On the other hand, if anyone has ever played f_1 or f_2, then it is easy to see that neither player could gain by choosing g_i when the other player is expected to follow this strategy and so always choose the selfish move hereafter. Thus, at any point in time, with any history of past moves, neither player can expect to gain by any unilateral deviation from this strategy.

The key in this generous equilibrium is that, whenever the players meet, they believe that there is a very high probability that they will play again; so the hope of inducing future generous behavior by the other payer can give each player an incentive to be generous. We could not construct such an equilibrium if the players knew in advance when the game would end. For example, if it were common knowledge that the game would be played for exactly 100 rounds, then the resulting 100-round game would have a unique sequential equilibrium, with both always playing (f_1, f_2) at every round. At the last (100th) round, a generous move cannot induce any further generosity by the other player (because there is not future), so there is no reason for either player to be generous. So the players should choose (f_1, f_2) at the 100th round, no matter what the prior history would be. Thus, at the 99th round, the players must know that their moves will have no impact on the 100th-round moves, so they should play (f_1, f_2) on the 99th round as well. Working backward through the game, it is straightforward to verify by induction that the unique sequential-equilibrium scenario is indeed to play f_1 and f_2 at every round.

Thus, rational behavior in a repeated game with potentially infinite time horizon may be quite different from rational behavior in the corresponding game in which the number of rounds is arbitrarily large but finite.

The subgame perfect equilibrium is a natural and standard way to analyze repeated games.

8.7 Cooperative games

An outcome of a cooperative game is defined by the values of the players utilities, $\mathbf{u} = (u_1, \ldots, u_N)$. The set of all possible outcomes is called a bargaining domain, U. Cooperative game theory assumes that bargaining domains are convex, closed, and bounded sets of $R^N : R \geq 0$ where N denotes the

number of players. The outcome of the game also depends on the starting point $\mathbf{s} = (s_1, \ldots, s_N)$: $\mathbf{s} \in U$. This point can be interpreted as a pre-game assumption that the solution point u^* can not be worse than the starting point $(u_i^* \geq s_i)$. Let $U_0 = \{\mathbf{u} \in U : u_i^* \geq s_i$ denote the set of possible outcomes when the bargaining process starts at $\mathbf{s}$.

The main objective of cooperative game theory is to define a fair solution to the game defined by $(\mathbf{s}, U)$. In general, two different approaches can be considered. The first one assumes that there is a bargaining process. It means that the solution is achieved in several steps (in the limit, it might be a continuous process). The bargaining process can begin at the starting point or from a pair of solutions than can each be treated as an offer and a counter offer (both may be Pareto optimal). The second approach ignores the bargaining process and concentrates on a solution point determined by a set of axioms that should provide fairness features acceptable for all players. The situation is somewhat analogous to a conflict resolved by an arbiter, hence, the choice of game "arbitration scheme".

We study a class of arbitration schemes that is based on the following fairness axioms:

1. *Symmetry*: The solution does not depend on specific labels, i.e. users with the same minimum performance measures and the same utilities will have the same performances.

2. *Pareto optimality:* The solution is on the Pareto boundary.

3. *Invariance with respect to utility transformations*: The solution for any positive affine transformation of $(\mathbf{s}, U)$ denoted by $V(U), V(\mathbf{s})$ is $V(\mathbf{u}^*)$ where $\mathbf{u}^*$ is the solution to the original system.

4. *Independence of irrelevant alternatives*: If the solution for $(\mathbf{s}, U_1)$ is $\mathbf{u}^*$, and $U_2 \subset U_1, \mathbf{u}^* \in U_2$ then $\mathbf{u}^*$ is also the solution for $(\mathbf{s}, U_2)$. In other words, this axiom states that if U_1 is reduced, in such a way that the original solution and starting point are still included, the solution to the new problem remains the same.

Note that based on the axiom 3, without losing any generality, we can linearly transform the pair $(\mathbf{s}, U)$ in such a way that $\max\{u_i' : \mathbf{u}' \in U'\}$ and $s_i' = 0$ where the prime denotes the model after transformation. We call this case normalized.

From the fairness axioms formulated by Nash it can be shown the operating point is obtained by optimizing the product of normalized player utilities [15]:

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} \prod_{i=1}^{N} u_i \tag{8.7}$$

8.8 Applications

8.8.1 Bandwidth sharing

In [77] the problem of bandwidth sharing between different users is considered. A general network topology is studied, and the question is how much bandwidth, or extra capacity, should be allocated by the network to each of the users at each link.

The solution framework is based on the idea of the Nash bargaining solution from cooperative game theory, which not only provides the rate settings of users that are Pareto optimal from the point of view of the whole system, but are also consistent with the fairness axioms of game theory. The utilities represents the (normalized) traffic rates of the users and are given be the following optimization rule:

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} \prod_{i=1}^{N} u_i \tag{8.8}$$

It is shown in [77] that this sharing of the bandwidth has the proportional fairness property introduced in [44].

8.8.2 Call admission control

In [25] the issue of call admission control (CAC) on a single link is studied. To design a CAC policy on the call level one has must often make a trade off between fairness and efficiency. Grade of Service (GoS) measures the performance of the network on the call level. Typical GoS measures include call blocking probability and average call set-up delay. In the literature, the GoS issue for CAC can have been solved in at least two ways. In the first, the GoS constraints are directly taken into account during the optimization process. In the second, each call class is allocated a certain weight or a reward parameter and the control objective is to minimize the network cost or to maximize the reward from the network. Thus, by changing the values of the weights or reward parameters, different GoS distributions among the call classes can be achieved.

Hence, one seeks a CAC policy that efficiently utilizes network resources while at the same time being fair to the various classes being supported. The theory of cooperative games provides a natural and precise framework for formulating such a multi criterion problem as well as solution concepts.

From the fairness axioms formulated by Nash it can be shown the operating point is obtained by optimizing the product of per-class normalized utilities.

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} \prod_{i=1}^{N} u_i \tag{8.9}$$

The utility for each class is measured in terms of carried traffic:

$$\overline{a}_j = \overline{\lambda}_j b_j \mu_j^{-1} \tag{8.10}$$

where b_j denotes the bandwidth requirement of a class j call, $\overline{\lambda}_j$ denotes the acceptance rate of class j calls, and μ_j^{-1} denotes the average holding time of a class j call.

The optimization of the product of per-class carried traffics is done as follows.

1. An initial value of the reward parameter vector $\mathbf{r} = (r_1, \ldots, r_N)$ is chosen.
2. The carried traffic for each class is evaluated from the value determination algorithm from Markov decision theory.
3. A new value of the reward parameter vector is chosen, e.g. using the concept of interval halving.
4. Repeat from 2 until the new CAC policy π' is the same as the old policy π.

A numerical comparison indicates that the trunk reservation and dynamic trunk reservation can provide fair, efficient solutions, close to the optimal ones.

8.8.3 Routing

Single-shot game

In this sub section we describe a routing game that is played only once. We consider a set $\mathcal{I} = \{1, \ldots, I\}$ of users, that share a set $\mathcal{L} = \{1, \ldots, L\}$ of communication links interconnecting a common source to a common destination node. Let c_s be the capacity of link s, and $C = \sum_{s \in \mathcal{L}} c_s$ be the total capacity of the system. Each user i has a throughput demand that is some process with average rate $r^i > 0$. We assume that $r^1 \geq r^2 \geq \ldots \geq r^I$. Let $R = \sum_{i \in \mathcal{I}} r^i$ denote the total demand of the users. We only consider capacity configurations $\mathbf{c} = (c_1, \ldots, c_L)$ that can accommodate the total user demand, i.e. configurations with $C > R$.

User i ships its flow by splitting its demand r^i over the set of parallel links, according to some individual performance objective. Let f_s^i denote the expected flow that user i sends on link s. The user flow configuration $\mathbf{f}^i = (f_1^i, \ldots, f_L^i)$ is called a routing *strategy* of user i and the set $F^i = \{\mathbf{f}^i \in \mathbb{R}^L : 0 \leq f_s^i \leq c_s, s \in \mathcal{L}; \sum_{s\in\mathcal{L}} f_s^i = r^i\}$ of strategies that satisfies the user's demand is called the strategy space of user i. The system flow configuration $\mathbf{f} = (\mathbf{f}^1, \ldots, \mathbf{f}^I)$ is called routing *strategy profile* and takes values in the product strategy space $F = \otimes_{i\in\mathcal{I}} F^i$.

The performance objective of user i is quantified by means of a cost function $J^i(\mathbf{f})$. The user aims to find a strategy $\mathbf{f}^i \in F^i$ that minimizes its cost. This optimization problem depends on the routing decisions of the other users, described by the strategy profile $\mathbf{f}^{-i} = (\mathbf{f}^1, \ldots, \mathbf{f}^{i-1}, \mathbf{f}^{i+1}, \ldots, \mathbf{f}^I)$, since J^i is a function of the system flow configuration $\mathbf{f}$. A *Nash equilibrium* of the routing game is a strategy profile from which no user finds it beneficial to unilaterally deviate. Hence, $\mathbf{f} \in F$ is a Nash equilibrium if

$$\mathbf{f}^i \in \arg\min_{\mathbf{g}^i \in F^i} J^i(\mathbf{g}^i, \mathbf{f}^{-i}), \qquad i \in \mathcal{I}. \tag{8.11}$$

The problem of existence and uniqueness of equilibria has been investigated in [65] for certain classes of cost functions. Here, we consider cost functions that are the sum of link cost functions:

$$J^i(\mathbf{f}) = \sum_{s\in\mathcal{L}} J_s^i(\mathbf{f}_s), \qquad J_s^i(\mathbf{f}_s) = f_s^i T_s(f_s), s \in \mathcal{L}, \tag{8.12}$$

where $\mathbf{f}_s = (f_s^1, \ldots, f_s^I)$, and $T_s(f_s)$ is the average delay per unit flow on link s that depends only on the total flow $f_s = \sum_{i\in\mathcal{I}} f_s^i$ on that link. The average should be interpreted as a general *congestion cost* per unit flow, that encapsulates the dependence of the quality of a finite capacity resource on the total load f_s offered to it. In the present method, we concentrate on congestion costs of the form

$$T_s(f_s) = \begin{cases} (c_s - f_s)^{-1}, & f_s < c_s \\ \infty, & f_s \geq c_s \end{cases} \tag{8.13}$$

Given a strategy profile $\mathbf{f}^{-i}$ of the other users, the cost of user i, as defined by (8.12) and (8.13), is a convex function of its strategy $\mathbf{f}^i$. Hence, the minimization problem in (8.11) has a unique solution. How to obtain the optimal routing strategy profile $\mathbf{f} = (\mathbf{f}^1, \ldots, \mathbf{f}^I)$ is described in [50].

Repeated game

In this subsection we describe a routing game that is played repeatedly an infinite number of times. The basic model of the game is the same as for the single-shot routing game. In a repeated game there is possibility of strategies that results in Nash equilibrium points (NEPs) which are more efficient than in a single shot game.

The existence of a subgame-perfect NEP that not only achieves the system-wide optimum cost but also yields a cost for each user no greater than its single-shot game NEP cost is shown in [52] for two-node multiple link networks. It is shown that more general networks where all users have the same source-destination pair have a subgame-perfect NEP that achieves the minimum total system cost, under a mild technical assumption. It is shown that general networks with users having multiple source-destination pairs do not necessarily have such an NEP.

In order for every user to benefit from the cooperation in comparison to the single shot game NEP, the total system cost should decrease from that of the single shot game. Obviously, the total system cost, even if all users cooperate, can not be smaller than the system optimum. Therefore, if the users can achieve the system optimum through cooperation in a dynamic setting, some users, if not all, can reduce their costs while the other users still do not have an incentive to deviate due to the existence of credible threats.

8.8.4 Virtual path bandwidth allocation

The issue of virtual path bandwidth allocation is studied in [53]. A set of N users is considered, denoted by $\mathcal{I} = \{1, \ldots, N\}$, that share a resource of total capacity (bandwidth) of B units. Each user reserves some of the resource capacity in order to establish a virtual path for its incoming calls. Calls that are blocked on the virtual path level leads to performance degradation.

Users are noncooperative, meaning that each seeks to optimize its own performance. To that end, the users minimizes a cost function. This function should account for the following tradeoff. On the one hand, each user should try to minimize the blocking probability of its incoming calls at the virtual path level, which is a decreasing function of the reserved capacity of the user's virtual paths. On the other hand, reserving capacity becomes more difficult (and thus costlier) as the system's resources are less available. This tradeoff is formally quantified as follows. Denote by C_i the *strategy* of user i, i.e. the amount of capacity that user i reserves. We can assume, without loss of generality, that a user always obtains the amount of capacity it requests. The *strategy space* of user i is $[0, B]$, i.e.

$C_i \in [0, B]$. Hence, $\mathbf{C} = (C_1, C_2, \ldots, C_N)$ is the *game strategy vector*, and the *game strategy space* is $\mathcal{C} = \{\mathbf{C} | \forall i 0 \leq C_i \leq B\}$. We also denote by C the total amount of reserved capacity, i.e. $C = \sum_{i \in \mathcal{I}} C_i$. The cost function for user i, denoted by J_i, is of the following form:

$$J_i(\mathbf{C}) = J_i(C_i, C) = F_i(C_i, C) + G_i(C_i) \tag{8.14}$$

where the function F_i accounts for the availability of resources as perceived by the ith user, whereas the function G_i accounts for the effect that the amount of reserved capacity has on the performance of that user. The functions F_i and G_i have well defined properties and examples of these functions are given in [53].

Since the cost function of each user depends on the strategies of all the other users, we are faced with a *noncooperative game*. We are interested in the Nash equilibrium solution of the game, i.e. we seek a game strategy vector such that no user finds it beneficial to change its strategy. Formally, a game strategy vector $\mathbf{C}^*$ in the strategy space $\mathcal{C}$ is a Nash equilibrium point (NEP) if, for all $i = 1, 2, \ldots, N$, the following holds:

$$J_i(\mathbf{C}^*) = J_i(C_1^*, \ldots, C_{i-1}^*, C_i^*, C_{i+1}^*, \ldots C_N^*) = \tag{8.15}$$

$$\min_{0 \leq C_i \leq B} J_i(C_1^*, \ldots, C_{i-1}^*, C_i, C_{i+1}^*, \ldots C_N^*) \tag{8.16}$$

The NEP can be found from an iterative scheme. At each iteration, each user recomputes its reserved capacity so as to optimize its cost function with respect to the current state of the system. Two different versions of the iterative scheme is proposed in [53]: Gauss-Seidel scheme and Jacobi scheme.

The resources available to customers cannot always be modeled by a single link. For example, capacity might be available in various links of different quality, thus a user would have preferences among the links. Moreover, such links may not directly connect the source node and destination node, meaning that resources may take the form of a graph of general topology. Such extensions are investigated in [53].

Bibliography

[1] Adas A., Traffic models in broadband networks, *IEEE Communications Magazine*, Vol 35., No 7, July 1997

[2] Andersen A. and Nielsen B., A Markovian approach for modelling packet traffic with long range dependence, *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 5, June 1998.

[3] Albin S., Approximating a point Process by a Renewal Process, *Opretations Research*, Vol. 32, No. 5, pp. 1133-1162, 1984.

[4] Ash G., An analytical model for adaptive routing networks, *IEEE Transactions on Communications*, Vol. 41, No. 11, November 1993.

[5] ATM Forum, Traffic Management Specification, Version 4.0, 1996.

[6] Balcioglu B., Jagerman D., Altiok T., Merging and splitting autocorrelated arrival processes and impact on system performance,preprint, Industrial and Systems Engineering, Rutgers University, USA, 2004.

[7] Baskett F., Chandy K., Muntz R., Palacios F., Open, closed and mixed networks of queues with different classes of customers, *Journal of the ACM*, Vol. 22, No. 2, pp. 248-260, 1975.

[8] Beran J., Sherman R., Taqqu M., Willinger W., Long-range dependence in variable-bit-rate video traffic, *IEEE Transactions on Communications*, Vol 43, pp. 1566-1579, 1995.

[9] Bertsimas D., An exact FCFS waiting time analysis for a general class of G/G/s queueing systems, *Queueing Systems: Theory and Applications*, 3, pp. 305-320,, 1988..

[10] Bertsimas D., An analytic approach to a general class of G/G/s queueing systems, *Operations Research*, 38, pp. 139-155, 1990.

[11] Bolch G., Greiner S., De Meer H., Trivedi K., *Queueing networks and Markov chains*, John Wiley and Sons, Inc., 1998.

[12] Brichet F., Roberts J., Simonian A. and Veitch D., Heavy traffic analysis of a storage model with long range dependent on/off sources, *Queueing Systems*, No 23., pp. 197-215, 1996.

[13] Brockmeyer E., Halstrom H., Jensen A., *The life and works of A.K. Erlang, Acta Polytechnica Scandinavia*, 1960.

[14] Burke P, The output of a queuing system, *Operations Research*, 4, pp. 699-704.

[15] Cao XR, Preference functions and bargaining solutions, In *Proc. CDC-21*, Orlando, FL, USA, Dec. 1982.

[16] Cao XR, Shen H, Milito R, Wirth P,Internet pricing with a game theoretical approach: concepts and examples, *IEEE/ACM Transactions on Networking*, Vol. 10, No. 2, pp. 208-216, 2002

[17] Cherry W., The superposition of two independent Markov renewal processes, PhD Dissertation and technical Report No. 72-10, Department of Industrial Engineering and Operations Research, University of Michigan, Ann Arbor, MI, 1972.

[18] Cox D., The analysis of non-Markovian stochastic processes by inclusion of supplementary variables, *Proc. Camb. Phil. Soc.*, 51, pp. 433-441, 1955.

[19] Cox D., Miller H., *The Theory of Stochastic Processes*, Chapman and Hall, 1990.

[20] Crovella M., Bestavros A., Self-similarity in World Wide Web traffic – evidence and possible causes, *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 835-846, 1997.

[21] Deng S., Empirical model of WWW document arrivals at access link, In *Proc. of International Conference on Communication*, ICC'96, 1996.

[22] Downey A., Evidence for long-tailed distributions in the Internet, *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.

[23] Dziong Z., *ATM Network Resource Management*, first edition, McGraw-Hill, 1997.

[24] Dziong Z., Mason L., Call admission and routing in multi-service loss networks, *IEEE Transactions on Communications*, Vol. 42, No. 2/3/4, pp. 2011-2022, February/March/April, 1994.

[25] Dziong Z., Mason L., Fair-efficient call admission control policies for broadband networks – a game theoretic framework, *IEEE/ACM Transactions on Networking*, Vol. 4, No. 1, Feb. 1996.

[26] Eun D., Ahn H., Roh H. and Kim J., Effects of long-range dependence of VBR video traffic on queueing performances, *Proc. of GLOBECOM'97*, pp. 1440-1444, Phoenix, USA, Nov. 1997.

[27] Farago A., Blaabjerg S., Gordos G. and Henk T., A new degree of freedom in ATM network dimensioning: optimizing the local configuration, *IEEE Journal of Selected Areas in Communications*, Vol 13., No. 7, September 1995.

[28] Feldmann A., Characteristics of TCP connection arrivals, Tech. Rep., AT&T Labs Research, 1998.

[29] Feldmann A., Gilbert A., Willinger A., and Kurtz T., The changing nature of network traffic: scaling phenomena, *Computer Communication Review*, April 1998.

[30] Frost V., Melamed B., Traffic modelling for telecommunication networks, *IEEE Communications Magazine*, Vol. 32, No. 2, February 1996.

[31] Garrett M and Willinger W., Analysis, modelling and generation of self-similar VBR video traffic, *Proc. of the ACM SIGCOMM'94*, London, UK, pp. 269-280. 1994

[32] Gibbens R., Kelly F. and Key P, Dynamic alternative routing – modelling and behaviour, *Proc. 12th International Teletraffic Congress*, Turin, Italy, 1988.

[33] Girard A. and Cote Y., Sequential routing optimization for circuit switched networks, *IEEE Transactions on Communications*, Vol. COM-32, No. 12, December 1984.

[34] Gordon W., Newell G., Closed queueing systems with exponential servers, *Operations Research*, 15, pp. 254-265, 1967.

[35] Harsani J. Games with incomplete information playe by Bayesian players, Part I-III. *Management science*, Vol. 14, No. 3, 1968.

[36] Heffes H., Lucantoni D., A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance, *Journal of Selected Areas in Communications*, Vol. SAC-4, No. 6, 1986.

[37] Howard R., *Dynamic programming and Markov processes*, MIT press, 1960.

[38] Hui J., A layered broadband switching architecture with physical or virtual path configurations, *IEEE Journal on Selected Areas in Communications*, vol. 9, no 9., pp. 1416-1426, Dec. 1991.

[39] Jacobsen S., Dittman L., A fluid flow queueing model for heterogeneous on/off traffic, RACE BLNT Workshop, Munchen, 1990.

[40] Jackson J., Networks of waiting lines, *Operations Research*, 5, pp. 518-521, 1957.

[41] Jain R., Congestion control and traffic management in ATM networks: recent advances and a survey, *Computer Networks and ISDN Systems*, Vol. 28, No. 13, pp. 1723-1738, October 1996.

[42] Kalai E., Smorodinsky M., Other solutions to the Nash's bargaining problem, *Economtrica*, Vol. 43, pp. 513-518, 1975.

[43] Kelly F., Routing in circuit-switched networks: optimization, shadow-prices and decentralization, *Adv. Appl. Prob*, No. 20, 1988.

[44] Kelly F., Charging and rate control for elastic traffic, *European Transactions on Telecommunications*, Vol. 8, No. 1, 1997.

[45] Kaj I., Convergence of scaled renewal processes to fractional brownian motion, preprint, Uppsala University, Sweden, 1999.

[46] Kendall D., Some problems in the theory of queues, *Journal Royal Statistcal Society*, 13, pp. 151-173, 1951.

[47] Kendal D.G., Stochastic processes occuring in the theory of queues and their analysis buy the method of imbedded Markov chain, *Annals of Mathemetical Statistics*, Vol. 24, pp. 277-289, 1953.

[48] Kimura T., A two-moment approximation for the mean waithing time in the GI/G/s queue, *Management Science*, 32, pp. 751-763, 1986

[49] Kleinrock L., *Queuing systems, Vol I, Theory*, John Wiley & Sons,1975.

[50] Korilis Y., Lazar A., Orda A., Capacity allocation under noncooperative routing, *IEEE Transactions on Automatic Control*, Vol. 42, pp. 309-325, 1997.

[51] Kreps D., Wilson R. Sequential equilibrium, *Econometrica*, Vol. 50, pp. 863-894, 1982.

[52] La R., Anantharam V., Optimal routing control: repeated game approach, *IEEE Transactions on Automatic Control*, Vol. 47, No 3, 2002.

[53] Lazar A., Orda A., Pendarakis D., Virtual path bandwidth allocation in multiuser networks, *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, 1997.

[54] Leland W, Taqqu M., Willinger W. and Wilson D., On the self-similar nature of Ethernet traffic (extended version), *IEEE/ACM Transactions on Networking*, Vol 2, No. 1, pp 1-15, 1994.

[55] Miyao Y., A call admission control scheme in ATM networks, *Proceedings of ICC'91*, pp. 391-396, 1991.

[56] Maglaris B., Anastassiou D., Sen P., Karlsson G. and Robbins J., Performance lodels of statistical multiplexing in packet video communications, *IEEE Transactions on Communications*, vol 36, no 7, pp. 834-844, July 1988.

[57] Mazumdar R., Mason L., Douligeris C., Fairness in network optimal flow control: optimality of product forms, *IEEE Transactions on Communications*, Vol. 39, No. 5, May 1991.

[58] McDysan D., *QoS and Traffic Management*, first edition, McGraw-Hill, 2000.

[59] Michiel H. and Laevens K., Teletraffic engineering in a broadband era, *Proceedings of the IEEE*, Vol 85, No 12., December 1997.

[60] Myerson R., *Game theory: analysis of conflict*, Harvard University Press, 1991.

[61] Nash J., The bargaining problem, *Econometrica*, Vol. 18, 1950.

[62] Nash J., Noncooperative games, *Annals of Mathematics*, 54, pp. 289-295, 1951.

[63] Norros I.,On the use of fractional brownian motion in the theory of connectionless networks, *IEEE Journal of Selected Areas in Communications*, Vol. 13, No. 6, August 1995.

[64] Onvural R., *Asynchronous Transfer Mode networks*, second edition, Artech House, 1995

[65] Orda A., Rom R., Shimkin N., Competitive routing in multiuser communication networks, *IEEE/ACM Transactions on Networking*, Vol. 1, No. 5, 1993.

[66] Partridge C., *Gigabit networking*, Addison-Wesley, 1994

[67] Paxson V. and Floyd S., Wide-area traffic: the failure of Poisson modelling, *Proc. ACM SIGCOMM'94*, London, UK, 1994.

[68] Peterson L, Davie B., *Computer Networks*, 2nd edition, Morgan Kaufman, 2000.

[69] De Prycker M., *Asynchronous Transfer Mode: solution for broadband ISDN*, third edition, Prentice Hall, 1995.

[70] Raiffa H., Arbitration schemes for the generalized two-person games, in *Contribution to the Theory of Game II*, H.W. Kuhn, A.W. Tucker Eds. Princeton, NJ, Princton University Press, 1953.

[71] Reiser M, Lavenverg S., Mean value analysis of closed loop queueing networks, *Journal of the ACM*, Vol. 27, No. 2, pp. 313-322, 1980.

[72] Robertazzi T., *Computer networks and systems: queueing theory and performance evaluation*,third edition, Springer, 2000.

[73] Selten R., Reexamination of the perfectness concept for equilibrium points in extensive games, *International Journal of Game Theory*, Vol. 4, pp. 25-55, 1975.

[74] Songhurst D. (editor), *Charging communication networks: from theory to practice*, Elsevier, 1999.

[75] Takahashi Y., Takami Y., A numerical method for the steady-state probabilities of a GI/G/s queueing system in a general class, *Journal of the Operations Research Society of Japan*, 19, pp. 147-157, 1986.

[76] Taqqu M., Willinger W., Sherman R., Proof of a fundamental result in self-similar traffic modeling, *ACM/SIGCOMM Computer Communications Review*, Vol. 27, pp. 5-23, 1997.

[77] Yaiche H., Mazumdar R., Rosenberg C., A game theoretic framework for bandwidth allocation and pricing in broadband networks, *IEEE/ACM Transactions on Networking*, Vol. 8, No. 5, Oct. 2000.

[78] Yechiali U., Customers optimal joining rules for the GI/M/s Queue, *Management Science*, Vol. 18, No. 7, pp. 434-443, 1972

[79] Whitt W., Approximating a point process by a renewal process, I:two basic methods, *Operations Research*, Vol. 30, No. 1, pp. 125-147, 1982.

[80] Whitt W., Approximations for the GI/G/m queue, *Productions and Operations Management*, Vol. 2, No. 2, pp. 114-161, 1993

[81] Whitt W.,A diffusion approximation for the G/GI/n/m queue , *Operations Research*, 2003.