Flow-based Networking and Quality of Service

Flow-based Networking and Quality of Service

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. J. T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op donderdag 7 mei 2009 om 10.00 uur

door

Teunis Johannis KLEIBERG

elektrotechnisch ingenieur
geboren te Dordrecht.

ISBN 90-407-2523-3

Keywords: QoS, flows, network performance

This research has been funded by Technology Foundation STW, project Network Dynamics and QoS (DTC.6421).

Printed in The Netherlands by Wöhrmann Print Service.

Cover art: a sample network consisting of routers (squares) carrying flows between hosts (circles).

aan mijn ouders,
voor hun onvoorwaardelijke steun
en vertrouwen

Thesis summary

Flow-based Networking and Quality of Service

During the past two decades the Internet has been widely deployed and integrated into the society, radically altering the way people communicate and exchange information. Through cheap, fast and reliable information transfer, the Internet interconnects billions of users covering practically the entire (developed) world. Although the Internet was intended as a research network between few a institutions in the United States, it has grown to take a central role in day-to-day communications and is by now considered indispensable. The last decade we have seen a migration of the classic telecommunication services, telephony, radio and television, to the Internet and the emergence of new services, such as online gaming and e-government. Hence, the Internet is replacing the existing, dedicated telecommunication networks and embodies a single medium for all (wired) communications. However, since these dedicated networks are tailored for one type of service, the migration to the Internet, which design goals are primarily scalability and reliability, creates a field of tension. The low latencies and losses for which the dedicated networks were optimized, cannot be guaranteed by the Internet, which philosophy relies on best-effort routing. Packets in the Internet can literally travel across the entire network, making it very hard to predict where the packets may travel, yielding tremendous uncertainties regarding packet-delays and loss. Consequently, in order to provide guarantees regarding the quality of a service, new techniques need to be implemented, which are commonly denoted by "Quality of Service" (QoS). Various implementations have been proposed in literature that aim at providing QoS in the Internet. One of these implementation relies on flow-based communication between end-hosts. In flow-based communications, the packets from a source to a destination that belong the same stream are forced along the same path, offering a better control over the packets and an improved quality control.

In this thesis we study the performance of networks using flow-based communication and we show by means of measurements that the current Internet can yield highly unpredictable behavior in packet delivery. We will discuss measurements performed on the Internet that serve as a motivation for the flow-level approach that is used throughout the thesis. Based on traceroute measurements we show that the best-effort environment of the Internet can lead to highly unpredictable behavior of packets. The measurements indicate that the routes that the packets in the Internet follow are very changeable due to dynamics in the routing plane. The lifetime of a route, i.e. the time between the first and last consecutive occasion that the particular route is used, appears to follow a power-law distribution. This unpredictable behavior may hamper the performance of real-time applications and serves as a motivation for the flow-based networking used in this thesis.

We introduce a new model that describes the network performance by using an analogy with queueing theory. The analysis of network performance is complicated by the many dependencies between the properties of a network. The objective is to express the network performance in terms of these properties and reveal the root of the problem. The model facilitates to express the emergent performance characteristics, such as the blocking rate of traffic flows and the maximum throughput of the network, in terms of network parameters that are given by design, such as the number of nodes. The model considers the network as a black box and minimizes the degrees of freedom in order to reduce the dependencies and improve the comprehensibility of the model. Due to the small number of parameters, we are able to discern the influence of these parameters on the performance. In addition, we address the difficulties of studying network dynamics, which can be traced back to the dependencies between the links and the importance of traffic fluctuations in the network on the overall performance.

In a more static scenario where flows are assumed to have an infinite duration, we study the maximum throughput of a network. Now, the network performance is measured in terms of the average number of flows that can be allocated in a network before rejection occurs. Through the use of the Erdös-Renyí random graph model [17] we can accurately compute the ratio of the available links as a function of the number of allocated flows in the full-mesh and we present an upper-bound on the maximum number of flows that can be allocated in the full-mesh.

For real networks, the performance analysis of network dynamics is often too complex to model. The heterogeneity of these networks and the complexity of the network protocols, including Quality-of-Service mechanisms, may prohibit the use of mathematical models, such that simulation seems the only viable option. We introduce DeSiNe, a flow-level network simulator with a special focus on Quality-of-Service, which can study and compare the performance of various Quality-of-Service implementations at a system level. In this thesis we detail the architectural and functional design and illustrate the use of DeSiNe by means of several examples. In particular, DeSiNe supports constraint-based routing and dynamic Quality-of-Service routing. The strength of DeSiNe lies in its ability to simulate classes of networks in an automated fashion. This is particularly useful when studying the performance of e.g. a new routing protocol and compare it between different classes of networks.

Tom Kleiberg

Thesis samenvatting

Flow-based Networking and Quality of Service

Gedurende de laatste twintig jaar is het Internet op grote schaal uitgerold en geïntegreerd in de maatschappij, resulterend in verregaande veranderingen in de manier waarop mensen communiceren en informatie uitwisselen. Door goedkope, snelle en betrouwbare gegevensoverdracht verbindt het Internet miljarden mensen verspreid over praktisch de gehele (ontwikkelde) wereld. Hoewel het Internet oorspronkelijk was voorbestemd als een onderzoeksnetwerk tussen een aantal onderzoeksinstituten in de Verenigde Staten, heeft het vandaag de dag een centrale rol aangenomen in ons dagelijkse communicatie en wordt het inmiddels als onmisbaar beschouwd. Gedurende de laatste tien jaar is er een migratie gaande van de klassieke telecommunicatiediensten, telefonie, radio en televisie, naar het Internet en zijn er daarnaast nieuwe diensten ontstaan, zoals online computerspellen en online overheidsdiensten. De bestaande telecommunicatienetwerken worden derhalve door het Internet vervangen en ontstaat er één enkel medium voor alle (bedrade) communicatie. Maar aangezien deze telecommunicatienetwerken zijn toegespitst op leveren van één specifieke dienst, veroorzaakt de migratie naar het Internet, waar betrouwbaarheid en schaalbaarheid de voornaamste ontwerpcriteria zijn, echter enige spanning. De lage vertragingen en kleine verliezen waarvoor de telecommunicatienetwerken geoptimaliseerd zijn, kunnen niet gegarandeerd worden door het Internet, waarvan de filosofie leunt op het "best-effort" routeren. De pakketjes in het Internet kunnen daardoor letterlijk door het hele netwerk stromen, waardoor het uitermate lastig wordt om de route van de pakketjes te voorspellen en er grote onzekerheid ontstaat omtrent vertragingen en verliezen. Om alsnog garanties te kunnen afgeven omtrent de kwaliteit van een dienst, dienen er derhalve nieuwe technieken te worden geïmplementeerd, welke vallen onder de noemer "Quality of Service" (QoS). In de literatuur zijn diverse implementaties voorgesteld die als doel hebben om QoS te bieden in het Internet. Eén van deze implementaties is gebaseerd op communiceren via "flows". In deze flow-gebaseerde communicatie worden de pakketjes van een bron naar een bestemming, via dezelfde route gestuurd, waardoor de stroom pakketjes, en dus de overdracht, beter te controleren is.

In deze thesis bekijken we de prestatie van netwerken waarbij de overdracht plaatsvindt met behulp van flows en tonen we aan dat het huidige Internet sterk onvoorspelbaar gedrag vertoont in het bezorgen van pakketjes. Aan de hand van analyses van metingen aan het Internet motiveren we het gebruik van de flow-gebaseerde communicatie zoals deze in de thesis gebruikt wordt. Uit “traceroute” metingen komt naar voren dat de routes die de pakketjes volgen, uitermate veranderlijk zijn tengevolge van dynamische processen in de routeerlaag. De levensduur van een route, zijnde de tijd tussen de eerste en de laatste opeenvolgende waarneming van de route, lijkt een “power-law” verdeling te hebben. Dit onvoorspelbaar gedrag kan de prestatie van de real-time diensten belemmeren en dient zodoende als motivatie voor de flow-gebaseerde communicatie zoals gebruikt in deze thesis.

We introduceren een nieuw model welke de netwerkprestatie beschrijft met behulp van een analogie met de wachtrijtheorie. De analyse van de netwerkprestatie wordt gecompliceerd door de vele afhankelijkheden tussen de netwerkeigenschappen. Het doel is om de netwerkprestatie te beschrijven als functie van deze eigenschappen en de kern van de complicatie te achterhalen. Het model maakt het mogelijk om de prestatie-eigenschappen van het netwerk, zoals de blokkeringskans of de maximale doorstroming, uit te drukken in termen van de netwerkparameters, zoals het aantal knooppunten. Het model beschouwt het netwerk als een zwarte doos en minimaliseert het aantal vrijheidsgraden om zodoende het aantal afhankelijkheden te beperken en de doorzichtelijkheid van het model te bevorderen. Het kleine aantal parameters stelt ons in staat om de invloed van deze parameters op de algehele prestatie te isoleren. Tevens stellen we de problemen van netwerkdynamica aan de orde, welke te herleiden zijn tot afhankelijkheden tussen de verbindingen en de invloed van verkeersfluctuaties op de algehele prestatie.

In een meer statisch scenario, waarbij aangenomen wordt dat flows een oneindige levensduur hebben, bestuderen we de maximale doorstroming in het netwerk. Hier wordt de prestatie gemeten aan de hand van het gemiddeld aantal flows dat gealloceerd kan worden in het netwerk totdat de eerste flow afgewezen wordt. Met gebruik van het Erdös-Renyí graaf model [17] kunnen we zeer nauwkeurig de ratio van het aantal beschikbare verbindingen als functie van het aantal gealloceerde flows berekenen en geven we een bovengrens voor het maximum aantal flows dat gealloceerd kan worden in de volledige graaf.

Voor echte netwerken is de prestatieanalyse van netwerkdynamica vaak te complex om te kunnen modelleren. De heterogeniteit van deze netwerken en de complexiteit van de protocollen, inclusief de QoS mechanismen, zullen het gebruik van mathematische modellen verhinderen, zodat simulatie vaak de enig overgebleven optie is. Wij introduceren hier DeSiNe, een netwerksimulator welke gebruik maakt van flows en welke een speciale focus op heeft op QoS, waarmee de prestatie van diverse QoS implementaties op systeemniveau bestudeerd en vergeleken kan worden. In deze thesis beschrijven we de architectuur en het functionele ontwerp van DeSiNe en illustreren het gebruik aan de hand van enkele voorbeelden. DeSiNe ondersteunt in het bijzonder “constraint-based” routeren en dynamisch QoS routeren. De sterkste kant van DeSiNe ligt in de mogelijkheid om via een geautomatiseerde methode klassen van netwerken te simuleren. Dit is met name geschikt bij het bestuderen van de prestatie bijvoorbeeld een nieuw routeringsprotocol en deze vergelijken voor verschillende klassen van netwerken.

Tom Kleiberg

Contents

Chapter 1

Introduction

1.1 A brief history

The first developments, that lead to the Internet and data-networking in general, date back to the 1960's with the deployment of the ARPANET. The ARPANET was initiated by the United States Department of Defense and its main goal was to develop a robust communication network that could survive the failures of some of the links and nodes. The ARPANET provided a nationwide network that connected universities and research organizations in the United States to exchange information and share resources. Many other organizations recognized the potential of the ARPANET and connected to it, such that a large network evolved that later became known as the Internet. One of the driving factors that lead to the success of the early Internet is the development of the electronic mail service (e-mail) in 1972, which provided a fast, reliable and cheap means on communication between the research communities. With the introduction of the World Wide Web in 1989, the Internet evolved to a virtually inexhaustible source of information and online services that would mean the breakthrough of the Internet to the common public. Nowadays, the Internet has taken a central role in the modern society and many sectors depend on the Internet. However, the design principles of the ARPANET, that are still prevalent in the Internet philosophy, today, are unable to provide a solid foundation for the services that have emerged over the last decade. A prominent problem that we envisage is the lack of support for real-time applications, such as Voice-over-IP (VoIP), online gaming and video streaming. The increasing popularity of these real-time applications have fueled the research on new methods to increase the reliability of end-to-end communications and provide mechanisms to guarantee service levels. Without such methods, the development of these real-time applications on Internet may be hampered and the Internet may be unable to sustain the growing demand in the future.

1.2 Routing in the Internet

The protocol that is used in the Internet for communication is the Internet Protocol (IP). The IP protocol has the task of delivering the data from the source to the destination solely based on the addresses. For this purpose, the IP protocol provides addressing schemes and methods to encapsulate the data into packets. Each packet is marked with its destination address. When a node receives a packet, it reads the destination address and determines where to send the packet next. This path selection process is called *routing*. The routing process is based on the routing tables maintained by each node, which holds a record of the routes to various network destinations. When a packet enters a node, the destination address of the packet is compared with the network destinations in the routing table. The packet is then send towards the destination that provided the best match with the packet's final destination. The IP protocol is a *connection-less* protocol, which means that the packets can be sent from source to destination without prior arrangement. The device that transmits the packets does not know if the destination can be reached and the path towards the destination is not known in advance. If the network conditions change, successive packets between end-hosts may follow different paths in the network. The major advantages of connection-less communication are scalability and robustness to failures. Scalability is achieved since nodes do not require to maintain the state of all the active connections. Each packet is examined individually and forwarded based on the information in the routing tables. Hence, the routing tables scale with the number of destinations and not the number of connections. Robustness is accomplished by the distributed path computation process. Each node acts autonomously and determines where to send a packet based on the information in the routing table. When a node fails, the routing tables of the neighboring nodes are updated and the packet is routed along a different path.

The counterpart of connection-less networking is *connection-oriented* networking. In connection-oriented networking, a connection is arranged prior to sending the messages. The sending node requests a communication session between itself and the destination node. Upon receiving the request, the network attempts to find a suitable path between the source and the destination. When a path is found, the session is established and all the packets from this session travel along this path. Each session receives a unique identifier, which is used by the forwarding nodes to determine how to forward the packets. The major advantage of connection-oriented networking is the improved control over the packets and communication sessions in comparison to connection-less networking. Each communication session can be identified and managed individually, providing a fine-grained control over the packets in the network and consequently the network performance. The disadvantage of connection-oriented networking is the cost, in terms of time and overhead, to establish the connection, introducing extra delays prior to the transmission. Furthermore, in case of failures, the connection-oriented sessions require complex recovery schemes to redirect the traffic and minimize loss and delay.

1.3 Assuring the quality of Internet services

Today, e-mail is still widely used and can be considered indispensable in modern communications, but the Internet has exceeded its original purpose and acts now as a medium that globally interconnects billions of users. The low cost and global connectivity boosted the development of new services and applications in the Internet. Multimedia services and interactive real-time applications prelude to a new generation of services in the Internet that sets special demands on the Internet performance. Characteristic to this new generation of services is the connection-oriented nature: they require a communication session to ensure that the packets arrive on-time and in a correct order. While the traditional Internet services, as e-mail, FTP (File Transfer Protocol) and the World Wide Web, are very robust and tolerant to delay, these next generation services are time-critical and highly susceptible to the network performance. Packet delay and loss generally have a detrimental effect on the quality of these services. Hence, even though the architectural design principles of the Internet have essentially remained the same the last 30 years, its use has radically altered. The increasing widespread popularity of multimedia and interactive services in the Internet has created a field of tension between the connection-less nature of IP and the connection-oriented nature of these services. Internet's traditional best-effort service model cannot provide guarantees required by the quality sensitive applications. The delivery and performance has thus far largely been assured by over-provisioning the core and access networks. Over-provisioning is commonly used to protect the network against traffic fluctuations and reduce the possibility of congested links. Over-provisoning involves dimensioning links so that their capacity exceeds the expected amount of traffic by a certain margin, which is selected to ensure that the link can absorb the expected and unexpected traffic fluctuations. Over-provisioning has proven to be a simple and effective, yet costly, method to cope with the traffic increase and surges in the Internet; the utilization of the backbone links currently seldom reaches 60 percent. However, in the nearby future, when Fiber-to-the-Home will be a fact, the wide deployment of bulky services that we envisage, will place high capacity demands on the core and access networks. Over-provisioning may not suffice anymore and network providers may be forced to use alternate measures to meet the high performance requirements of the next generation Internet services. Moreover, traffic fluctuations in over-provisioned networks may still lead to unacceptable delay variations, e.g. in the case of VoIP.

The limitations of the best-effort paradigm in the Internet to support services sensitive to network performance has lead to the introduction of the Quality-of-Service (QoS) concept. Several definitions of Quality-of-Service are being used, for example in the ITU[1] standard E.800 [27] QoS is defined as, *"The collective effect of service performances which determine the degree of satisfaction of a user of the service."* Another definition used by the ITU, standard X.902, is *"A set of quality requirements on the collective behavior of one or more objects."* Finally, the QoS Forum uses the following definition: *"Quality of Service is the ability of a network element to have some level of assurance that its traffic and service requirements can be satisfied."* The primary goal of QoS is to

[1] International Telecommunication Union

provide a better end-to-end service to selected network traffic over various technologies. Quality-of-Service involves differentiation and prioritization of users and traffic in order to guarantee a certain level of performance of a data flow. The level of performance is generally measured in terms of required bit rate, packet delay, delay variations and packet loss. To provide a satisfactory service, these measures need to stay within the bounds that are relevant for the service. For example, video streaming can tolerate a large delay, as long as the delay variations are kept small, because the traffic flows primarily from the sender to the recipient. In the case of telephony, the traffic flows in both directions and the requirements on the end-to-end delay are more strict.

The QoS Framework offers several solutions that collectively provide a better control over the network traffic. These solutions include real-time link-state-based routing, multi-constrained routing, resource reservation, connection-oriented communication and differentiation between traffic classes. Real-time link-state routing involves routing where the actual state of the links is taken into account in the path computation process. This is especially useful to circumvent congested links and divert the traffic to less occupied parts of the network. In multi-constrained routing, the path must satisfy a set of constraints instead of e.g. simply minimizing the distance. An example of a multi-constrained path is a path that has a delay of less than 300 ms *and* a packet loss rate less than 10^{-5}. Resource reservation refers to allocation of capacity on the links dedicated to selected traffic, e.g. the allocation of 3 Mbit/s on a link dedicated to a video conference call. When the resources are allocated, the dedicated traffic is not interfered by the other traffic on the link. Resource reservation is often used in combination with connection-oriented transmissions, e.g. MPLS (Multi-Protocol Label Switching) in combination with RSVP (Resource reSerVation Protocol). MPLS marks different data flows with unique labels, such that each data flow is easily identified. Prior to the transmission of data, the path is routed between the end-hosts and each intermediate node maintains a table with information with the active sessions. When a packet enters the node, the label is inspected and the packet is forwarded on the appropriate port. Consequently, all the packets belonging to the same flow travel along the same path. The connection-oriented communication facilitates the use of resource reservation on individual flows. Differentiation between traffic classes is readily implemented in most routers in the form of Differentiated Services (DiffServ). DiffServ involves an architecture where packets are inspected at ingress routers and receive a special treatment, depending on their priority. Packets belonging to voice calls and other time-critical applications, will be forwarded with higher priority than the regular best-effort traffic, such as web traffic. As a result, the prioritized traffic suffers less from queuing latencies, improving its performance.

1.4 Network dynamics

Insight into the network behavior and properties is of vital importance to facilitate any form of QoS. Many QoS mechanisms require up-to-date information on the network performance in order to make sound decisions during routing or traffic management. Obtaining this information is often difficult, owing to the large scale and anonymous nature of the Internet. The Internet consists of a constantly evolving complex hierarchical architecture where routers are grouped into Autonomous Systems (AS) that interconnect to provide global connectivity. Insights into the the Internet's characteristics are difficult to obtain, since ASes are privately managed and today's Internet lacks a central monitoring or regulating authority. Measuring the network performance is often not possible beyond the boundary of an AS. To facilitate QoS beyond AS boundaries, the involved ASes will need to establish agreements, so-called Service Level Agreements (SLA), where both parties agree on service level guarantees of selected network traffic. Within each AS, the routing protocol has the task of gathering and disseminating the state of the network and its resources throughout the network. Based on this information, the routing algorithm determines the best path.

The main research challenge lies in how to take the dynamic changes in communication networks into account to provide end-to-end QoS for individual flows. Monitoring any change in the network is simply not possible and even not desirable, because not all changes are important. Hence, insight into the dynamic properties of the network is indispensable. The term that is generally used to capture the dynamic properties of network is *network dynamics*. Network dynamics captures the patterns or processes of change or activity acting on the network.

To narrow the definition of network dynamics down, we classify network dynamics into four categories that correspond to the four layers in the TCP/IP model [42, pp. 18] [64]. The TCP/IP model is derived from the well-known Open Systems Interconnection (OSI) model [28]. The OSI model divides the network architecture into seven layers, where each layer is a collection of conceptually similar functions that provides services to the layer above it and receives service from the layer below it. The TCP/IP model is a simplification of the OSI model, as it reduces the number of layers from seven to four by combining functionally similar layers and omitting layers that in practice proved redundant. Figure 1.1 compares the OSI model with the TCP/IP model. Network dynamics takes place at each of the layers in the TCP/IP model, however the origin of the dynamics is often fundamentally different per layer.

The host-to-network layer represents the combined physical and data link layer in the OSI layering stack. The physical layer forms the actual medium, which is responsible for transforming the digital information into electromagnetic signals and transmitting it to the receiver. The data link layer in the OSI model is concerned with packet transmission, error detection and correction. The packets are converted into separate bits and then delivered to the physical layer. A well-known example of a data link layer protocol is the Ethernet standard. Combined, the host-to-network layer functionality is represented by the network topology. The topology forms the network infrastructure and is modeled by a graph. The vertices in the graph are formed by the hosts and routers, while the edges correspond to the physical links. Network dynamics on the level of the topology is often referred to as *topology dynamics*. Dynamics in the host-to-network layer are primarily related to changes on the hardware level or in the network structure. The time-scale of these changes is often very large compared to the fluctuations of the network traffic. Research in the field of topology dynamics is abundant and is often concerned with understanding the evolution of the Internet and what factors in the evolution have

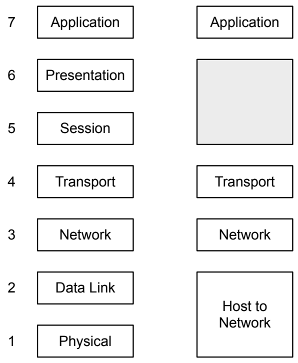


Figure 1.1: The OSI reference model versus the TCP/IP layering model.

lead to the topological properties the Internet possesses, today. Key mechanisms in this evolution are exponential growth, competition for resources and adaptation to cope with the evolving environment and maintain functionality in a demand and supply balance. A poorly designed network topology can have dramatic effect on the network performance as a whole. Insight in the topology dynamics is important to cope with the changing environment and prevent network failures.

The Internet layer implements the routing function in the protocol stack, i.e. delivering the packets from source to destination. Moreover, the Internet layer provides an addressing scheme, such that each host is identified by a unique identifier. Dynamics on the Internet layer correspond to changes in the routing plane and are often referred to as *routing dynamics*. Routing dynamics occur when the route towards a destination changes, due to a change in the topology or configuration. The change is noticed by the routing protocol and distributed throughout the network, causing network wide route changes. In the meanwhile, until the changes have fully propagated through the network, the routing plane resides in a transient state, possible causing packet delivery failures and increased latencies.

The Transport layer provides the connection-oriented session between services at the Application layer on top of the (possibly) connection-less Internet layer. The Transport layer hides the networking details from the Application layer and assures that the data from the Internet layer is passed to the correct application. The Transport layer breaks the data it receives from the Application level into datagrams and arranges a reliable transmission. Through flow-control and retransmission, the Transport layer takes care of lost packets, prevents network congestion and rearranges out-of-order packets. After the original datagram has been reassembled, it is passed on to the Application layer. Dynamics at the Transport layer are related to the behavior of the retransmission and flow control schemes in the Transport layer protocol. The dynamic properties of the Transport layer are critical to real-time applications, such as Voice-over-IP and online gaming. Since retransmission of lost packets or waiting for out-of-order packets can introduce prohibitively long delays, several Transport layer protocols are implemented

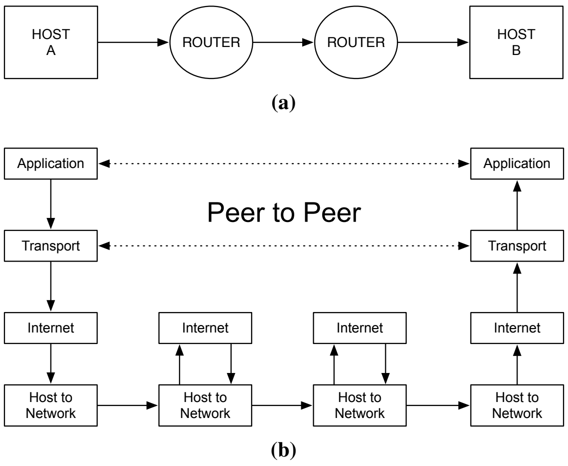


Figure 1.2: (a) Host to host networking, where A sends a message to B. The messages traverse two intermediate routers before they finally reach B. (b) The TCP/IP model stack of the host to host connection in (a). At the Application layer, a peer-to-peer session is established between the hosts A and B. In the routers, the packets are examined and forwarded at the Internet layer.

providing stateless, unreliable communication. The Transport layer then assumes that error control is not necessary or performed by the Application layer.

The Application layer implements the high-level services that directly communicate with the applications operated by the end users. The communicating partners in the Application layer are called *peers*. The Application layer is oblivious of the lower layers and assumes the lower layers arrange the delivery of the data provided by the Application layer. Network dynamics at the Application layer can be associated with the dynamics of the data streams. User applications may generate highly fluctuating traffic, for example in the case of video streaming or online gaming. Figure 1.2 presents a schematic representation of a session between end users. At the network level, the session is established from host A to host B, traversing two intermediate routers, see Figure 1.2a. Figure 1.2b depicts how the information passes through the TCP/IP layers of each of the hosts and routers, until it finally reaches the Application layer at host B.

In summary, the overall network performance is the result of many processes acting on the network. The complexity of network performance analysis is rooted in the multitude of variables and their cross-correlated nature. The interaction of the individual processes and local components of the Internet gives rise to system global behavior that cannot be traced back to the individual parts. These system properties that derive from the collective behavior are generally referred to as *emergent properties*. They are global features and capabilities which are not specified by network design parameters and are hard to predict from the knowledge of its constituents. Examples are the hopcount, the diameter and the centrality measures, such as the node and link betweenness [43, Sect.

6]. Computation of such measures is sometimes possible for networks in equilibrium if they can be modeled by simple mathematical model, such as the Erdös-Renyí random graph. But little is known about these measures when the load varies and the network is in a transient state.

1.5 Structure of the thesis

The thesis can be divided into three parts, namely measurement, modeling and simulation, that are all positioned around a central theme, which is evaluating the network performance where traffic is modeled at the flow level. The goal of the thesis is to provide insight into the difficulties of analyzing the network performance and propose a model that expresses the network performance in terms of the network parameters, that are given by design. As pointed out earlier, the performance analysis is troubled by the many dependencies inside the dynamic network environment. In general, it is not possible to gather the global network behavior from the parameters, such as the number of nodes or links. However, through the use of a simplified model with a small number of parameters, we isolate the effects of several parameters and study their effect on the global network performance. Figure 1.3 depicts the three parts and how they are incorporated in the thesis.

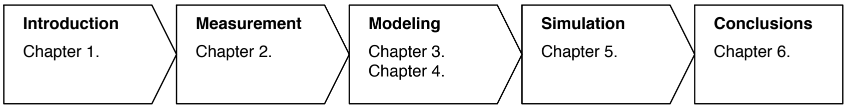


Figure 1.3: Structure of the thesis.

Chapter 2 motivates the flow-based approach that is used throughout the thesis and exposes the highly dynamic behavior of the routing plane in the Internet. Through the analysis of measurements on Internet paths we find that the way that packets are routed in the Internet can be highly unpredictable. These uncertainties may cause unexpected delays and out-of-order delivery of packets yielding impairments in the performance of time-critical services. Hence, it is generally expected that for the accomplishment of Quality-of-Service in the Internet, the traffic belonging to time-critical services must be routed in the Internet using a flow-based approach. When flows are used, the packets are forced along the same path, which largely eliminates the dynamics in the routing plane for the particular flow and provides a better control over the packets. The routing dynamics observed in Chapter 2 are linked with Self-Organized Critical behavior (SOC), which explains the heavy tail in the path duration and the $1/f$-noise spectrum. SOC is characterized by the presence of “avalanches”, where a single event can cause an avalanche of other events. In this case, the change of a single route can cause a great number of changes of other routes which are directly or indirectly dependent on this route. The remainder of the thesis is centered around flow-level models of communication.

Chapter 3 introduces a new model that describes the network performance by using an analogy with queueing theory. The model facilitates to express emergent performance characteristics, such as the blocking rate of traffic flows and the maximum throughput of the network, in terms of network parameters that are given by design, without making use of simulations or difficult computations. The model considers the network as a black box and minimizes the degrees of freedom in order to reduce the dependencies and improve the comprehensibility of the model. In Chapter 3 we also address the difficulties of studying network dynamics, which can be traced back to the dependencies between the links and the importance of traffic fluctuations in the network on the overall performance.

In Chapter 4 we study the maximum throughput of a network in a more static scenario where flows are assumed to have an infinite duration. Now, the network performance is measured in terms of the average number of connections, or flows, that can be allocated in a network before rejection occurs. Through the use of the Erdös-Renyí random graph model [17] we can accurately compute the ratio of the available links as a function of the number of allocated flows in the full-mesh and we present an upper-bound on the maximum number of flows that can be allocated in the full-mesh.

As shown in Chapters 3 and 4, the performance analysis of dynamic networks is often very complex. The heterogeneity of real networks and complexity of network protocols, including Quality-of-Service mechanisms, may prohibit the use of mathematical models, such that simulations seem the only viable option. In Chapter 5 we introduce DeSiNe, a flow-level network simulator with a special focus on Quality-of-Service. Chapter 5 details the architectural and functional design and presents several examples that illustrate the use of DeSiNe. The purpose of DeSiNe is to study and compare the performance of various Quality-of-Service implementations at a system level. In particular, DeSiNe supports constraint-based routing and dynamic Quality-of-Service routing. The strength of DeSiNe lies in its ability to simulate classes of networks in an automated fashion. This is particularly useful when studying the performance of e.g. a new routing protocol and compare it between different networks. Finally, we present the conclusions in Chapter 6.

A large part of the thesis consists of unpublished work. In particular the modeling part, which forms the main body of the thesis, deserves explanation on why it has not been published, yet. The modeling work was initiated by the simple question: "Can we find a simple model that computes the average number of flows in a network?" Chapters 3 and 4 will demonstrate that no simple answer to this simple question exists, unless we make strong assumptions about the network traffic and topology. Even in the simple case, where we have managed to empirically find a model that describes the network behavior, we have not succeeded to establish a mathematical proof to sustain our findings. Our efforts to mathematically model the network behavior stranded due to the complexity that is introduced by the dependencies between the links. Chapter 4 and Appendix B illustrate the effect of ignoring these dependencies. The submissions of this work were indeed rejected primarily due to the lack of such a proof. However, we feel that the work presents an attractive approach and solution to a very complex model and may serve as an input for future work on the topic.

Chapter 2

The unpredictable behavior of Internet paths

2.1 Introduction

Interactive services in the Internet place strict bounds on the performance of end-to-end paths. Packet delay, delay variations and packet-loss have a severe impact on the quality of the Internet service and therefore it is important that end-to-end communication is reliable and predictable. The ability to control the end-to-end performance is seriously complicated by the connectionless nature of the Internet Protocol and the lack of any widely deployed Quality-of-Service implementation. As a consequence, the packets in the Internet are exposed to erratic network performance due to traffic fluctuations and routing dynamics. Traffic fluctuations can lead to temporary congestion of the router buffers, causing delay variations between the packets and packet-loss. Although congestion occurs very frequent in the Internet, measurements indicate that the traffic fluctuations are highly transient and the impact on the service performance often remains within bounds [69]. Routing dynamics correspond to the process where routing messages are propagated between sets of routers to advertise a route change. When a network event, for example a link or node failure, causes a route change, the network temporarily resides in a transient state while the routing tables of other routers are being updated. Routing dynamics contribute to most prolonged path disruptions and can last as long as 10 minutes, leading to serious degradation of Internet services [52, 55, 69, 70, 78].

In this chapter we study the dynamics of Internet paths with the use of an extensive dataset of traceroute measurements. In particular, we analyze how many routes are used between two end-hosts and how long a route remains operational. We regard the Internet as a “black box” and consider the path dynamics as the result of a collective behavior that organizes the thousands of autonomous nodes into a single complex system. Route changes correspond to perturbations in the Internet and we find that the statistical properties of the measured perturbations hint towards self-organized critical behavior in the

Internet. Self-organized critical systems are often found in nature and other complex systems and by comparing the characteristic features of SOC systems we argue that the Internet also exhibits self-organized criticality. The presence of the SOC mechanism in the Internet indicates that routing in the Internet is unpredictable in the sense that routes can change unexpectedly. The inter-event time between two routing events has no typical value and is widely varying. Packets associated with the same stream may follow entirely different routes, introducing a wide spread between the delivered packets and large delay variations. For the increasing number of real-time applications, such as interactive gaming, IP telephony, video and others, these variations can lead to dramatic degradation of the experienced quality. From the observations in this chapter, we argue that it is recommended to control the path, e.g. by means of MPLS, to assure the quality of service of these interactive services.

Furthermore, we find that permanent changes in the Internet cause the breakdown of existing end-to-end routes and the discovery of new ones. On average, the number of discovered routes increases linearly in time at a fixed rate.

This chapter is organized as follows: in Section 2.2 we briefly introduce self-organized criticality and present some features that are typical for SOC systems. Next, the measurements are described in Section 2.3. Section 2.4 contains the observations, followed by a discussion in Section 2.5. Section 2.6 presents an overview of related Internet measurement projects. Finally, Section 2.7 presents a summary of the chapter.

2.2 Self-Organized Criticality

A definition for self-organization used by De Wolf and Holvoet [72] is, "Self-Organization is a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control." One of the key features of self-organizing systems is the adaptability or robustness with respect to changes. The system should be able to cope with external influences and maintain its organization autonomously. Self-organized criticality was introduced by Bak *et al.* [10] as a property of a system that, through a self-organized process, always evolves to a critical state, regardless of the initial state of the system. A common feature observed in SOC systems is the power-law temporal or spatial correlations that can extend over several decades. SOC systems organize into clusters, with a scale-free spatial distribution, with minimally stable, "critical", states. A perturbation in that system can propagate through the system at any length scale from a local change to an avalanche by upsetting the minimally stable clusters. The magnitude of the perturbations is only limited by the size of the system. The lack of a characteristic length leads directly to the lack of a characteristic time for the resulting fluctuations. Hence, a power-law distribution arises for the lifetime distribution of the fluctuations in the system. From the inter-event time of the fluctuations a time signal can be constructed, where the perturbations are modeled as a series of Dirac pulses [33],

$$I(t) = \sum_k \delta(t - t_k) \tag{2.1}$$

where t_k corresponds to the time of the k-th event. The time signal can now be transformed into the frequency domain and the power-spectrum of (2.1) is found as,

$$S(f) = \lim_{T \to \infty} \frac{2}{T} \left\langle \left| \sum_{k=k_{min}}^{k_{max}} e^{-j2\pi f t_k} \right|^2 \right\rangle \tag{2.2}$$

where T denotes the whole observation time and k_{min} and k_{max} are the minimal and maximal values of the index k in the interval of observation. The brackets $\langle \ldots \rangle$ denote the averaging over different realizations of the process. One realization can correspond to the measurements or observations of the system at one particular time or from one particular perspective. By averaging the measurements or observations of the same system taken from different perspectives or at different times, we can acquire the average behavior of the system [1]. The averaging is necessary since we are only interested in the process that leads to the power-spectrum and not a realization of the process. From (2.2) it follows that the estimation of the spectrum improves as the observation time increases. It can be shown that a power-law distribution in the inter-event time leads to a power-law spectral density [32],

$$S(f) \propto 1/f^{\varphi} \tag{2.3}$$

where φ is typically close to 1. The power-law spectrum in (2.3) is referred to as $1/f$ noise and is widely found in nature. The $1/f$ noise can be seen as a measure of the complexity of the system. The $1/f$ noise phenomenon is often observed in large systems that act together in some connected way. The $1/f$ noise can arise as the result of the coherence between events, the so-called long-range-dependence. It is also seen as a naturally emergent phenomenon of the SOC mechanism [10, 15, 74].

The examples of SOC systems in nature are abundant. The sandpile and forest fire models are probably the most well-known [9]. Several works have related the SOC mechanism and $1/f$-noise to the Internet. The self-similarity and long-range dependence of traffic patterns often reported in the Internet are considered related to the $1/f$ noise phenomenon [18, 71]. Csabai [16] measured the round trip times of packets and showed that the correlation between the round-trip times produces $1/f$ noise in the power spectrum. Ohira *et al.* [50] and Solé *et al.* [61] demonstrated that computer networks with self-organizing behavior show the maximum information transfer and efficiency at the critical state. In addition, Solé *et al.* demonstrate that near criticality, the network performance shows the highest variability in terms of packet latency. In this work we study the inter-event times of route changes in the Internet and argue that unpredictability and instability of Internet routes may be related to SOC.

[1]Here we assume that the average behavior of the system is stationary and does not change during the observation intervals.

Route result	probes	% probes	routes	% routes
Successful delivery	30986955	98.90	13496	43.19
Persistent forwarding loop	24732	0.07	1422	4.55
Transient routing loop	411	0.00	231	0.73
Infrastructure failure, destination not reached	36240	0.12	2561	8.20
Packet delivered at non-listed IP address	3995	0.01	619	1.98
Anonymous reply, destination reached	278066	0.88	12920	41.34

Table 2.1: Statistical overview of the pathologic routes and probes in $\mathcal{D}$.

2.3 Measurements

2.3.1 Methodology

The measurement apparatus consists of a set of testboxes that are deployed by RIPE as part of the Test Traffic Measurements service (TTM) project[2]. A testbox measures the Internet paths towards a set of pre-determined destinations by repeatedly probing the router-level path from source to destination. The path is measured by the traceroute tool, which sends probes to the destination host and infers the forwarding path by analyzing the response from the intermediate hosts. The traceroute messages are transmitted at exponentially distributed random intervals, with on average of 10 messages per hour from one source to one destination. Beside the IP path, the AS path is obtained by inspection of BGP[3] data and matching each address in the IP path with an AS prefix[4]. In the translation from the IP route to the AS path, the duplicate AS entries are removed, which result from the multiple IP hops in one AS. Hence, the AS path consists of a list of ASes that the route traverses, where each AS number can only appear once.

Each source maintains its own list of destinations, which is a subset of the other testboxes deployed by RIPE. The testboxes are placed at customers' sites, typically ISPs residing in various countries, just behind their border routers. Since the enrollment of the TTM project in 1999, the number of testboxes has increased from approximately 30 up to around 150, today. Between the roughly 160 testboxes that exist, or have existed, around 10,000 source-destination pairs have been registered, where pair (A,B) is different from (B,A). Hence, the data that is available from 1999 does not include all the testboxes available today. In addition, testboxes can be temporarily offline for managerial or other purposes and several testboxes have disappeared completely, indicating the termination of the TTM service at the customer's site. The configuration and (geo)location of the testboxes is very stable. The IP address of a testbox seldom changes and only few testboxes are indeed discontinued. The stability of the testboxes facilitates the measurement

[2]A detailed technical description of the design and features of the TTM testboxes can be found in [21] and on the TTM website, http://www.ripe.net/projects/ttm/.

[3]BGP (Border Gateway Protocol) is the routing protocol that is used to route packets between different ASes. By comparing the IP address of the packet with the BGP prefixes of the destination, the destination AS can be extracted.

[4]The AS path information has not been recorded in the initial phase of the project and is available only from the beginning of 2003.

trustworthiness in the sense that the observations indeed reflect the network state and not so much the measurement setup. The high fidelity of the measurements and the extend of the observation time make the TTM measurements an excellent set to study long-term Internet path dynamics. In fact, it is the only publicly available dataset containing router-level information for this time span with such high accuracy.

2.3.2 Dataset

Section 2.2 emphasizes the importence that the observation time is long with respect to the interval times between subsequent events. On the other hand, increasing the observation time reduces the number of usable source–destination pairs, because less testboxes were available in the early stage of the project. Furthermore, we must also consider that measurements between source–destination pairs can be inoperative: the list of destinations of each testbox can change over time and testboxes can be temporarily offline. To decrease the influence of these dynamics in the analysis, the set of usable source–destination pairs is restricted by the maximum time a source–destination pair was inactive. Increasing the stringency on the outage restrictions will reduce the number of usable source–destination pairs. Hence, a trade-off must be made between the number of usable source–destination pairs versus the observation time and outage restrictions. The resulting dataset consists of the traceroutes between all the source-destination pairs that were active the entire period from January 1, 2003 until January 1, 2008, where any outage between a source–destination pair is restricted to maximally 28 days. Source–destinations pairs of which both source and destination belong to the same AS are excluded from the dataset. The dataset, that we will denote by $\mathcal{D}$, contains 64 source-destination pairs out of a set of 10 testboxes located within 8 different European countries.

2.3.3 Measurement artifacts

The anonymous and dynamic nature of the Internet inherently adds noise to the measurements which consequently incurs errors in the analysis. These measurement artifacts include persistent forwarding loops, transient routing loops, infrastructure failures and anonymous replies by the intermediate routers. Persistent forwarding loops are generally related to mis-configured routers, while transient forwarding loops are often a manifestation of routing dynamics. Infrastructure failures lead to pre-mature termination of the traceroute probe. Anonymous replies are due to unresponsive routers or rejected probes. For a detailed discussion on these pathologies we refer to [52, 53]. We have also found several cases where the packet was not delivered at the correct destination address. This may be the result of erroneous routing or a configuration problem in the measurement setup such that the packet is delivered at an unknown IP address. Finally, we would like to mention the presence of “third-party” addresses as a source of noise in the traceroute measurements [24]. But since such occurrences are rare [24] we disregard them in our analysis. The classic traceroute tool developed by Van Jacobson [29] and used in the TTM project is unable to detect such pathologies and different modifications have been developed to address short-comings of the classic traceroute tool [7,20,67]. These recent changes were not available in 1999 and are therefore not included in the TTM project. Routes which exhibit the above mentioned artifacts have been filtered from the dataset before processing. Table I presents an overview of the frequency of the measurement artifacts. From Table I, we can deduce that the pathologies contribute to slightly more than 1 percent of the measured probes. Hence, we argue that the influence of the pathologies on the accuracy of the measurements is sufficiently small to ignore.

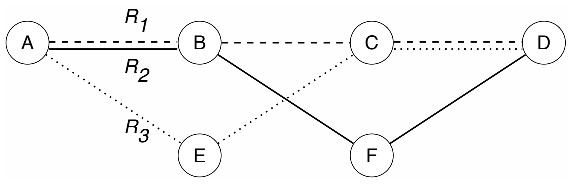
2.3.4 Route fluttering

Between the successful routes, there is also a significant fraction of aliasing routes. Route aliasing can be a manifestation of load-balancing, where a group of packets that are traveling between the same source and destination traverse different routes. The packets are separated based on their packet header or simply in a round-robin fashion. As a result, the samples of the IP path in the presence of load-balancing routers will consist of rapidly alternating routes, so called fluttering routes [52]. Load-balancing is the result of a decision process inside one router, it does not involve the advertisement of any routing updates to neighboring routers and does not affect the state of the routing tables[5]. Hence, load balancing does not contribute to the routing dynamics and we will handle fluttering routes as a single route, i.e. as if the packets were sent along one route.

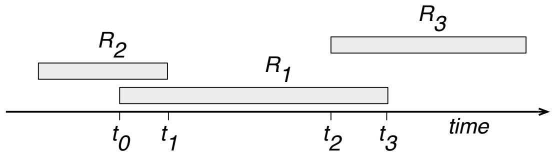
To cope with fluttering routes, we will adopt the heuristics presented by Paxson [52] to classify fluttering routes: two routes are considered the same when the paths have an equal length and differ at maximally one consecutive hop. When routes are considered the same route, the samples of all the routes are aggregated as if they were samples from one route. E.g., if route R_1 is observed 1000 times and route R_2 is observed 500 times, then the aggregated route has 1500 observations. Figure 2.1a provides an example; routes R_1 and R_2 differ only at the third hop. According to the heuristics, the addresses corresponding to nodes C and F are considered the same, such that eventually routes R_1 and R_2 are considered the same route. More advanced situations have been found, however, which we also address here. Figures 2.1 and 2.2 sketch examples of real data. In Figure 2.1a we can distinguish three routes, R_1, R_2 and R_3. The routes R_2 and R_3 differ at two consecutive hops and would count as different routes. However, due to the presence of route R_1, the nodes B and E are considered to belong to the same router, such that eventually all three routes are combined to a single route. Another example is given in Figure 2.2, where the there are 4 different routes. The routes R_1 and R_4 are different at two consecutive hops, and would not classify as identical route. The routes R_2 and R_3 make that the nodes B and E in Figure 2.2a are treated as the same node BE, yielding the situation depicted in Figure 2.2b. Eventually, the formation of the new node BE makes routes R_1 and R_4 to be the same.

A matter that arises when studying routes at large time scales is how routes based on the IP path information would classify as fluttering, but in fact do not coexist in time. Figure 2.1b illustrates the situation by means of the example in Figure 2.1a. Between t_0

[5]Here we assume that the routing tables are not affected by the actual traffic due to some form of traffic engineering.



(a)



(b)

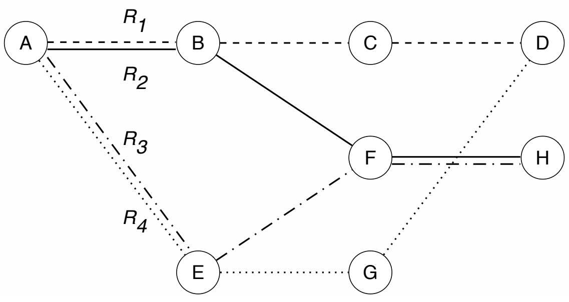
Figure 2.1: (a) Schematic representation of the three routes R_1, R_2 and R_3 that cross nodes A, B, C, D, E and F, where a node corresponds to one IP address in the IP path. Route R_1 differs one hop from R_2 and one hop with R_3, but route R_2 and R_3 differ at two consecutive hops. Although R_2 and R_3 differ at two hops, they are considered the same route due to the presence of R_1. (b) Example of transient behavior of R_1, R_2 and R_3. Route R_1 is first observed at t_0 and last at t_3. Route R_2 is last observed at t_1 and R_3 is first observed at t_2.

and t_1 both the routes R_1 and R_2 are being observed, while between t_2 and t_3 both the routes R_1 and R_3 are coexistent. However, $t_1 < t_2$ and routes R_2 and R_3 are never observed simultaneously. The issue here is how to act if the routes do not overlap in time, i.e. when $t_1 < t_0$ or $t_3 < t_2$. Although the IP path information would suggest route fluttering, the transient analysis disagrees. In this work, oscillating routes are considered the same if both the IP path requirement and the requirement of overlapping observation periods hold.

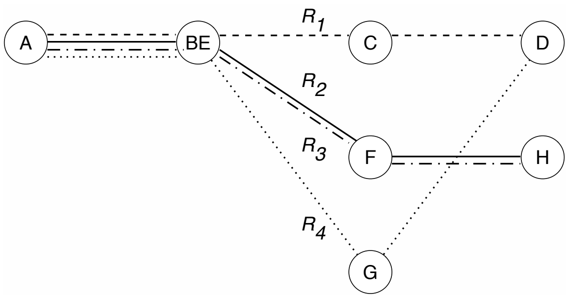
Prior to filtering the fluttering routes, the dataset $\mathcal{D}$ contained 13496 routes. Afterwards 8612 routes remained.

2.3.5 Metrics

The routing dynamics in the Internet can be measured by means of the length and coherence of the time intervals between subsequent route events. A route event can affect one or more routers on the route between a source–destination pair, such that the route is changed. Hence, the time-interval between two route events corresponds to the time that a route is operational, which we will denote by the *route duration*. In the RIPE measurement setup, the path between a source–destination pair is sampled at independent, exponentially distributed random intervals with an average of 360 seconds. The exact time of the traceroute call is rounded to seconds and recorded in the database. When the same path is sampled multiple, consecutive, times, only the time of the first and last call are recorded. Hence, there is no accurate information of the exact time of each call, only the number of calls and the start and end time of the sequence of calls. The number of



(a)



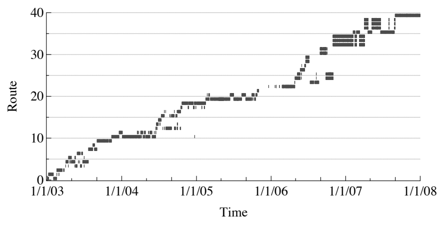
(b)

Figure 2.2: Schematic representation of four different routes, R_1,R_2,R_3 and R_4. Each node corresponds to a different IP address in the IP path. In (a) the routes R_1 and R_4 differ at two consecutive hops and are initially considered different routes. Due to the presence of routes R_2 and R_3, the IP addresses of the nodes B and E are considered to belong to the same router and treated as a single node, BE, as shown in (b). Routes R_1 and R_4 eventually classify as identical routes.

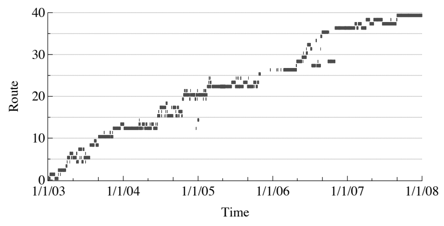
calls qualifies as an alternative measure for the duration of the path, hence the route duration is defined as the number of successive occurrences once it is selected. Due to the Poisson measurement times the PASTA property applies and the sampled time averages indeed reflect the real time averages [52]. The sampling rate prevents us from detecting the typically highly transient failures at the data plane due to congestion, which are typically in the order of seconds. Yet, the average inter-arrival time is sufficiently small to detect the slow route dynamics, which can last many minutes.

2.4 Observations

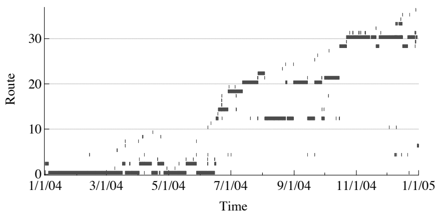
The RIPE TTM measurement setup is well suited for both short-term and long-term analysis of Internet routes. It allows the study of Internet paths at widely varying timescales, ranging from minutes and hours up to years. Figure 2.3 shows the IP routes between a typical source–destination pair, considered over a long period of five years. Only the 40



(a) Before filtering the route oscillations.



(b) After filtering the route oscillations.



(c) Detail of 2004 after filtering the route oscillations.

Figure 2.3: Example of a set of observed routes between one source–destination pair, specifically from a testbox in Amsterdam, The Netherlands, to a testbox in Geneva, Switzerland. The horizontal axis represents the time axis, ranging from January 1, 2003 to January 1, 2008. The vertical axis indicates the routes between the source–destination pair. Only the 40 most prevalent routes have been shown out of a total of 213 routes for (a) and 158 for (b). The routes are sorted from bottom to top in order of their first appearance. The marked areas indicate when a particular route is operational. In (a) the raw routes are shown, before filtering the oscillations, where at several times multiple routes are operational simultaneously. In (b) the oscillating routes have been merged and are then considered as one. This reduces the number of unique routes and changes the prevalence of the existing routes. Hence, the routes presented in (a) and (b) may be different, e.g. route 10 in (a) is a different route than route 10 in (b). (c) shows all the observed routes for the year 2004 after filtering.

most dominant routes are displayed. Figure 2.3 depicts when routes are operational and demonstrates the presence of several phenomena in the Internet, which we will discuss here.

First, Figure 2.3 exemplifies that route fluttering is a common artifact in the Internet and it is important to consider these oscillations in the analysis. Figure 2.3a shows several cases of route fluttering, e.g. at the end of 2006 equivalent routes appear that overlap in time. After resolving the equivalent IP paths, the fluttering routes have been merged, as illustrated in Figure 2.3b. The routes presented in Figure 2.3b are considered unique routes and a route change indeed corresponds to a change in the routing table and not to load-balancing.

Second, Figure 2.3 illustrates that route events occur frequently and at all time scales.

Most of the time a single route prevails between a source–destination pair. For example, Figure 2.3b demonstrates that for the first six months of 2004 only two routes are used predominantly. However, the dominant route is sometimes interrupted, where the interruption can be very brief or sometimes last for days. The inset in Figure 2.3b shows all the routes between this source–destination pair for the whole 2004. From the inset we can gather that many routes are observed only occasionally and briefly. These routes can be the result of temporal route failures or routing dynamics.

Finally, we can conclude from Figure 2.3 that the routes have a limited lifetime. Between a route's first and last appearance many other routes can be operational, however, in all cases the route eventually disappears and is never seen again. The time between a route's first and last appearance varies widely per route: several routes are seen for months while other routes, although not displayed in Figure 2.3, are seen once. The restriction on these lifetimes is a consequence of the evolution taking place in the Internet. The Internet is constantly evolving and the actors in the Internet are continuously trying to optimize their position to increase their revenues and/or performance. Frequent changes in the peering relations between ASes result in the birth of new AS paths and the death of existing ones. At the intranet level, ASes need to manage and improve their networks to accommodate for the increasing demand on network resources [51]. This includes addition of routers and reconfiguration of their intranet. The router-level path changes, but the AS path remains the same. Consequently, multiple IP routes can be observed for a single AS path.

Figure 2.4 presents the measurements on the number of *unique* routes learned since January 1, 2003 till January 1 2008, on both the AS- and IP-level, averaged over the source–destination pairs. We consider an IP route *unique* when there exists no other route with the same sequence of IP addresses. Similarly, we consider an AS path *unique* when there exists no other AS path with the same sequence of AS numbers. The measurements in Figure 2.4 exhibit a remarkable linear behavior on both the AS and IP level, which is in agreement with the findings in [51]. If $R(t)$ represents the number of routes that are learned as a function of the time t, then according to Figure 2.4 we can write $R(t) = \alpha t - g(t)$, where $g(t) = o(t)$ for large t. Hence, $\lim_{t\to\infty} R(t)/t = \alpha - \lim_{t\to\infty} g(t)/t = \alpha$, which corresponds to the "rate" at which new routes are discovered. Figure 2.4 shows that the discovery rate remains fairly constant for the entire observation period. A new IP route is learned approximately every 14 to 15 days, on average. The first few months of 2003 the discovery rate is slightly higher due to a learning phase. At the AS-level, a new route is discovered every 38 to 39 days, on average.

Figure 2.5 shows the complementary cumulative distribution function (CCDF) of the duration of a route at the IP-level. The duration is measured as the number of successive occurrences of the same route until a new route becomes operational or the routing fails and an error message is received. We have selected five source–destination pairs that do not have any end-points in common and for which the routes show typical behavior. For each pair we have computed the route duration and plotted the CCDFs in Figure 2.5a. Figure 2.5a demonstrates that the shape of the distributions is very similar. The fact that the source–destination pairs do not have any node in common argues that the behavior that we observe in Figure 2.5a is a true feature of the Internet and not an artifact of the

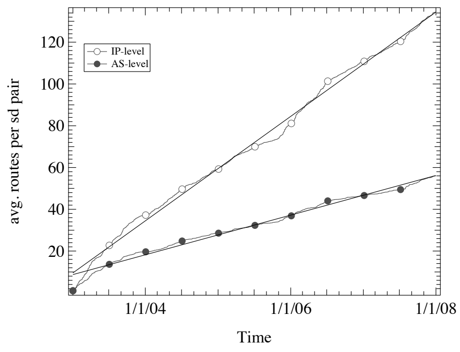
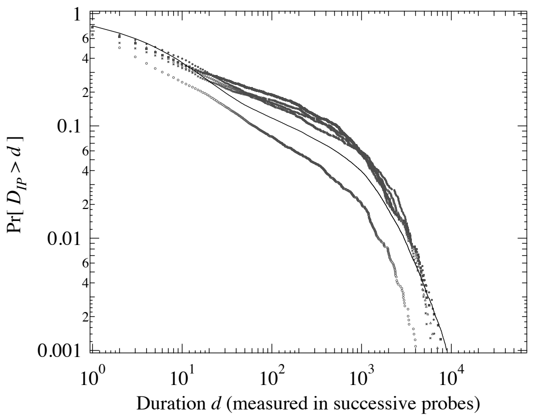


Figure 2.4: The average number of discovered routes between the source–destination pairs at both the AS- and IP-level, counting from January 1, 2003. The measurements have been fitted with a line. The fit at the IP-level resulted in a discovery rate of 14.6 days per route, while at the AS-level the fit provided 38.4 days per route.

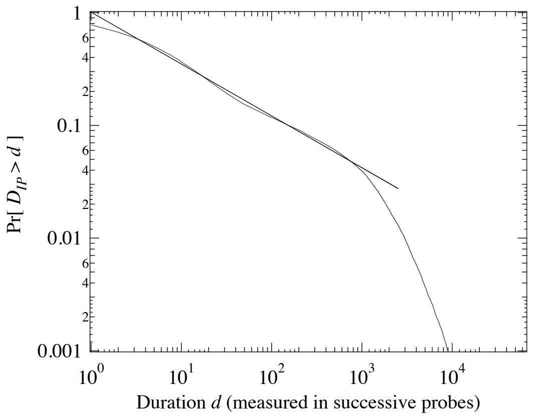
dataset. The aggregate result of the entire dataset, which is obtained by accumulating the results for all the source–destination pairs, is presented by the solid line in Figure 2.5a. The aggregate distribution strongly resembles that of the individual pairs, which indicates that the mean reflects a real property and can be considered as the average behavior of any source–destination pair in $\mathcal{D}$.

Figure 2.5a illustrates that the duration relatively closely follows a power-law distribution for the first three decades. At approximately $t = 10^3$ the distribution is cutoff followed by a steep decline. The CCDF of the aggregate result has been fitted with a power-law and is presented in Figure 2.5b. The power-law exponent of the fit is $\gamma \approx 1.46$. Power-law distributions with an exponent $\gamma < 2$ exhibit extreme behavior and have a divergent mean [48]. When sampling a power-law with extreme behavior, the mean is determined by the sample with the highest value, which will go to infinity when the number of samples becomes large. In real-world systems the mean is finite: the distribution is cut off in the tail because the system has a limited size. The measurements on the route durations are restricted by the limited sample space of the Internet (we cannot sample the entire Internet) and possibly the limitations of the measurement architecture (e.g. the limited uptime of testboxes due to managerial purposes, etc.)

Figure 2.5b provides insight into the duration of the routes once they are established, but does not reveal whether the established route appears often or if the route is dominant. A route is considered dominant, when it is preferred above the other routes. The dominance of a route is computed as the number of route occurrences of that route with respect to the total occurrences of all the routes within some time interval. Figure 2.6 correlates the duration of a route with its dominance. Figure 2.6 shows the probability that when a route is observed, the observation lies within an interval where the route is observed d successive times and has a dominance δ. Figure 2.6 reveals that a large fraction of the observations fall within $d \leq 2$, which is also shown in Figure 2.5b. However,



(a)



(b)

Figure 2.5: The complementary cumulative distribution function of the duration of the IP routes on log-log scale. The duration is measured as the number of successive occurrences the same IP route is measured. In (a) the solid line indicates the aggregated result, which combines the measurements from all source–destination pairs into a single distribution. The markers represent the individual distributions for five source–destination pairs with typical behavior. The measurements suggest that the duration closely follows a power-law distribution. In (b) the aggregated measurements have been fitted with $F(d) = Cd^{-\gamma+1}$. The fit result is presented as the straight line, the power-law coefficient is found as $\gamma = 1.46$.

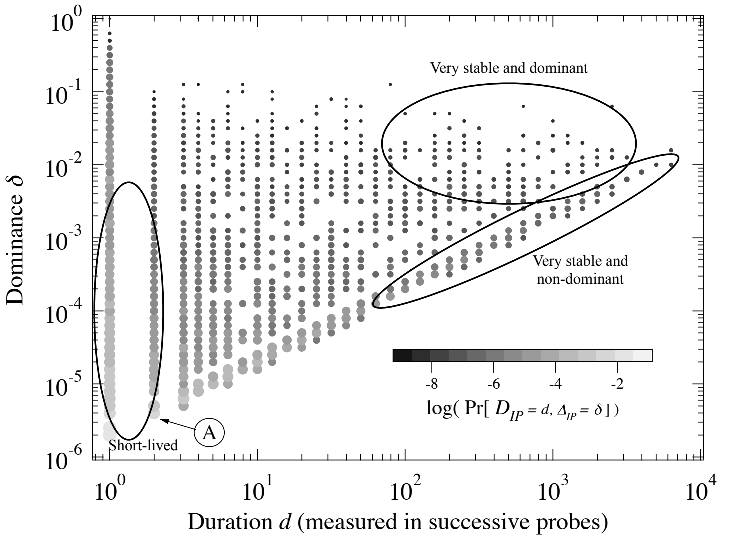


Figure 2.6: Each route observation is measured at the IP level and the route duration is correlated with the route dominance. On the x-axis the route duration is plotted against the route dominance on the y-axis. The size and opacity of the dots designate $\Pr[D_{IP} = d, \Delta_{IP} = \delta]$, which is the probability that an observation of a route has some duration d and dominance δ. As the dots are larger and less opaque, more traceroutes are measured with the corresponding duration and dominance. Note that the axis and the dots are presented on log-scale. For example, the point A in the figure represents the probability that a route observation corresponds to a route with dominance $\delta \approx 0.3 \cdot 10^{-6}$ and to a route duration $d = 2$.

Figure 2.6 also shows that from the short-lived routes, the majority has a small dominance meaning that they are seen seldom or only once. Hence, these routes are due to the highly transient nature of the Internet and a typical manifestation of routing dynamics. Furthermore, we can recognize a line in Figure 2.6 that forms the lower bound on the dominance w.r.t. the route duration. Since the dominance is measured as the sum of the observations, the increase in duration implies an increase in the dominance. A significant fraction of the observations are situated around this lower bound. They correspond to observations of routes that are only seen once, but have a prolonged duration.

Figure 2.7 presents the CCDF of the route durations at the AS-level. Similar to Figure 2.5a we have plotted the aggregate distribution together with the individual distributions of the same five source–destination pairs as in Figure 2.5. The distribution at the AS-level resembles the shape of the IP-level distribution. For the first three decades the CCDF follows a power-law, followed by a steep cutoff. The distribution has been fitted with a power-law distribution, presented in Figure 2.7b. The power-law exponent was found as $\gamma = 1.17$. Compared to the IP routes, the power-law exponent at the AS-level is smaller, yielding a fatter tail. At the AS-level there are more long-lasting routes, which can be explained by considering that a route change at the IP level does not necessarily lead to a different AS path. This observation agrees with our finding from Figure 2.4. The discovery rate of AS paths is smaller than that of the IP routes, implying that multiple IP routes exists per AS path. The average duration of an AS path must therefore be greater than that of the IP routes.

Now we will compute the spectral density of the routing dynamics and examine the existence of $1/f$ noise. The time signal (2.1) requires the time of occurrence of the routing events, which can be constructed from the route duration measurements by,

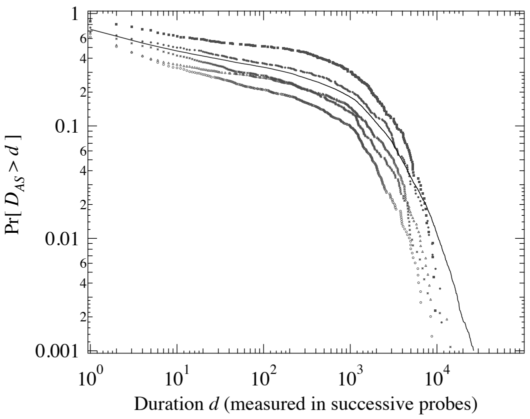
$$t_k = \sum_{j=0}^{k-1} D_j \tag{2.4}$$

where D_j is the duration of the route after j route changes. We have computed $I_{IP}(t)$ separately for all the source–destination pairs at the IP-level. The power-spectrum $S_{IP}(f)$ is then computed by averaging the transformed signal between the source–destination pairs. The result is presented in Figure 2.8. The power spectrum shows a steep decline in the first three decades. The spectrum has been fitted with (2.3), where the value for φ was found as $\varphi = 0.85$. The fit demonstrates the presence of $1/f$ noise.

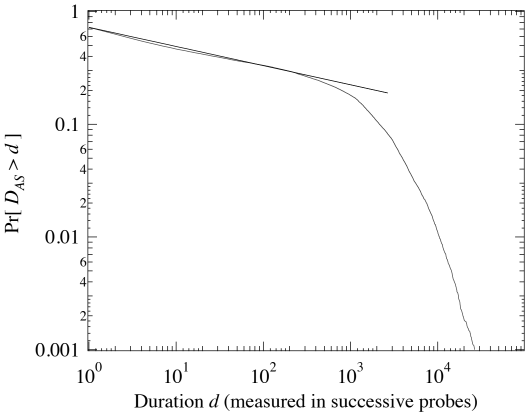
2.5 Discussion

The extreme power-law behavior observed in Figures 2.5 and 2.7 is often seen in real world systems [48]. When a distribution possesses such a heavy tail, the expected value and the variance tend to infinity. In practice this implies that the observed measure is highly unpredictable. In our case it means that the route is unpredictable and packets associated with the same stream may follow (many) different routes. Figure 2.6 underlines the presence of highly transient routes, likely due to routing dynamics. At first glance, the unpredictable nature of the routes seems remarkable. The packets in the Internet are routed by the connectionless Internet Protocol, where each packet is routed individually without establishing a connection prior to the transmission. However, most of the lower layers, the datalink and physical layer, use connection-oriented technologies, such that the route towards the destination is known in advance. The changes of the topology are relatively sparse and slow, because it is often manually managed. Hence, one would expect more stable paths. Yet, the measurements indicate that route changes occur at all time scales. Such frequent changes can have negative impact on streaming services that rely on packets arriving on time and in the correct order [66, 70]. At the same time, the high variability demonstrates that the Internet is resilient and fastly adapts to changes.

The power-law spectral density and power-law behavior of the inter-event time can be seen as a manifestation of self-organized criticality. Bak *et al.* [9, 10] argue that SOC naturally arises in interactive dynamical systems with many degrees of freedom. The Internet is clearly a dynamical system with self-organizing behavior. The notion of SOC is a plausible explanation of the observed dynamics.



(a)



(b)

Figure 2.7: The complementary cumulative distribution function of the duration of the routes at the AS-level printed on log-log scale. The duration is measured as the number of successive occurrences of the same AS path. In (a) the solid line indicates the aggregated result, which combines the measurements from all source–destination pairs into a single distribution. The markers represent the individual distributions for the same five source–destination pairs as in Figure 2.5a. The measurements suggest that the duration closely follows a power-law distribution. In (b) the aggregated measurements have been fitted with $F(d) = Cd^{-\gamma+1}$. The fit result is presented as the straight line, the power-law coefficient is found as $\gamma = 1.17$.

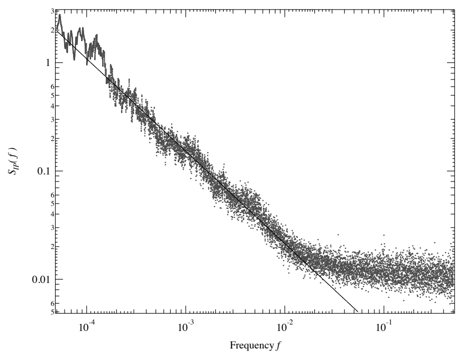


Figure 2.8: Power specrum $S_{IP}(f)$ of the inter-event time-signal $I_{IP}(t)$ printed on log-log scale. The observations are fitted with (2.3), where the exponent of the fit is found as $\varphi = 0.85$.

2.6 Internet measurement projects

There has been considerable work devoted to Internet path measurement in several projects, such as Skitter[6], PlanetLab[7] and Rocketfuel[8]. To our knowledge, the database from RIPE is the only database that actually has IP-level measurements for a prolonged period (more than 9 years) at a relatively high sample rate between relatively fixed and stable testboxes. The Skitter project, which also measures IP path information, was initiated around the same time as the RIPE TTM project, however its goals are different as well as the measurement architecture. The PlanetLab architecture was established to provide a measurement infrastructure to a large community, where each party can design and conduct its own measurements. Measurements on PlanetLab are criticized since the PlanetLab nodes are typically placed at well connected sites, such as universities, providing a skewed view of the Internet's connectivity and performance [12]. The Rocketfuel project combines BGP data with traceroute measurements to infer the ISP topologies [62]. RIPE controls roughly 150 testboxes that send measurement packets between each other, while the Skitter project controls a set of approximately 20 source nodes that query a huge list of approximately 400,000 destinations. The RIPE testboxes are designed for measurement purposes and its location is very stable, making them very suitable for long-term analysis. The selection criterion for destinations in the Skitter project is to achieve a representative coverage of the routed IPv4 address space. The destinations are not controlled by the Skitter project and the list is updated every few months. The goal of the RIPE TTM project is to study path properties (traceroute, end-to-end one-way delay and packet-loss), that resulted in frequent sampling of the end-to-end path between testboxes. The Skitter project's main focus is Internet tomography and the sample rate is, with ap-

[6]http://www.caida.org/tools/measurement/skitter/
[7]http://www.planet-lab.org
[8]http://www.cs.washington.edu/research/networking/rocketfuel/

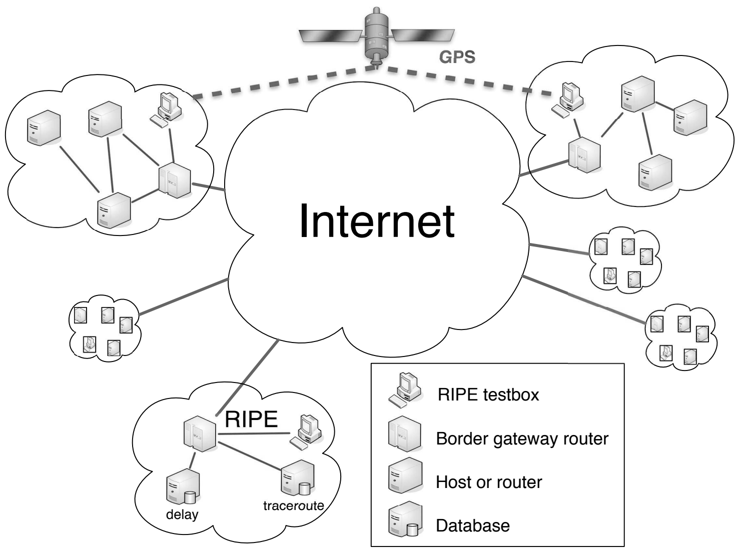


Figure 2.9: Schematic overview of the RIPE TTM measurement architecture. RIPE testboxes are placed at customer's sites, typically just behind their firewall. The clock of the testboxes is synchronized by means of GPS signals. Each testbox send test packets to a selection of the other testboxes and performs traceroute measurements as a complementary service to measure the path towards the destinations. The data is collected and distributed at RIPE.

proximately several queries per destination per day, considerably lower. Thus, we believe that the RIPE database is the most suitable at hand for long-term analysis of Internet paths between fixed end-points. Figure 2.9 presents a schematic overview of the RIPE TTM measurement architecture.

Several works have examined routing dynamics in the Internet, but none of them have examined the long-term correlations between routing events. Early work by Paxson [52] studies the stability of Internet paths through traceroute measurements performed during several months. Labovitz *et al.* [37] studies the path stability by combining IP and AS information. Iannaccone *et al.* [25] and Markopoulou *et al.* [40] study the path properties by monitoring the route update-messages within an AS and conclude that a small fraction of the links contributes to a large fraction of the route updates. Pucha *et al.* [55] and Wang *et al.* [70] study the impact of route changes on the path performance w.r.t. packet delay and jitter. Our work differs from the previous works in that the dataset that we use extends over a period of five years, which exposes long-term effects. Furthermore, we associate the routing dynamics with self-organized criticality and argue that routing is unpredictable leading to large variations in the path performance.

2.7 Chapter summary

The Internet consists of a constantly evolving complex hierarchical architecture where routers are grouped into autonomous systems (ASes) that interconnect to provide global connectivity. Routing is generally performed in a decentralized fashion, where each router determines the route to the destination based on the information gathered from neighboring routers. Consequently, the impact of a route update broadcasted by one router may affect many other routers, causing an avalanche of update messages broadcasted throughout the network. The RIPE TTM project provides measurements between a set of testboxes located primarily at ISP and well-connected sites. Through analysis of traceroute measurements we have studied the lifetime of routes and the route dynamics. Based on the observation from our dataset we find that routing in the Internet is highly dynamic and results in unpredictable route durations. The extreme power-law behavior suggests that the Internet exhibits self-organized criticality. The power spectrum obtained from the inter-event time of route changes confirms our conjectures. The variability in the Internet paths demonstrates the resilience against failures and attacks. For time-critical services, the unpredictability introduced by SOC may lead to quality degradation.

Chapter 3

A queueing system to model network dynamics

3.1 Introduction

When modeling the performance of a complex system, it is generally desired that the multitude of variables that describe the system can be expressed as independent factors. In most cases these performance measures are emergent properties, that arise from the complex interaction between the many individual parts of the system. With respect to networks, the individual parts are represented by the nodes, which interact through the complex infrastructure represented by the topology. In this chapter we present a network model described by a very small set of parameters and measure the performance under the influence of these parameters. Using a small parameter set reduces the model complexity and facilitates to isolate the effect of the each parameter on the network performance. When the network performance can be expressed in terms of simple network parameters, then this would overcome the need of doing difficult computations or lengthy simulations. Furthermore, it is possible to benchmark other networks based on the performance measures. The term "network dynamics" is here broadly understood as the set of network properties such as the blocking and loss rate, the number of allocated paths, the hopcount of these paths, etc. that change over time when loading the network with traffic. In Section 3.2 we detail the network model that is used throughout this chapter. Section 3.3 presents a queueing analogue for the network model that facilitates a convenient way to measure the network performance. The remainder of the chapter studies the various network classes and traffic patterns and discusses the results.

3.2 Network model

The loading of a network with traffic needs to be detailed. In general, traffic is injected into the network and leaves the network elsewhere. First, we assume that traffic is only injected in one node, the source, and that it leaves the network at one other node, the destination. In other words, we confine to unicast. The network is considered a fixed topology, without emerging, vanishing or mobile nodes or links, that consists of N nodes and L links. The source-destination pair is assumed to be uniformly chosen over all N nodes of the network. This assumption for the Internet is quite realistic as argued in [43, pp. 340]. A slightly more realistic setting is to take the density population of users on earth and measurements of traffic matrices into account, at the expense of a considerably more complex model.

Second, since modeling the network at packet-level will render the analysis highly complicated, we confine to flows. A flow can be regarded as a connection set-up between a source and a destination node in the network. The ensemble of all packets of that flow follow the same path from a source to destination during the life-time of that flow. A flow can be a regarded as a connection with minimal-capacity requirements, e.g. a Label Switched Path (LSP) with Quality-of-Service constraints associated with multimedia and interactive streaming applications and services. In addition, a flow can be an accumulation of connections from source to destination, e.g. packets originated at various sources entering and leaving the backbone at the same ingress and egress routers.

Since only flows are observed in our setting, the details of the packet level such as packet inter-arrival times and packet correlations can be omitted, but only the flow arrival rate plays a role. We assume a Poisson arrival process with average rate λ. The Poisson assumption for flows is commonly regarded as realistic: it is very precise for telephony, and, on the aggregate level, also for the Internet. The flow duration is determined by the servicing process, which is detailed in Section 3.3. The flows are serviced at a mean rate μ, such that the load in the network becomes $\rho = \lambda/\mu$. Figure 3.1 depicts the network model with the arrival process. Flow arrivals occur at an average rate λ and are routed and allocated in the network. The flows expire at an average rate μ.

Next, we need to specify the capacity consumption of a flow. The capacity consumption of a flow is considered fixed at one unit for all flows and is independent of the already present flows on a link. The link capacity C is divided into a discrete number of *channels*, where each channel can accommodate precisely one flow-unit. All links have the same capacity, which must be at least one channel. Furthermore, the links are considered unweighed and bidirectional. We exclude the existence of self-loops and multiple links between nodes. Thus, the network is homogeneous. Each flow is initiated with a flow set-up request. The flow request consists of the source-destination pair and the required capacity. Upon the arrival of a flow set-up request, the routing algorithm inspects the network and disregards the links with insufficient capacity during the path computation. Then a minimum-hop path is computed between the source-destination pair using Dijkstra's shortest path algorithm. Hence, we do not restrict to static routes between source-destination pairs, but the flow is dynamically routed based on the availability of the network resources. If a feasible path is found, the request is honored and the connec-

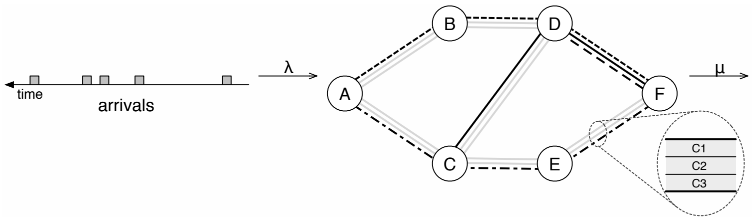


Figure 3.1: Example of a network model with 6 nodes and 7 links. In this example, each link consists of 3 channels, as shown by the detail. On the left-hand-side of the network model, the arrival process is depicted, where each square indicates the event of a flow arrival. Arrivals occur at random intervals at a rate λ and the requests are routed in the network. Each flow consumes one channel of the links along its path. In this example, the unused channels are identified by the solid, gray coloring. Channels that are marked with the same line-style correspond to the same flow, e.g. flow $A-B-D-F$. Flows between the same in- and egress nodes may follow different routes, e.g. compare flows $A-B-D-F$ and $A-C-E-F$.

tion is established by allocating one unit capacity on the links along this path. If multiple shortest paths are found, one of these paths is chosen randomly. In case of insufficient resources, the flow is rejected by the network and the request is rejected. When the full capacity of a link is occupied by the flows, the allocated link remains unavailable for future requests until at least one unit capacity is released.

Although the network model may seem overly simplified with respect to reality – a fact that we do not deny –, this comment should be placed in some perspective. The major reason for choosing such an extreme scenario was inspired by the question: “How many flows (light paths) can be set-up in a network?” The maximal possible number of flows that can be set-up in our setting appears in the complete graph because any other graph is a subgraph of the complete graph. Hence, we first focus on the complete graph. The simpler model of a lattice in which each flow just consists of one link already leads to a difficult percolation problem [22]. Indeed, when removing random links in a lattice, the lattice is disconnected with high probability if the link density $p = \frac{L}{L_0} \rightarrow \frac{1}{2}$ for large N, where L is the number of links in the percolated lattice and L_0 is the number of links in the full lattice. When considering the complete graph instead of the lattice, the complete graph is disconnected by removing random links when $p < p_c$ and the threshold link density is $p_c \approx \frac{\ln N}{N}$ for large N. This is a key result in the theory of the Erdös-Rényi or classical random graph $G_p(N)$, see e.g. [30]. In the first stages of loading the network – equivalent to removing paths, a set of correlated links –, our network model shows resemblance with the random graph $G_p(N)$. However, the fact that paths and not random links are removed, is shown to considerably complicate the understanding of our results.

3.3 A queueing model of our network model

We regard the network as a single system at which flows arrive and depart at random times. We assume that the network functionalities such as routing, signalling and admission-control, are not visible outside the network. Hence, the network can be viewed as a black box of which we evaluate its performance by means of external measures, such as rejection rate and average throughput. Figure 3.2 illustrates our model, the network is illustrated as a cloud at the top of Figure 3.2. We compare the network model with a single-server queue with finite buffer size K at which jobs (flow requests) arrive and depart, as visualized at the bottom in Figure 3.2. The correspondence between the network model and its queueing analogue requires that the network and the queue experience a same in- and output. The arrival of a flow at the network coincides with adding a new job to the queue's buffer. The analogue of the life time of a flow is the sojourn time of a job at the queue, where each job consists of (*i*) selecting an arbitrary source-destination pair, (*ii*) performing a shortest path computation, and (*iii*) setting-up of the path/connection or, if not possible, announcing an error/rejection message. The completion of a job at the server coincides with the release of a flow from the network. Hence, the average service rate at the queue is equal to the average release or termination rate of a flow in the network.

An important difference between the network model and the regular queue is the service order. The regular queue services the jobs in the order of arrival, hence in First-In-First-Out (FIFO) order. The flows in the network are terminated arbitrarily: at a fixed average rate μ one of the flows present in the network is chosen uniformly and terminated, irrespective of the time the flow already resides in the network. Consequently, the service discipline in the queueing system should be random, in the sense that the server selects a uniformly chosen job in the buffer. Our network modeling assumptions have been taken in such a way that the corresponding queue arrival process is a Poisson process with rate λ and the service process is exponentially distributed with mean service rate μ. Hence, the analogue of the network model is of the M/M/1 queuing family. The sequel of this section is devoted to further discuss and motivate the analogy.

The M/M/1 queue is one of the few queueing systems for which a time-dependent analysis is available. The mean system size $N_S(t)$, i.e. the number of jobs present in the buffer plus server, of the M/M/1 queue as a function of time is given for $\rho < 1$ by [63],

$$\mathrm{E}\left[N_{S_{M/M/1}}\right] = \frac{\rho}{1-\rho} \tag{3.1}$$

$$\mathrm{E}\left[N_{S_{M/M/1}}(t)|N_{S_{M/M/1}}(0)=0\right] = \frac{\rho}{1-\rho} - \frac{2}{\pi}\int_0^{\pi} \frac{e^{-\gamma(y)\mu t}\sin^2 y}{\gamma(y)^2}dy \tag{3.2}$$

where (3.1) corresponds to the average system size in steady state and where $\gamma(y) = 1+\rho-2\sqrt{\rho}\cos y$. Abate and Whitt [1] and Sharma *et al.* [58,59] provide alternate expressions for the mean system size. The simple M/M/1 queue has infinite buffer capacity and the average number of jobs is entirely determined by the load. Hence, all jobs are accepted by the system. In the network model analogue, the system size corresponds to

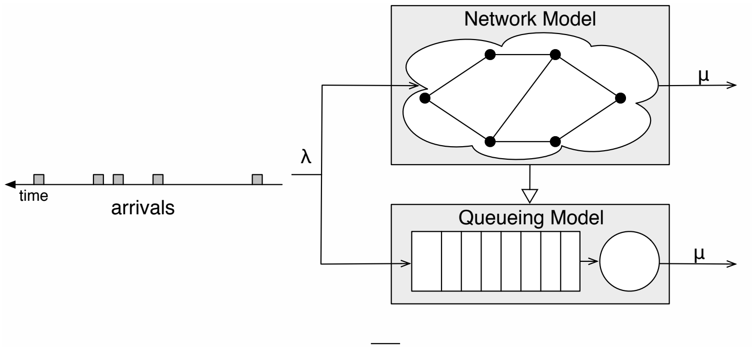


Figure 3.2: The network is represented by a graph and modeled by a single queue. The cloud illustrates that the network is regarded as a black box. The flow arrivals at the network correspond to arriving jobs at the queue. The jobs in the queue are scheduled and processed in FIFO order, while the flows in the network are processed in arbitrary order.

the network throughput: the number of flows. The network capacity is finite and determined by the network parameters, such as the number of links and the topology. When the network lacks resources to accommodate a new flow request, the request is rejected. The rejection rate is equal to the loss probability for each new flow request and is defined as,

$$r = \frac{\mathrm{E}\left[R_{\text{reject}}\right]}{R_{\text{total}}} \tag{3.3}$$

where R_{reject} and R_{total} represent the sum of rejected requests and the total number of requests, respectively. The rejection rate is a good measure to evaluate the network performance, because it indicates the probability of a successful transmission.

In our queueing model, we can translate the finite network resources into a finite-sized buffer. The size of the buffer then relates to the maximum number of concurrent flows in the network. If we assume an M/M/1/K model as analogy for the network, then K corresponds to the maximum number of flows that the network can accommodate. In the network perspective, we will refer to K as the “”network capacity”. The network capacity is an emergent property of the network and can not immediately be determined based on the network parameters. Analytic solutions for the M/M/1/K queue in both the transient and stationary domain are found by Tarabia [65]. Equations (3.4)–(3.6) give the first and second order moments of the system size in steady state and the first order moment in the transient domain,

$$\mathrm{E}\left[N_{S_{M/M/1/K}}\right] = \begin{cases} \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}, & \text{if } \rho \neq 1 \\ \frac{K}{2}, & \text{if } \rho = 1 \end{cases} \tag{3.4}$$

$$\mathrm{E}\left[N^2_{S_{M/M/1/K}}\right] = \begin{cases} \frac{\rho C(\rho)}{(1-\rho)^2\left(1-\rho^{K+1}\right)}, & \text{if } \rho \neq 1 \\ \frac{1}{6}K + \frac{1}{3}K^2, & \text{if } \rho = 1 \end{cases} \tag{3.5}$$

$$\mathrm{E}\left[N_{S_{M/M/1/K}}(t)\right] = \mathrm{E}\left[N_{S_{M/M/1/K}}\right] - \frac{1}{K+1}\sum_{j=1}^{K} A(\rho,\nu)e^{-(\lambda+\mu)t+2t\sqrt{\rho}\cos\nu}, \quad \text{if } \rho \neq 1 \tag{3.6}$$

with $\nu = \frac{\pi j}{K+1}$ and where

$$A(\rho,\nu) = \frac{2\rho\left(\sin\nu + \rho^{\frac{K+1}{2}}\sin K\nu\right)\sin\nu}{\left(1+\rho-2\sqrt{\rho}\cos\nu\right)^2} + \rho^{\frac{K+2}{2}}\left(2+4K+\rho\left(2K-1\right)\right)\sin K\nu$$

$$C(\rho) = 1+\rho-(K+1)^2\rho^K + \left(2K^2+2K-1\right)\rho^{K+1} - K^2\rho^{K+2}$$

The steady state rejection rate for the M/M/1/K queue is found as the probability that the system contains K items,

$$r_{M/M/1/K} = \Pr\left[N_{S_{M/M/1/K}} = K\right] = \begin{cases} \frac{(1-\rho)\rho^K}{1-\rho^{K+1}}, & \text{if } \rho \neq 1 \\ \frac{1}{K+1}, & \text{if } \rho = 1 \end{cases} \tag{3.7}$$

In this work we do not consider scenarios where the load exceeds 1. From equation (3.7) it can be shown that for $\rho > 1$. the rejection rate $r_{M/M/1/K} \simeq 1 - 1/\rho$. The system will always stay in a "stressed" state with excessive losses. This behavior is neither desirable nor typical for networks and therefore we disregard these scenarios from our analysis.

The average system time, the time each jobs spends in the system, can be obtained by applying Little's Law [43, Sect. 13.6]. Little's Law requires that the arrivals are not affected by the service discipline. An important difference between the classical M/M/1 queue and the network model is the service order: the classical M/M/1 queue governs a FIFO order, whereas the network releases flows arbitrarily. Since we are interested in average quantities to first order, Little's law is not affected by the service discipline. Little's Law also assumes a work-conserving system where all offered load is serviced. Due to the finite capacity of the system, the arrival rate is affected by the rejections in the network. Consequently, Little's Law must be applied with use of the effective arrival rate. The average system time can be written as,

$$\mathrm{E}\left[T_{S_{M/M/1/K}}\right] = \frac{\mathrm{E}\left[N_{S_{M/M/1/K}}\right]}{\lambda(1-r)} = \frac{1-\rho^{K+1}}{\lambda(1-\rho^K)}\mathrm{E}\left[N_{S_{M/M/1/K}}\right]$$

```
MAIN-PROGRAM
 1  INITIALIZE($K_N$)
 2  queue $Q_{flows} \leftarrow \varnothing$
 3  $t_{arrivals} \leftarrow t_{departures} \leftarrow 0$
 4  $t_{int.departure} \leftarrow$ RANDOM-EXPONENTIAL($\frac{1}{\mu}$)
 5  for $j \leftarrow 1$ to $j_{max}$
 6       do $t_{arrivals} \leftarrow t_{arrivals} +$ RANDOM-EXPONENTIAL($\frac{1}{\lambda}$)
 7          while $Q_{flows} \neq \varnothing$ and $t_{departures} + t_{int.departure} \leq t_{arrivals}$
 8              do $t_{departures} \leftarrow t_{departures} + t_{int.departure}$
 9                 RELEASE-FLOW($K_N, Q_{flows}$)
10                 $t_{int.departure} \leftarrow$ RANDOM-EXPONENTIAL($\frac{1}{\mu}$)
11          if $Q_{flows} = \varnothing$
12              then $t_{departures} \leftarrow t_{arrivals}$
13          flow $f \leftarrow$ GENERATE-FLOW($K_N$)
14          if $length[f] > 0$
15              then ALLOCATE-FLOW($K_N, Q_{flows}, f$)
```

Figure 3.3: Meta-code for the MAIN-PROGRAM. The MAIN-PROGRAM forms the basic flow of the network simulation.

A non-trivial issue is the determination of K. In the M/M/1/K queueing model, K is simply a parameter that is fixed. In the network perspective, K is a random variable that, in a complicated manner, emerges from the topology and traffic parameters. The objective is to identify and explore the relation between these parameters and the network capacity K. We use simulation to measure the network performance in several scenarios with different parameter settings and then compare the performance of the network with that of the M/M/1/K queueing system.

The meta-code for the simulation of the network is presented in Figures 3.3 to 3.6. The simulation uses a queue to store the flows that are active in the network. The flows are stored in the order of arrival, however, the flows are dequeued at random with equal probability. The simulation maintains two timelines, corresponding to the two processes in the system, namely the arrival process and the departure process. When a flow arrives, the arrival timeline is advanced with the inter-arrival time of the forthcoming flow. Similarly, when a flow departs, the departure timeline is advanced. When the queue is empty, the timelines are synchronized. The simulation is driven by the events of the flow-arrivals. Just before a new flow arrives at the network, the active flows are examined and the flows with a departure time prior to the forthcoming arrival are terminated. Hence, the network performance, such as the number of flows momentarily in the network, is only evaluated at the time of a flow arrival. This simplification is justified because the primary performance measures of interest, being the number of flows, hopcount and rejection rate, are measured at the event of an arrival.

Figure 3.3 presents the meta-code for the main program of the simulation. The network is initialized in line 1 as the complete graph K_N with unweighed, bi-directional links

```
GENERATE-FLOW($K_N$)
1  flow $f \leftarrow \varnothing$
2  node $u \leftarrow v \leftarrow$ RANDOM-NODE($K_N$)
3  while $u = v$
4      do $v \leftarrow$ RANDOM-NODE($K_N$)
5  $f \leftarrow$ DIJKSTRA($K_N, u, v$)
6  return $f$
```

Figure 3.4: Meta-code for GENERATE-FLOW subroutine. The GENERATE-FLOW subroutine choses two random nodes from the network and computes the shortest path between these nodes using DIJKSTRA, Dijkstra's shortest-path algorithm.

with equal capacity. The network is empty and unloaded. In line 2 the queue Q_{flows}, which stores the active flows in the networks, is initialized. Line 3 initializes the variables $t_{arrivals}$ and $t_{departures}$ that correspond to the flow-arrival and -departure timelines, respectively. In line 4 the inter-departure time is drawn from an exponential distribution with mean $1/\mu$. This is the inter-departure time for the next flow that will leave the network. The simulation starts in line 5, where it enters a loop that is executed for each flow-arrival. The number of repetitions, which equals the number of flow arrivals j_{max}, determines the length of the entire simulation. The time of the next flow arrival is determined in line 6 and the corresponding variable $t_{arrivals}$ is advanced. Next, the active flows that have expired before $t_{arrivals}$ need to be terminated. This is done in lines 7-10. In line 7, a conditional loop is started that is repeated for each flow departure. During the loop, the departure timeline is advanced (line 8), the flow is terminated (line 9) and the inter-departure time is determined for the next flow (line 10). The loop ends when the queue Q_{flows} is empty, or when the time of the next flow departure $t_{departures}$ exceeds $t_{arrivals}$. If Q_{flows} is empty after the flow departures, the two timelines need to be synchronized, this is done in lines 11 and 12.[1] In line 13 the actual flow is generated and stored in the variable f. Finally, in lines 14 and 15, the length of the flow is tested and if it is greater than zero, meaning that a path has been found, the flow is allocated in the network.

The meta-code for the generation of a flow is detailed in Figure 3.4. The subroutine takes a single argument, the network K_N, which is needed to select the source-destination pair and compute the path. First, an empty flow f is initialized in line 1. Next, two (different) nodes are chosen randomly from K_N in lines 2-4. In line 5 the shortest path between u and v is computed using Dijkstra's shortest path algorithm and inserted into f. If two paths are found with equal length, one path is chosen uniformly. If the path does not exist, an empty path is inserted into f. Line 6 returns the resulting flow.

In Figure 3.5 the meta-code is presented for the allocation of a flow. The subroutine takes three arguments, the network K_N, the queue Q_{flows} and the flow f that must be allocated. The subroutine traverses the path stored in f and allocates the links between

[1]If the two timelines were not synchronized, the departure time of the next flow would be based on the previous departure time, which could be prior to its own arrival.

```
ALLOCATE-FLOW($K_N, Q_{flows}, f$)
1  for $i \leftarrow 1$ to $length[f] - 1$
2      do node $u \leftarrow f[i]$
3          node $v \leftarrow f[i+1]$
4          ALLOCATE-LINK($K_N, u, v$)
5  ENQUEUE($Q_{flows}, f$)
```

Figure 3.5: Meta-code for ALLOCATE-FLOW subroutine. The ALLOCATE-FLOW subroutine allocates the links along the path of a flow.

```
RELEASE-FLOW($K_N, Q_{flows}$)
1  flow $f \leftarrow$ DEQUEUE-RANDOM($Q_{flows}$)
2  for $i \leftarrow 1$ to $length[f] - 1$
3      do node $u \leftarrow f[i]$
4          node $v \leftarrow f[i+1]$
5          RELEASE-LINK($K_N, u, v$)
```

Figure 3.6: Meta-code for RELEASE-FLOW subroutine. The RELEASE-FLOW subroutine releases the links along the path of a flow, such that they become available again for future flow requests.

along the path (lines 1-4). The ALLOCATE-LINK procedure in line 4 allocates one of the available channels on the link between the nodes u and v. Note that if the link would not have a free channel, the link would not be part of the path. In line 5, the flow is added to the queue Q_{flows}. Figure 3.6 details the release of a flow. The RELEASE-FLOW subroutine only takes two arguments, the network K_N and the queue Q_{flows}. The flow that will be released is selected uniformly from the set Q_{flows}. The flow is stored in f and then removed from Q_{flows} (line 1). Next, the links are released along the path in f (lines 2-5). The release of a link involves freeing an occupied channel of the link and making it available again for future flows.

The performance measures obtained through the simulations are applied to the queueing model and the network capacity is obtained through fitting. A novel point in our analysis is the relation (3.8) between K and the network capacity, at least for the complete graph K_N, where $\beta \approx 0.42$,

$$K = \beta L_0 \tag{3.8}$$

where β is the fit-parameter and L_0 is the number of links in the complete graph,

$$L_0 = \binom{N}{2} \tag{3.9}$$

Writing the network capacity as a function of the number of links, is motivated by the notion that in our model the links carry the actual traffic. The nodes can be regarded as an *adhesive* between the links, but do not contribute to the real capacity themselves.

Reviewing relation (3.8) in the light of the queueing model, we can regard β as an upper-bound on the fraction of links that is used simultaneously. After all, in our queueing model, K forms a hard upper-bound on the maximum number of jobs in the system. Hence, a high value for β implies that a large fraction of the links is used simultaneously. In other words, β can be viewed as the *efficiency* of the network usage. We assess the quality of the proposed relation (3.8) and study β for a range of networks with various parameters. By tuning a single parameter in each scenario, we can concentrate on the effect of this parameter on the network performance.

First, we study the case of the complete graph K_N and explore the dependency between the network capacity and the network size. Next, we investigate the effect of decreasing the link density p by removing random links from the complete graph. For this class of graphs, the Erdös-Renyí random graphs, we are interested in the relation between the link density and the network capacity. Third, the complete graph is reconsidered, but now a link may hold multiple flows at the same time. Fourth, we will focus on the two-dimensional lattice with one channel per link. Finally, we will test relation (3.8) in the case of a different service model. In each simulation, we have generated 10 network realizations and issued $1,100,000$ flow requests per realization. At the start of each simulation, the network is empty. In the assessment of the steady-state behavior, the initial $100,000$ flows of each realization are ignored in our analysis to account for the warm-up phase. The arrival rate is chosen as $\lambda = 0.99$ while the service rate is $\mu = 1.00$. Hence, the load equals the arrival rate $\rho = \lambda = 0.99$.

3.4 The fully connected graph

In our initial set-up we will examine the complete graph with a single channel per link. The complete graph has only a single parameter, namely the number of nodes, and is therefore the most suitable model to start our analysis with. We use simulations to study the network performance. To begin with, we are interested how the average number of flows in steady state is a function of the number of nodes N. Furthermore, we will examine the average hopcount and the rejection rate related to N.

At the start of the simulation, the network resides in a temporal warm-up phase, where the average amount of incoming traffic exceeds the average amount of traffic being served. The load gradually increases until a steady state is reached where the average number of flows entering and leaving the network are in balance. The pace at which the system is loading, can be measured by means of the relaxation time. The relaxation time is a common performance measure in queueing theory and it reflects the convergence rate to the stationary regime. For the M/M/1 queue, the relaxation time is given by [43, pp. 217],

$$\tau_r = \frac{1}{\mu\left(1-\sqrt{\rho}\right)^2} \tag{3.10}$$

The speed of convergence to the steady state is compared for different network sizes. We have simulated 10 network realizations and issued $1,100,000$ flow requests per realiza-

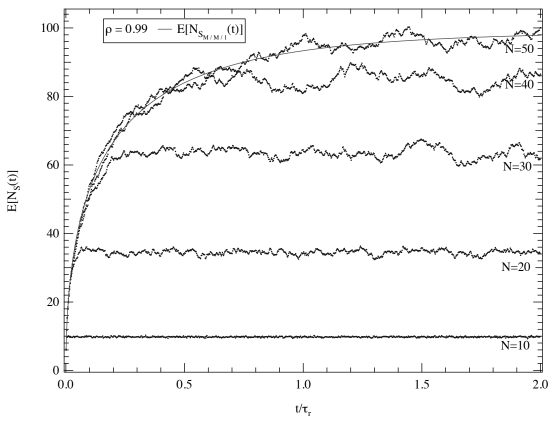


Figure 3.7: Simulations of the average number of flows for networks of various sizes and the average number of jobs in the M/M/1 queue as a function of time. The time-axis is normalized with respect to the relaxation time ($\tau_r \approx 39800$). Computations of (3.2) are added for reference. The simulation results presented in this figure are obtained by averaging over 1000 network realizations and 100,000 flows per realization.

tion. In the analysis of the steady-state behavior, the first 100,000 flows are ignored. The arrival and service rate are chosen as $\lambda = 0.99$ and $\mu = 1$, respectively.

3.4.1 Number of flows in the network

The throughput of the network can be measured in terms of the number of concurrent flows in the network. Figure 3.7 presents the average number of flows as a function of time for networks of various sizes. Computations of mean system size of the M/M/1 queue (3.2) have been added to Figure 3.7 for reference. With $\rho = \lambda/\mu = 0.99$, the mean steady state system size of the M/M/1 queue is $E\left[N_{S_{M/M/1}}\right] = 99$. Since the M/M/1 queue does not induce rejections, $E\left[N_{S_{M/M/1}}\right]$ can be seen as an upper-bound for the average number of flows in the network; which is also evident from (3.4). The average number of flows in the stationary regime, the so-called steady state, depends on the network size. Small networks lack resources to accommodate each request, resulting in early rejections. As the network size increases, the average number of flows tends towards the computations of (3.2), implying that only few of the flow request are being rejected. For $N = 50$ the average number of flows in Figure 3.7 closely follows equation (3.2). Networks of more than 50 nodes have therefore not been considered with this load. Choosing ρ closer to 1 will increase the variance of the average throughput and will lead to large fluctuations in the flow arrivals. Moreover, an extremely high load may result in numerical instability. The fluctuations around the average number of flows in the steady state increase as the network size grows, which agrees with (3.4) and (3.5). To

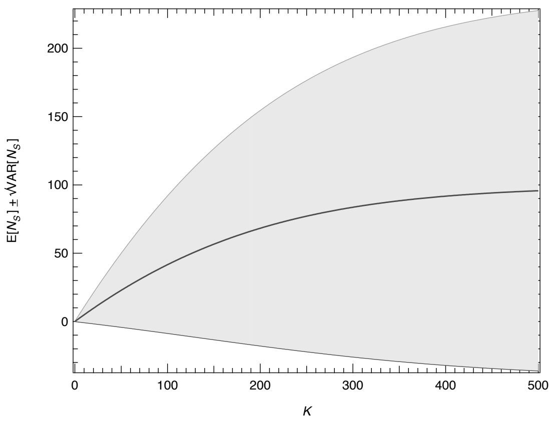


Figure 3.8: Computations of equation (3.4), the mean system size of the M/M/1/K queue as a function of K and $\rho = 0.99$. The mean has been plotted with the square root of the variance, cf. equation 3.5. At this high load, the standard deviation exceeds the mean.

get an impression of the magnitude of the variance as compared to the mean number of flows in the M/M/1/K queue, the average system size as a function of K has been plotted in Figure 3.8 with error bars ($\pm$ one standard deviation). Due to the high load, the standard deviation nearly equals the average system size and the coefficient of variation is very close to one, $C_v = \sqrt{\mathrm{Var}[N_S]}/\mathrm{E}[N_S] \approx 1$ (see [43, pp. 273]).

To find a relation between the network size N and buffer size K, we have computed the sample mean of the number of flows in the steady state from Figure 3.7 over an interval of $1{,}000{,}000$ arrivals. The mean has been plotted against the number of nodes in Figure 3.9. The sample mean $\mathrm{E}[N_S]$ as a function of the number of nodes has been fitted with (3.4) where $K = \beta\binom{N}{2}$. Figure 3.9 demonstrates that (3.4) agrees remarkably well with the simulations. The spread of the samples that occurs for larger N is due to the high variance, as mentioned previously. The result $K = \beta\binom{N}{2}$ is remarkable, as it implies that there is a linearity between the network capacity and the number of links in K_N. The best value for $\beta = 0.42$ (see the legend of Figure 3.9). The number of links that effectively contributes to the capacity appears constant at some 42 percent of the total number of links, independent of the network size. Figure 3.10 compares the simulations with computations of (3.6) with $K = \beta\binom{N}{2}$ for various N. The correspondence of the simulations with (3.6) is very good; both the steady state behavior as well as the transient warmup phase are closely approximated by (3.6).

3.4.2 Rejection rate

When the network lacks resources (capacity) to accommodate a new flow, the flow request is rejected. With respect to the M/M/1/K model, this is analogous to the situation where the buffer is full. An important difference with the M/M/1/K is that the flow re-

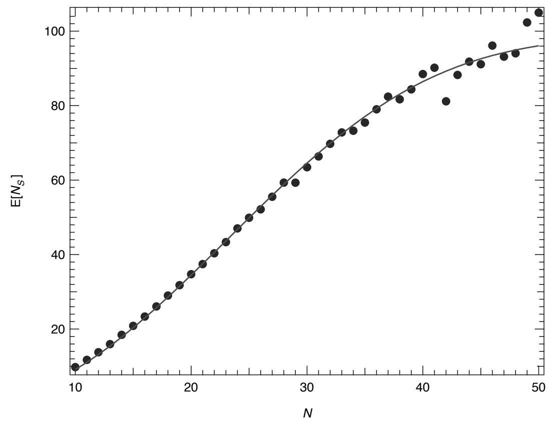


Figure 3.9: The average number of flows in steady state as a function of the number of nodes N (see also Figure 3.7). The simulations (dots) have been fitted with computations of (3.4) (line).

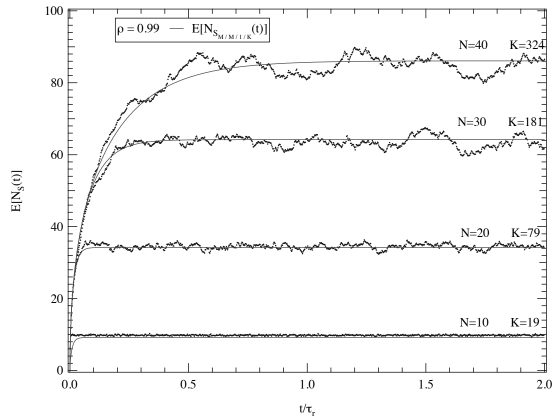


Figure 3.10: The average number of flows in the network as a function of time, compared with computations of (3.6). The simulation results presented here are the same as in Figure 3.7.

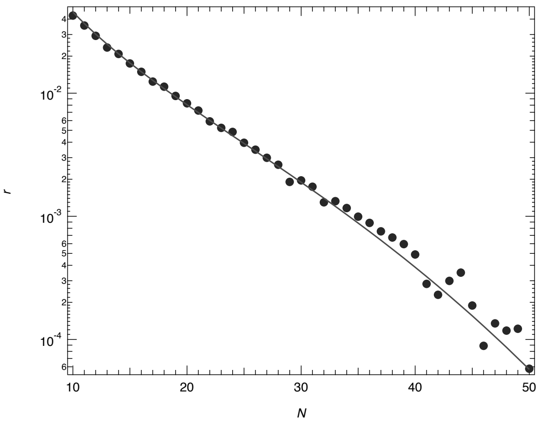


Figure 3.11: Simulation results of the average rejection rate in steady state as a function of the network size (dots). Computations of (3.7) have been added, where $K = \beta\binom{N}{2}$ and $\beta = 0.42$ (line).

quest may be rejected in cases where there is still sufficient capacity available in the network, but the routing fails because it is not possible to construct a path from source to destination with the available links. Figure 3.27 presents the simulation results for average rejection rate r in the steady state. Figure 3.11 also shows computations of (3.7) with $\rho = 0.99$, $K = \beta\binom{N}{2}$ and $\beta = 0.42$. Figure 3.11 demonstrates that rejections in the network are well approximated by the M/M/1/K queueing system, even in cases where the rejection rate is very small. The fluctuations in the tail of Figure 3.11 are an artifact of the limited size of the sample set. Running longer simulations will increase the number of samples and reduce the statistical inaccuracy. The good match again underlines the importance of the relationship $K = \beta\binom{N}{2}$ between the network capacity and the number of links in the network, as explained in the previous section.

3.4.3 Hopcount

The hopcount is the number of links (hops) a flow must traverse to reach a destination from a given source. It is a random variable that depends on many variables in the network, e.g. the number of nodes, the link density and the topological structure. The average hopcount indicates how many links are allocated per flow on average. Figure 3.12 presents the average hopcount of the shortest path for flows entering the network as a function of time. The time-axis has been normalized with respect to the relaxation time. At the start of the simulation, the average hopcount is precisely one, namely corresponding to the direct link between the source-destination pair. The average hopcount increases as more flows are allocated due to the depletion of resources. After some time, the average hopcount reaches a stationary regime, corresponding to the steady state. In the steady state, the average hopcount of the arriving flows is the same as for the departing

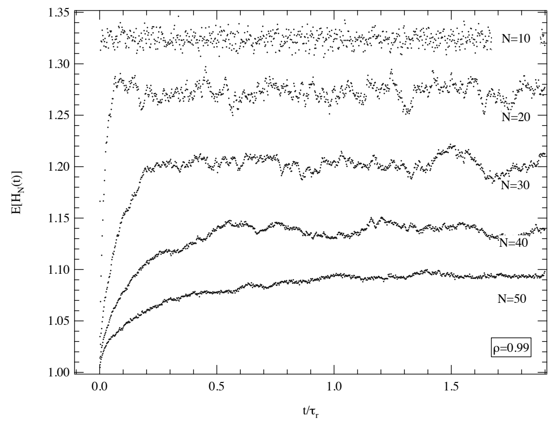


Figure 3.12: Average hopcount of the allocated flows as a function of time. The time axis has been normalized with respect to the relaxation time.

flows. Figure 3.12 demonstrates that the steady state value at which the average hopcount stabilizes becomes smaller when the network size increases. In addition, the variance of the hopcount also decreases when N increases. The explanation is that, when N grows, the number of potential routes between a particular source and destination pair increases. In particular, the number of short paths, e.g. the 2-hop paths, increases rapidly with N, resulting in a higher probability of finding a shorter route (lower mean) and more routes of the same size (lower variance). In addition, the number of source-destination pairs increases quadratically with N, hence the probability of choosing a source-destination pair of which the direct link is already consumed, is smaller.

In our model, the allocation of a link consumes the full capacity of that link which is analogous to the (temporal) removal of that link. The links remain unavailable for future requests until they are released again. The number of allocated links at some time t equals the sum of the length (in hops) of all the flows in the network at that time,

$$L_{allocated}(t) = \sum_{j=1}^{N_S(t)} H_j$$

where H_j equals the number of hops of flow j. After taking the expectation of both sides, invoking Wald's identity [43, pp. 34] and assuming that each H_j is i.i.d. as H_N and that $N_S(t)$ is independent from H_j, we obtain

$$\mathrm{E}[L_{allocated}(t)] = \mathrm{E}[N_S(t)]\,\mathrm{E}[H_N]$$

The number of allocated links relates to the number of available links as $L(t) = L_0 - L_{allocated}(t)$, where L_0 is the initial number of links in the network. The link density is defined as the ratio of available links with respect to the maximum number of links,

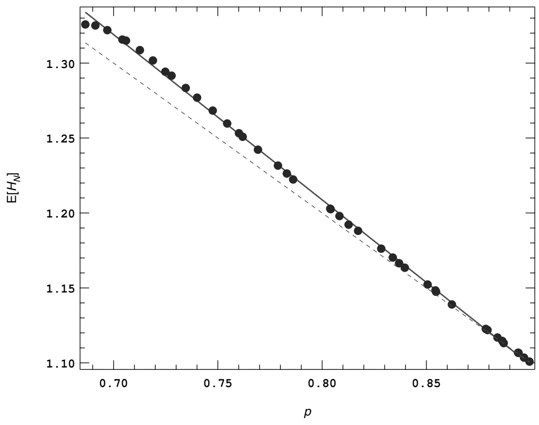


Figure 3.13: Simulation results of the average hopcount as a function of the link density p during steady state (dots). The simulation results have been fitted with a line (solid line). The dotted line corresponds to $\mathrm{E}[H_N] = 2 - p$.

which for the steady state converts to $p = \mathrm{E}[L]/L_0 = 1 - \mathrm{E}[L_{allocated}]/L_0$. Hence, in the steady state, the link density can be written as a function of the average number of flows and the average hopcount as,

$$p = 1 - \frac{\mathrm{E}[N_S]\,\mathrm{E}[H_N]}{L_0} \tag{3.11}$$

We have computed the average hopcount in the steady state and plotted the result in Figure 3.13 as a function of the link density. The average hopcount in the steady state appears to drop linearly with the link density. The linear behavior can be explained when we consider the allocation of the flows. Provided that the average path length (in hops) mostly spans a single link and seldom more than two, which appears to be the case according to Figure 3.13, the process of allocating flows is quite well modeled by the construction of the Erdös-Renyí random graph, where links are removed from K_N independently and at random with probability $(1 - p)$. As a result, the topology that originates after allocating the flows can be compared with the Erdös-Renyí random graph. The average hopcount for the Erdös-Renyí random graph is, for large N, approximated by [43, pp. 346],

$$\begin{aligned} \mathrm{E}[H_N] &\approx 2 - p + (1-p)(1-p^2)^{N-2} && (3.12) \\ &\approx 2 - p \qquad \text{for } N \to \infty && (3.13) \end{aligned}$$

where we have used that $\Pr[H_N = 3] \approx \Pr[H_N > 2]$. The linear behavior visible in Figure 3.13 then follows from (3.13). We have added computations of (3.13) to Figure 3.13. Equation (3.13) underestimates the simulation results, in particular for lower link densities. The difference stems from the fact that for lower link densities, the hopcount is

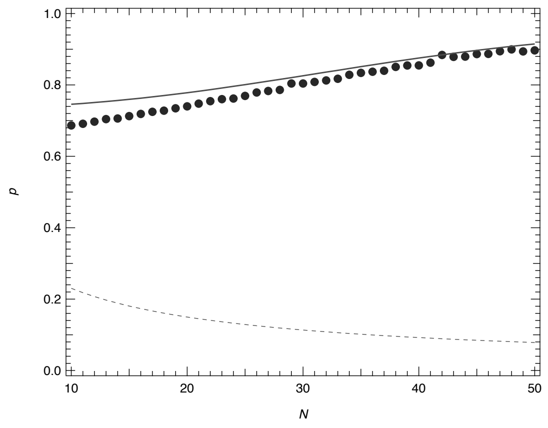


Figure 3.14: The link density in steady state. The simulations (dots) are compared with computations (solid line) of (3.14) using $E[N_S]$ results from Figure 3.9. The link density threshold p_c for the Erdös-Renyí random graph has been added to the figure (dotted line). Below p_c the graph is a.s. disconnected.

larger and dependencies between links become more pronounced. Using the average hopcount (3.13), the steady state link density (3.11) converts to

$$p \approx 2 - \frac{L_0}{L_0 - \mathrm{E}[N_S]} \tag{3.14}$$

Figure 3.14 shows the link density as a function of the number of nodes in steady state. Computations of (3.14) have been added using $E[N_S]$ from (3.4) where $K = \beta L_0$. Figure 3.14 illustrates that the first order estimate (3.14) agrees well. Equation (3.14) slightly overestimates the simulations for small N, which is caused by disregarding the probability of paths longer than 2 hops and the assumption that H_j is independent of $N_S(t)$. For larger networks, the probability of longer paths decreases and equation (3.14) better matches the simulations.

The link density is important as it inherently determines the blocking probability in the network. If the link density reaches beyond a critical value, the graph is almost surely disconnected and virtually each flow is blocked. Comparing the network with the Erdös-Renyí random graph, facilitates to express the critical link density p_c as a function of the network size N: $p_c(N) \approx \frac{1}{N} \ln N$. Computations of the critical link density have been added to Figure 3.14. Although the average link density is well above the critical threshold, the network still experiences blocking. Figure 3.15 explains this phenomenon. For a single sample path, i.e. a single network realization, the link density is plotted on the left axis and the number of rejected requests on the right axis as a function of time. Figure 3.15 shows that the average link density is well above the random graph's critical density. When the link density stays above the critical value, all flow requests are accepted. However, rejections still take place due to the high variations in the traffic.

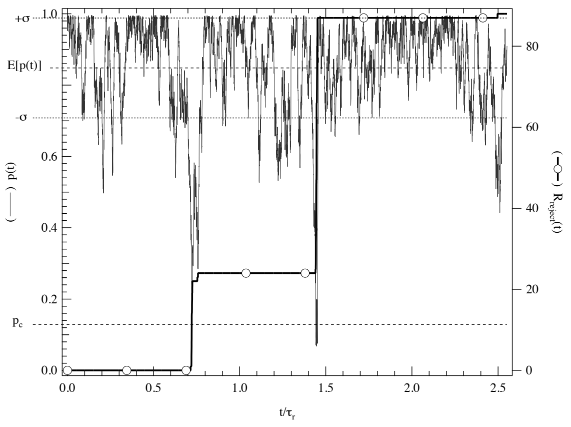


Figure 3.15: The link density and number of rejected requests as a function of time, for a single sample path, where $N = 25$ and $\rho = 0.99$. The upper dashed lines indicate the sampled mean $\mathrm{E}[p(t)] \pm \sigma$ (one standard deviation). The lower dashed line illustrates p_c for the Erdös-Renyí random graph with $N = 25$.

A few short bursts of strongly increased arrivals contribute most to the rejections by the network. At such moments, e.g. see Figure 3.15 at $t/\tau_r \approx 0.7$, the link density reaches far beyond the critical density p_c. Below the critical density, the network is a.s. disconnected and the probability that a flow is accepted is small. We have simulated a 10-node network and studied the allocation and release process by inspecting the topology. Figure A.1 in Appendix A presents the flow arrivals and departures for the 10-node network just before and after rejections are taking place. The first event shown in Figure A.1 is the 80th flow arrival. After the arrival of flow 81, the network becomes disconnected (node 5). Then 3 more flows are allocated before the first rejection occurs (flow 86). After the arrival of flow 91 the network connectivity reaches beyond the critical value and multiple disconnected nodes and clusters emerge. A remarkable observation is that in this phase the network still manages to allocate several flows. Due to the low connectivity, the hopcount of these flows is very large compared to the average case, see e.g. flow arrival 105. These observations imply that, beside the mean, also the tail probabilities are crucial.

3.5 The Erdös-Renyí random graph

In the previous section we have studied the network capacity for the complete graph with a single channel per link. The complete graph is entirely determined by the number of nodes and we have found relation (3.8) that describes the average number of flows as a function of the number of nodes in the graph. In this section we extend the results from the previous section by studying the class of Erdös-Renyí [17] random graphs. In the Erdös-Renyí random graph, the number of links is stochastic and determined by the probability of having a link between each pair of nodes. The Erdös-Renyí random graph is annotated as $G_p(N)$, where N denotes the number of nodes and p is the link probability. All links are considered independent. The average number of links is $\mathrm{E}[L] = pL_0$. Hence, the Erdös-Renyí random graph is described by the parameters N and p. In this section we investigate if the M/M/1/K approximation still holds and if we can write the network capacity K as a function of N and p. We examine how the link density, defined as $p = \mathrm{E}[L]/L_0$ affects the average throughput in the network. For K_N, the link density is always 1. For $G_p(N)$ the average link density equals p. We have performed simulations for link probability p ranging from 0.3 to 1.0 and N ranging from 10 to 50. The service rate $\mu = 1.00$, such that the load equals the arrival rate $\rho = \lambda = 0.99$. The link capacity is set at $C = 1$; each link only accommodates a single channel. For each combination of N and p we have simulated 10 randomized network realizations and issued $1,100,000$ flow requests per realization. At the start of each simulation, the network is empty. Since we are only focussing on the steady-state behavior, the initial $100,000$ flows of each realization are ignored in our analysis to account for the warmup phase.

3.5.1 Number of flows in the network

Since the Erdös-Renyí random graph is a subgraph of the complete graph, the average number of flows in the Erdös-Renyí random graph will never be greater than that of the complete graph. Hence, similar to the previous section, the average number of flows is bounded by $\mathrm{E}[N_S] = \mathrm{E}\left[N_{S_{M/M/1}}\right] = 99$. Figure 3.16a presents the average number of flows in the steady state as a function of N and p. The results have been obtained via simulation and are fitted with (3.4), where K is taken as (3.8) and β is the fit parameter. The goodness of the fits illustrates that the model also applies to the Erdös-Renyí random graph and that β is independent of N. However, the network capacity K *is* proportional to the number of links, hence K will be a function of the link probability p. Figure 3.16b presents the fit results for β as a function of p. The results for β in Figure 3.16b have been fitted with $\beta(p) = \beta_0 + A\,p^{\gamma}$. The constant term β_0 is taken zero, since for $p = 0$ the capacity must be zero. The scaling term A must be equal to $\beta(1) \approx 0.42$, hence we only need to find the power-exponent γ. The resulting value for γ is found as $\gamma \approx 1.34 \approx \frac{4}{3}$,

$$\beta(p) \approx 0.42\,p^{\frac{4}{3}} \tag{3.15}$$

Equation (3.15) indicates that β, and thus the network capacity, decreases rapidly with p. In the case of the fully connected network, the network capacity is approximately 42 percent of the number of links. In the case of the Erdös-Renyí random graphs, on average $42\,p^{1/3}$ percent of the links contribute to the network capacity. The decrease is a result of two combined phenomena. First, due to the smaller link probability, there are less resources available to carry the load, yielding a lower throughput. Second, since the link probability is smaller, the average hopcount is larger. Hence, each flow consumes, on average, more resources leading to a faster depletion of resources.

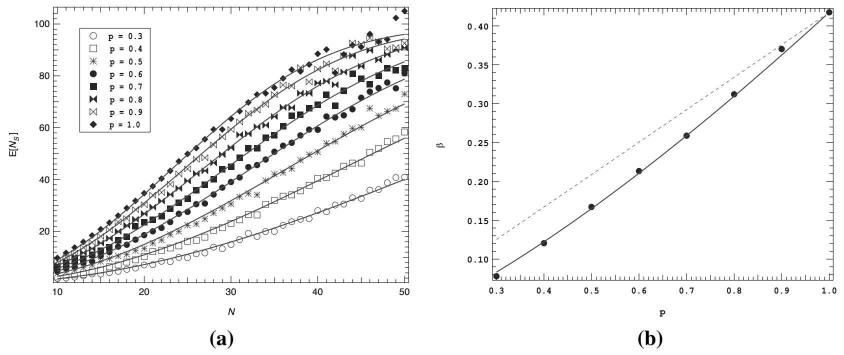


Figure 3.16: Simulation results for the Erdös-Renyí random graph. The number of nodes ranges from $N = 10$ to $N = 50$, the load $\rho = 0.99$ and link probability p is between $p = 0.3$ and $p = 1.0$. In (a) the average number of flows is shown in the steady-state as a function of N. The simulation results (dots) have been fitted with (3.4) (lines). The fit results for β are presented in (b). The simulation results (dots) have been fitted with $\beta(p) = \beta(1)p^{\gamma}$ (lines). The fit result is $\beta(1) \approx 0.42$ and $\gamma \approx 1.34$. As reference we have added the dashed line corresponding to $\gamma = 1$.

3.5.2 Hopcount

Figure 3.17 presents the average hopcount during steady-state. The average hopcount is computed as the sampled mean of the hopcount of the successfully routed flows. For N sufficiently large and the link probability well above the critical probability, the average hopcount in the Erdös-Renyí random graph is closely approximated by (3.13). From Figure 3.17 we can observe that the average path length often exceeds two. In particular for smaller networks, the link probability approaches the critical probability, yielding an increase in the average hopcount. Consequently, the Erdös-Renyí random graph model is inappropriate and we can no longer approximate the expected hopcount via equation (3.13). Figure 3.18 presents the link density in the steady state. At small link probabilities, the link density in the steady state appears to be the same for any value of N. For the Erdös-Renyí random graph, the link density in the steady state from equation (3.11) converts to,

$$p = p_0 - \frac{\mathrm{E}\left[N_S\right]\mathrm{E}\left[H_N\right]}{L_0}$$

3.5.3 Rejection rate

Rejections in the network are caused by the lack of resources. In our queueing model, this is analogous to the situation where the buffer is full and all K positions are occupied. Hence, the rejection ratio in the network can be approximated by using (3.7), where K is then the network capacity as defined by (3.8) with β given in (3.15). The rejection

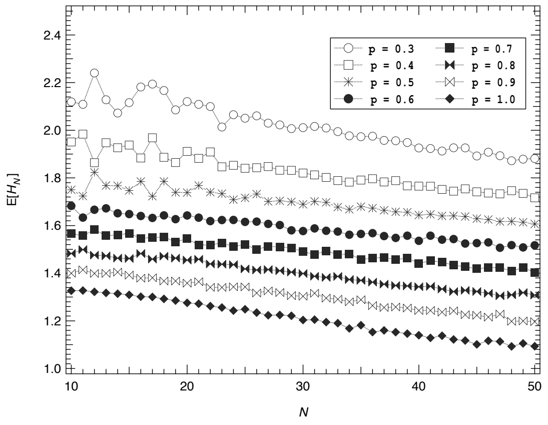


Figure 3.17: The average hopcount in the steady state as a function of N for the Erdös-Renyí random graph, where the number of nodes ranges from $N = 10$ to $N = 50$, load $\rho = 0.99$ and link probability $p = 0.3, 0.4, \ldots, 1.0$.

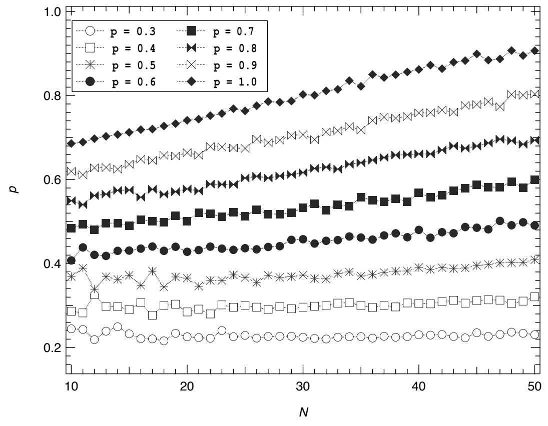


Figure 3.18: The average link density in the steady-state as a function of N for the ER random graph, where the number of nodes ranges from $N = 10$ to $N = 50$, load $\rho = 0.99$ and link probability $p = 0.3, 0.4, \ldots, 1.0$.

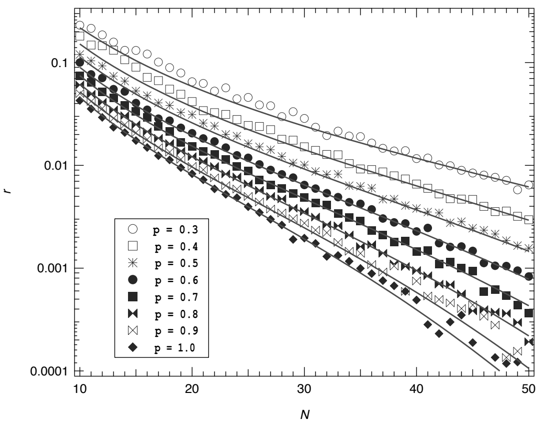


Figure 3.19: The rejection rate in steady-state as a function of L_0 for the Erdös-Renyí random graph, where the number of nodes ranges from $N = 10$ to $N = 50$, load $\rho = 0.99$ and link probability $p = 0.3, 0.4, \ldots, 1.0$. The simulation results (dots) have been fitted with (3.7) (lines), with $K = \beta L_0$ and β the fit parameter. The fit results for β were within four percent difference of the results from Section 3.5.1.

ratio has been measured during the simulations by counting the flow requests that are rejected divided by the total requests. The steady-state rejection ratio has been plotted in Figure 3.19 and the simulation results have been fitted with (3.7). The resulting β of these fits has been compared to the results from Section 3.5.1. As shown in Figure 3.19, the fit is good. The small differences are likely due to the fitting process and sampling accuracy.

3.6 The complete graph with multiple channels per link

In this section we study the effect of increasing the number of channels per link on the network capacity. Increasing the number of channels makes the model more realistic if we compare it with optical networks, where a fiber is split into a number of wavelengths. Blocking now only occurs if *all* channels in the fiber are allocated. Increasing the number of channels makes the network less susceptible for disconnectedness and consequently exhibits less extreme behavior in the sense that one flow immediately blocks all the other traffic. With respect to the topology, we shall confine ourselves to the complete graph K_N such that we can isolate the effect of the number of channels on the network capacity. Note that the resulting network is deterministic since the number of links and number of channels per link is deterministic. The service and arrival rate are taken as in Section 3.5, such that $\rho = 0.99$. The link capacity C is ranging from 1 to 8 channels per link. We have simulated 10 network realizations with 1,100,000 flow requests per realization. The steady-state values are obtained by sampling the last $1,000,000$ flows.

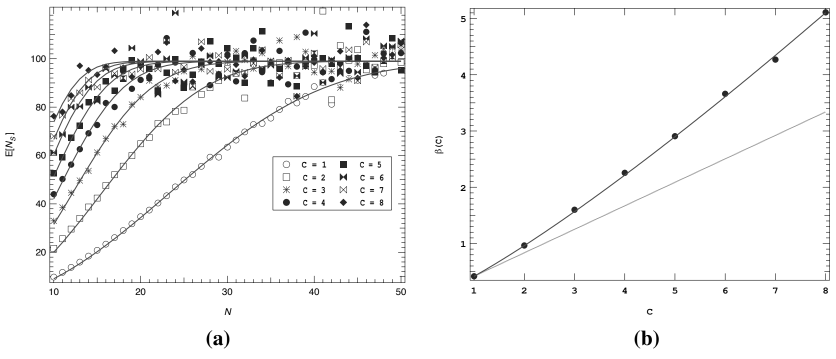


Figure 3.20: (a) The average number of flows in the steady state as a function of the number of nodes. The number of nodes ranges from $N = 10$ to $N = 50$, the load equals $\rho = 0.99$ and the link capacity ranges from $C = 1$ to $C = 8$. The simulation results are fitted with (3.4), the fit parameter $K = \beta L_0$. The dots represent the simulations while the lines are the curve fits. (b) β as a function of C. The simulation results (dots) have been fitted with $\beta = \beta(1)C^{\gamma}$. The dotted line corresponds to the linear function $\beta = \beta(1)C$.

3.6.1 Number of flows in the network

The average number of flows in the steady-state is plotted in Figure 3.20a. As in Section 3.5, the average is upper-bounded by (3.1) at 99. The simulation results have been fitted with (3.4). The fit result is presented in Figure 3.20a as a line connecting the dots. The good fit illustrates that the average number of flows in the system in the steady state can be well approximated by the M/M1/K queueing model. The resulting values for β are displayed in Figure 3.20b as a function of C. We have fitted $\beta(C)$ with $y = \beta(1)C^{\gamma}$, where $\beta(1) \approx 0.42$. The result is displayed as a line in Figure 3.20b. The fit result $\gamma \approx 1.20 \approx \frac{6}{5}$, such that

$$\beta(C) \approx 0.42 C^{\frac{6}{5}} \tag{3.16}$$

The fit result demonstrates that the network capacity grows more than linear with C. The non-linear capacity gain that is achieved when using multiple channels per link is comparable to the performance gain achieved in optical networks with wavelength changers. In optical networks without wavelength changers, a lightpath from source to destination must obey the “continuity constraint”, meaning that from source to destination, the wavelength of the optical path must be same on each link. An optical network without wavelength changers and with C wavelengths per fiber can then be considered as a aggregation of C independent networks. The performance of such a network, e.g. measured in throughput, is then a multiple of the performance of a single network. If wavelength changers are added to the network, the wavelength of the optical path can changed at the nodes, such that the routing process has more freedom in choosing the path. Hence, more paths are available yielding a better performance. The variance of the number of flows

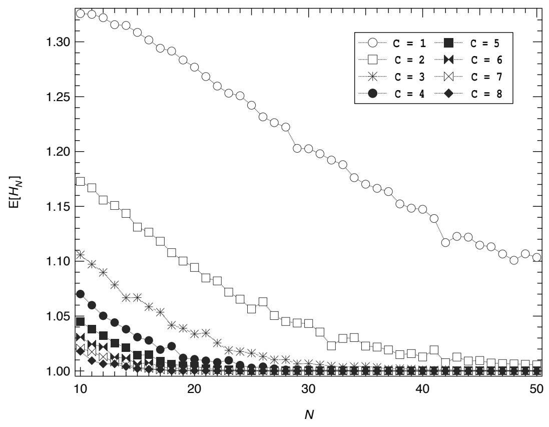


Figure 3.21: The average hopcount in the steady-state for the complete graph with multiple channels per link as a function of N. The number of nodes ranges from $N = 10$ to $N = 50$, the load equals $\rho = 0.99$ and the number of channels ranges from $C = 1$ to $C = 8$. For clarity, the results for networks with the same link capacity are connected with a dashed line.

also increases with both N and C, resulting in the intensified fluctuations for large N and C in Figure 3.20a.

3.6.2 Hopcount

Increasing the number of channels per link reduces the probability that the shortest path between a source-destination pair is blocked, hence it improves the resilience of the link against congestion. The average hopcount will decrease when increasing C. Figure 3.21 presents the average hopcount during steady-state. The average hopcount is monotonously decreasing for increasing N and C. When N increases, the number of links grows accordingly, leading to an increase of alternate routes between node-pairs. In addition, the number of source-destination pairs increases such that the direct link from source to destination is often still available. Increasing the number of channels per link quickly reduces the average hopcount to one, which implies that a direct link from source to destination is available in most cases.

3.6.3 Rejection rate

The simulation results for the rejection ratio are displayed in Figure 3.22. The results are compared with computations of (3.7), with K as in (3.16). Figure 3.22 illustrates that the simulation results are in agreement with the computations. Besides the average number of flows, we are also able to model the rejection rate of this network by the M/M/1/K queueing analogy. For larger values of C, the fluctuations increase, which is in agreement with the observations on the average number of flows. This effect is already

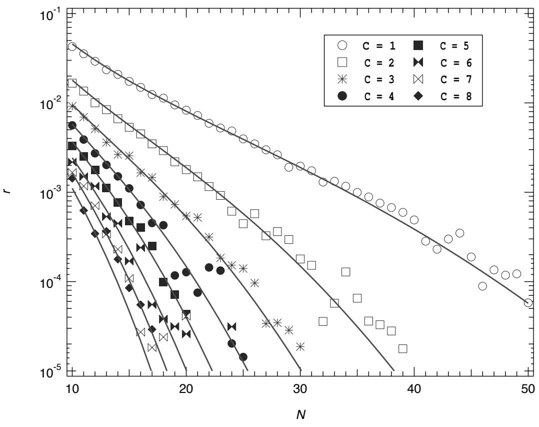


Figure 3.22: The rejection ratio in the steady-state for the complete graph with multiple channels per link as a function of N. The number of nodes ranges from $N = 10$ to $N = 50$, the load equals $\rho = 0.99$ and the number of channels $C = 1,2,3,4$. The simulation results (dots) have been fitted with (3.7) (lines), with $K = \beta L_0$ and β being the fit parameter.

visible in Figure 3.22 for small values of C, but intensifies as the number of channels increases.

3.7 The two-dimensional lattice

In this chapter we will study two-dimensional lattice networks, where the nodes are placed on a rectangular grid and, apart from the border nodes, all nodes are linked with four neighbors. The link capacity is fixed for all links at one channel per link. Networks with a lattice topology are quite common, e.g. in grid computing. Another application is in silicon on chips where point connections are made on a grid layout. We have simulated two-dimensional square lattice topologies with Z nodes in each dimension, having a total of $N = Z^2$ nodes. In the simulations, Z ranges from $Z = 4$ to $Z = 50$ and each link contains a single channel, $C = 1$. Note that for $Z = 50$, the network contains $N = Z^2 = 2500$ nodes and $L_0 = 2Z(Z-1) = 4900$ links. The load is fixed at $\rho = 0.99$. We have simulated 10 network realizations and issued $1,100,000$ flows per realization. The first $100,000$ flow requests are disregarded in the steady state analysis.

3.7.1 Number of flows in the network

The average number of flows in the steady state is plotted in Figure 3.23 against the size of the lattice Z. The average number of flows is significantly smaller than in the networks described in previous sections. The lower throughput stems from the, on average, long paths between source and destination. In the case of the lattice graph, the nodes are

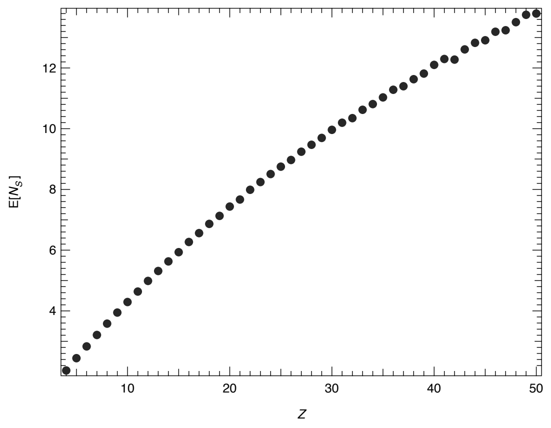


Figure 3.23: The average number of flows in steady-state for a two-dimensional square lattice with one channel per link. The dimensions of the lattice are $Z \times Z$, with $Z \in [4,50]$.

placed in a grid, yielding a maximum of four neighbors. The fixed structure of the lattice enforces long paths, such that the available resource are rapidly exhausted. In the case of random graphs, each node has a fairly equal chance of being selected as a hop, unlike the lattice graph, where the next hop can only be selected from maximally two neighbors, as the other two neighbors will only increase the path length. Moreover, since we are dealing with paths between uniformly chosen nodes, the likeliness that the path crosses the center of the lattice is high. If we compute the betweenness centrality[2] [19] of each node, then the nodes in the center exhibit the highest betweenness. Hence, the center rapidly becomes congested, which then becomes a critical and heavily loaded area. This effect decreases the chance of a successful route attempt for future flows, since these are also likely to cross the center section. In addition, most nodes have a degree of 4. In the general, and worst, case, the node serves as an intermediate hop along a path and as a result, the node becomes “congested” after the allocation of two flows. We have tested the network model on the lattice and fitted the simulation results from Figure 3.23 with (3.4). The simulations did not fit with the M/M/1/K queue, meaning that for the lattice we have not found an expression for the network capacity. Hence, the fixed structure of the lattice prohibits the application of the queueing model on the network.

[2]The betweenness centrality measure counts the number of shortest paths through a node or link w.r.t. the total number of node pairs.

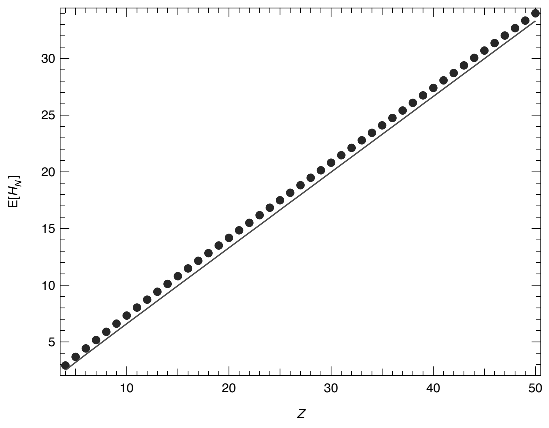


Figure 3.24: The average hopcount in steady-state for a two-dimensional lattice. The solid markers represent the simulation results. The solid line represents computations of (3.17).

3.7.2 Hopcount

The mean hopcount for the two-dimensional square lattice is given by eq. (3.17), see [43, pp. 513],

$$E[H_N] = \frac{2}{3Z}\left(Z^2 - 1\right) \tag{3.17}$$

Eq. 3.17 expresses the minimal distance between two nodes in a square lattice averaged over all the possible nodepairs. Hence, (3.17) is a lower-bound on the average hopcount in the steady-state, loaded network. Figure 3.24 presents the average hopcount for the two-dimensional lattice network in steady state. The markers correspond to the average hopcount obtained through simulation. The solid line represents calculations of (3.17). Figure 3.24 illustrates that, as expected from (3.17), the average hopcount for lattice networks follows an almost linear increase with Z. The simulation results for the square lattice slightly underestimate (3.17), which is caused by the traffic in the network: due to the existing connections, longer paths must be utilized to reach the destination.

3.7.3 Rejection rate

Figure 3.25 shows the rejection ratio for the two-dimensional square lattice. The rejection ratio seems to exhibit a power-law behavior with respect to Z and contrasts the exponentially decaying rejection ratio in M/M/1 queues. Increasing the network size only leads to a marginal decrease of the rejection rate. Increasing the network size implies an increase of the average hopcount, such that each flow consumes more resources. The bottleneck is formed by the center of the lattice, which still suffers from rapid exhaustion of resources. We did not succeed in finding an explanation for the power-law relation with respect

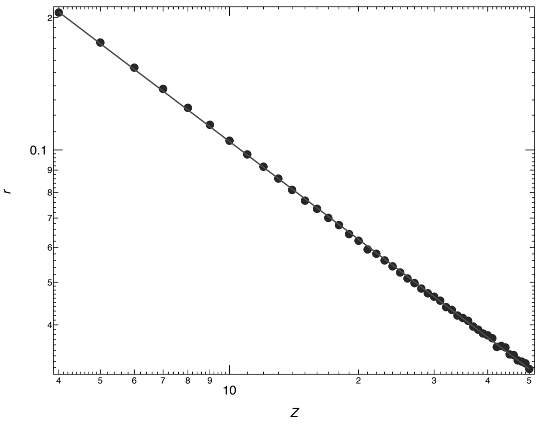


Figure 3.25: The rejection ratio in steady-state, for the two-dimensional square lattice with dimensions $Z \times Z$, as a function of Z. The simulation results (dots) have been fitted with a power-law (line), the fit result is $r = 0.57Z^{-0.74}$.

to Z. We believe that a rigorous proof will be very complex, when we realize that the uncorrelated percolation process [22] is already very complex.

3.8 Traffic dependency

In the previous sections we have studied the change in network capacity for networks with different topological characteristics. This section is devoted to the dependency of the network capacity on the traffic conditions. We conduct our analysis with the basic network scenario, which is the fully connected network with a single channel per link. First, we study the network model where the load is variant. Second, we choose another traffic model, where the service times are deterministic instead of stochastic.

3.8.1 Varying the load

The load on the network corresponds here to the intensity of the offered traffic, i.e. the ratio of the average service time and average inter-arrival time of the flows, $\rho = \mathrm{E}[T_s]/\mathrm{E}[T_a] = \lambda/\mu$. The average inter-arrival time is fixed at $\mathrm{E}[T_a] = 1$, the load then equals the average service time. In the previous sections the load was fixed at $\rho = 0.99$. In this section the load ranges from $\rho = 0.85$ to $\rho = 1.00$. The number of nodes ranges from $N = 10$ to $N = 40$. For each combination of N and ρ we have simulated 10 network realizations and $1,100,000$ flow requests have been issued per realization. The first $100,000$ flow requests have been omitted from the steady state analysis to account for the warmup phase. Figure 3.26 presents the simulation results of the average number of flows in the steady state. The simulation results are compared with computations of (3.4), where the same

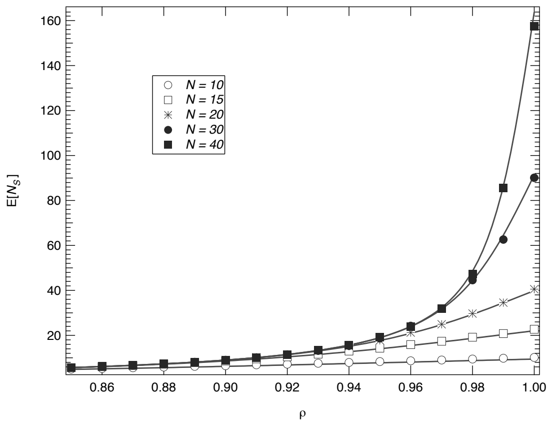


Figure 3.26: The average number of flows in steady state as a function of the traffic load for networks of various sizes. The simulation results (dots) are compared with computations of (3.4) (lines) using $K = 0.42\binom{N}{2}$.

value for the network capacity is used as in Section 3.4. Figure 3.26 illustrates that the simulation results closely follow the computations. When the load is varied between 0.85 and 1.00, the average number of flows in the network is still accurately described by (3.4). Moreover, the network size appears to play no effect on the closeness of (3.4) when the load is varied. The agreement indicates that the network capacity is not dependent on the load, i.e. β and consequently K are not dependent on the traffic intensity ρ. The same behavior is observed when the rejection rate is studied. Figure 3.27 presents the average rejection rate in the steady state as a function of the traffic load ρ. The simulation results in Figure 3.27 are in close agreement with the computations of (3.7). For $N = 10$, the computations slightly overestimate the simulation results, which may be due to the small size of the network. However, the difference appears to be the same for all the values of ρ. These findings confirm our previous conjecture that the network capacity is determined by the topology and is independent of the traffic characteristics. In the following section we will study the influence of the service process on the network capacity.

3.8.2 Deterministic service times

In this section we verify equation (3.8) in the case where the service times are deterministic. If equation (3.8) still holds in the case of deterministic service times, then the claim that the network capacity is indeed determined by the topology alone becomes stronger. In comparison to the previous sections, the service process then loses the Poisson property and the agreement of the network with the M/M/1/K queue no longer applies. When the network has deterministic service times, the service time is fixed such that we need to consider the M/D/1/K queue as an analogue for the network model. The analytic expressions for the average number of flows or the rejection rate in the steady state are

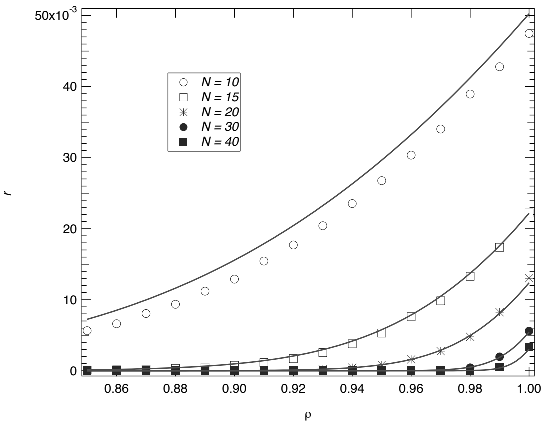


Figure 3.27: The average rejection rate in the steady state as a function of the traffic load for networks of various sizes. The simulation results (dots) are compared with computations of (3.7) (lines) using $K = 0.42\binom{N}{2}$.

significantly more complex with respect to the case of a Poisson service process. Analytic solutions for the average system size of the M/D/1/K queueing system in the steady state are found by Brun and Garcia [14],

$$\mathrm{E}\left[N_{S_{M/D/1/K}}\right] = K - \frac{\sum_{n=0}^{K-1} b_n}{1+\rho\, b_{K-1}} \tag{3.18}$$

with

$$b_n = \sum_{k=0}^{n} \frac{(-1)^k}{k!}(n-k)^k e^{(n-k)}\rho^k$$

The rejection probability is now given by,

$$r_{M/D/1/K} = 1 - \frac{b_{K-1}}{1+\rho\, b_{K-1}} \tag{3.19}$$

We have performed simulations on the complete graph with a single channel per link and the number of nodes ranging from $N = 10$ to $N = 50$. The traffic intensity equals $\rho = 0.99$. For each network realization, we have issued $1,100,000$ flow requests. The first $100,000$ flows are ignored during the steady state analysis to compensate for the warmup phase. Figure 3.28 presents the average number of flows in the steady state. The simulation results are compared with computations of (3.18), where $K = 0.42\binom{N}{2}$. The correspondence of the simulation results with the computations is again very good. The average number of flows is accurately described by the M/D/1/K queueing analogy. In addition we have measured the rejection rate in the steady state and the results are presented in Figure 3.29. The computations of the rejection rate for the M/D/1/K

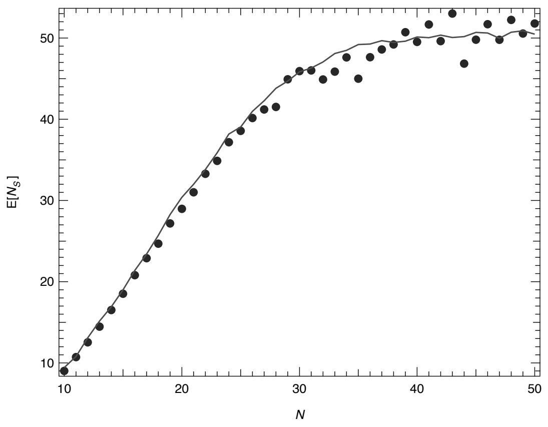


Figure 3.28: The average number of flows in the steady state for the network with deterministic service times. The simulation results (dots) are compared with computations of (3.18) (lines). The load is chosen as $\rho = 0.99$, the network capacity is computed as $K = 0.42\binom{N}{2}$.

queue (3.19) are very close to the simulation results. Hence, both the average number of flows as the rejection rate are accurately described by the M/D/1/K queueing analogy.

3.9 Chapter summary

This chapter investigates the traffic dynamics in networks at the flow level. The relation between the network and traffic characteristics is modeled using a queueing approach. We found that for the case of several networks, the average number of flows and the rejection rate can be modeled quite well by a finite-sized queue. In this approach, the network is regarded as a black box, where flows arrive at stochastic intervals and are being served, similar to a queueing system. In analogy with the buffer size in queueing systems, we define the “network capacity” as the maximum number of flows the network can sustain. In contrast to the buffer size in queueing systems, the network capacity is an emergent stochastic quantity that is not determined by definition but by the outcome of various network properties and the interaction of processes during the network operation, e.g. the routing process, the selection of source and destination nodes and of course topological properties.

In this chapter, we have applied the model on several network classes, including the fully connected graph with a single channel per link, the Erdös-Renyí random graph, the fully connected graph having multiple channels per link and the two-dimensional lattice with a single channel per link. In the case of the fully connected network with a single channel per link, we can view the network as a regular M/M/1/K queue and the (transient) solutions for the M/M/1/K queue can be applied to describe network behavior and provide insight into the loading of the network as well as blocking. A new finding is

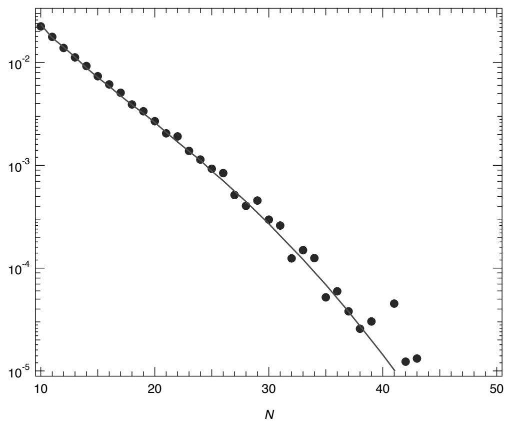


Figure 3.29: The average rejection rate in the steady state for the network with deterministic service times. The simulation results (dots) are compared with computations of (3.19) (lines). The load is chosen as $\rho = 0.99$, the network capacity is computed as $K = 0.42\binom{N}{2}$.

Class	β
Fully connected graph with single channel per link	β_0
Erdös-Renyí random graph with single channel per link	$\beta_0\, p^{4/3}$
Fully connected graph with multiple channels per link	$\beta_0\, C^{6/5}$
Two-dimensional lattice with single channel per link	n.a.

Table 3.1: Summary of the results for the scaling factor β for different classes of networks, where $\beta_0 = 0.42$. In the case of the two-dimensional lattice, the queueing model can not by applied, hence there is no value for β.

the linear relation $K = \beta\binom{N}{2}$ between the number of links in the network and the buffer capacity K. The origin of the scaling factor β and the sensitivity to network scaling, e.g. increasing the number of channels per link so that more flows are allowed per link, is investigated in Sections 3.5 and 3.6. In Section 3.7 we have studied the applicability of the model to the lattice graph. Table 3.1 presents a summary of the results on the network capacity. The capacity of the network appears dependent on the link density in case of the Erdös-Renyí random graph, where the scaling factor β decreases with the link probability p. This effect can be ascribed to the fact that sparser graphs are less robust since fewer alternate routes exist. The successive allocation of flows exhausts the resources more quickly and the network becomes more prune to disconnectedness. Additionally, the average hopcount increases for lower link probabilities, such that on average more links are used per flow and consequently less flows can be allocated. We also found that the scaling factor β increases with the number of channels per link. As each link has more channels, the average hopcount decreases and the probability of finding a direct link between source and destination increases. The network is more robust, as more alternate routes are available.

The meaning of β can be explained as the “efficiency” with which the links are allocated. By allocating links, parts of the network become mutually unreachable, compromising the maximum number of sustainable flows. Table 3.1 illustrates that the network is maximally loaded up to some 42 percent of its actual resources. A closer inspection of the transient behavior of the network revealed that most of the rejections occur due to the traffic fluctuations in the network. At moments of heavy bursty traffic, the network becomes congested leading to a surge of rejections. In the case of the lattice network, the queueing model is not applicable. We conjecture that the main reason lies in its geometric nature: the average hopcount is much higher than for Erdös-Renyí random graphs and consequently, blocking is more likely to occur. Moreover, the paths between the uniformly chosen nodes cross the center-section of the network, dividing the network into mutually unreachable clusters. The effect of correlation between the links becomes more clear when the rejection ratio is compared with a percolated lattice network. The rejection ratio is significantly higher than when there is no correlation and exhibits a power-law in the size of the network.

Section 3.8 demonstrates that the network capacity (and therefore β) is not determined by the traffic load. The network capacity seems embedded in the topological properties and is not determined by the traffic intensity or by the service process.

Chapter 4

An upper-bound on the maximum number of flows

4.1 Introduction

In Chapter 3 we have studied the capacity of a network with continuously arriving and departing flows. In this section, the average number of flows that can be allocated in a network before disconnectedness occurs, is addressed. The network is considered disconnected when at least one node is unreachable for any other node. Additionally, we are interested in the effect of removing paths in the graph. More precise, we analyze the evolution of the connectivity and the rejection rate as flows are allocated in the network.

4.2 The fully connected graph

The network model that is used throughout the chapter is the following. The network consists of N homogeneous nodes, which are interconnected by L identical, undirected links. We start our analysis with the complete graph with unit link weights. We choose two nodes uniformly and compute the shortest path using Dijkstra's shortest path algorithm. The paths are then fully occupied for an infinite duration, which is equivalent to removing the path's links from the graph. We repeat the process of choosing nodes and removing links whilst the graph remains connected. The process ends immediately after the first flow is allocated that disconnects the graph. The graph with N nodes, after removal of j paths is denoted by $\hat{G}(j,N)$. Figure 4.1 presents the meta-code of the subsequent steps to construct $\hat{G}(j,N)$. Initially, the graph $\hat{G}(j,N)$ equals the full mesh and between every node pair there exists a link. When j is small, the average length of the allocated paths equals one with high probability and single links are removed independently from each other. Hence, the resulting graph $\hat{G}(j,N)$ corresponds precisely with the Erdös-Renyí random graph $G_p(N)$. The correspondence with $G_p(N)$ allows us to apply some of the well-known results for $G_p(N)$ to $\hat{G}(j,N)$ The probability that the graph is connected after the allocation of j flows, is directly related to the link density in the graph. An exact computation of the link density $p[j]$ of $\hat{G}(j,N)$ is hardly possible since, as will be shown, it assumes that we know the hopcount distribution $\Pr[H=k]$ in the Erdös-Renyí graph, which is at present still an open problem. Assuming that $\Pr[H>2]=\varepsilon$ and sufficiently small to ignore, we have that [43, p. 346],

$$\begin{aligned} E[H(p)] &\geq 2-p+(1-p)(1-p^2)^{N-2} && (4.1)\\ &\geq 2-p+\varepsilon && (4.2) \end{aligned}$$

where ε is the correction for ignoring paths of more than 2 hops. By definition of the link density $p[j]$, the average number of links in $\hat{G}(j,N)$ as a function of j is $E[L[j]]=L_0p[j]$, where L_0 is the number of links in the initial situation. The initial conditions for $p[j]$ are $p[0]=1$ and $p[1]=1-\frac{1}{L_0}$. Using (4.2) with $\varepsilon=0$, the average number of links at stage $j+1$ can be written as

$$\begin{aligned} E[L[j+1]] &= L_0\,p[j]-E[H(p[j])] && (4.3)\\ &\leq L_0\,p[j]-(2-p[j]) \end{aligned}$$

This difference equation converts to

$$p[j+1]=\left(1+\frac{1}{L_0}\right)p[j]-\frac{2}{L_0}$$

Solving (4.3) yields

$$p[j]\leq 2-\left(1+\frac{1}{L_0}\right)^j \tag{4.4}$$

Relation (4.4) forms a tight upper-bound for the true link density, because it follows from (4.2) that $\varepsilon\geq 0$ and consequently from (4.3) it follows that $p[j]$ is maximum when $\varepsilon=0,\forall j$.

An upper-bound on the maximum number of flows is now obtained by considering the critical link density p_c of $G_p(N)$. In the extreme case, where each flow has

```
GENERATE $\hat{G}(j,N)$
1  INITIALIZE($K_N$)
2  **while** $\hat{G}(j,N)$ is connected
3     **do** node $v \leftarrow$ RANDOM-NODE($\hat{G}(j,N)$)
4        node $u \leftarrow$ RANDOM-NODE($\hat{G}(j,N)$)
5        path $P \leftarrow$ DIJKSTRA($\hat{G}(j,N),v,u$)
6        remove $P$ from $\hat{G}$
```

Figure 4.1: Meta-code of GENERATE for the construction of $\hat{G}(j,N)$.

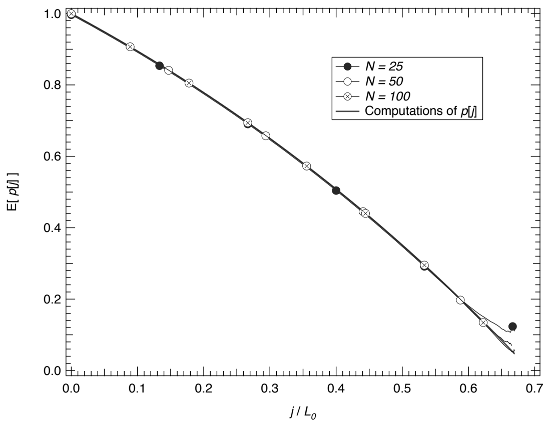


Figure 4.2: The link density in $\hat{G}(j,N)$ for $N = 25$, $N = 50$ and $N = 100$ after allocating j paths. Simulations (lines) are compared with computations of (4.4) (markers). The number of paths on the x-axis is normalized with respect to the number of links in the network.

a unique source-destination pair, the graph $\hat{G}(j,N)$ is precisely $G_p(N)$ and the graph is a.s. connected if $p[j] \geq p_c \approx \frac{\ln N}{N}$. In any other case, the probability that the direct link exists between source and destination decreases as more flows are allocated. Consequently, the average path length increases more than linear with j and $\Pr[H > 2]$ becomes non-negligible for large j. Moreover, the critical link density of $\hat{G}(j,N)$ is affected by the correlation between the links such that $p_{c_{\hat{G}}} \geq p_c$, which we will show later. Hence, for $\hat{G}(j,N)$, an estimate for the maximum number of flows is obtained from (4.4) when $p[j_{max}] \geq p_c$. Thus,

$$j_{max} \leq \frac{\ln(2 - p_c - \varepsilon)}{\ln(1 + 1/L_0)} \approx L_0 \ln\left(2 - \varepsilon - \frac{\ln N}{N}\right) \rightarrow 0.69 L_0 \tag{4.5}$$

Relation (4.5) presents an upper-bound on the number of flows that can be allocated in any network where each link can support a single flow. The upper-bound improves as the number of nodes increases and $\varepsilon \rightarrow 0$.

4.2.1 Distribution of the maximum flows

For several N ranging from 25 to 100 we have simulated 100,000 realizations of $\hat{G}(j,N)$. The simulation results for the link density are presented in Figure 4.2, together with computations of relation (4.4). The simulation results are nearly indistinguishable from the computations, except when j becomes large and N is small. The strong agreement supports our conjecture that $\hat{G}(j,N)$ closely resembles $G_p(N)$. The expected hopcount

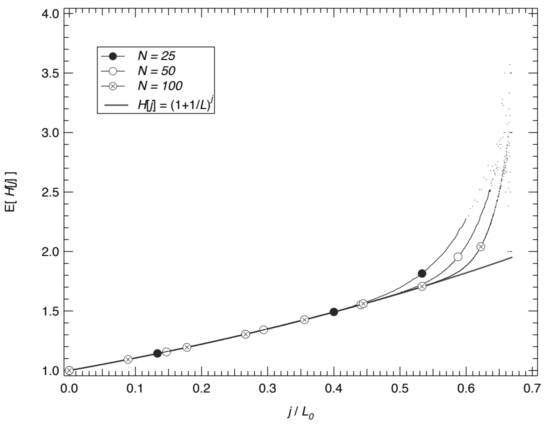


Figure 4.3: The average hopcount of $\hat{G}(j,N)$ for $N = 25$, $N = 50$ and $N = 100$ as function of j compared with computations of (4.6). The number of paths on the x-axis is normalized with respect to the number of links in the network.

in $\hat{G}(j,N)$ follows from (4.4)

$$\mathrm{E}\left[H\left[j\right]\right] \approx \left(1 + \frac{1}{L_0}\right)^j \tag{4.6}$$

Figure 4.3 presents the average hopcount measured during the simulations. The results show that (4.6) initially predicts the expected hopcount very well, but increasingly deviates as more flows are allocated. The dispersion in the tails in Figure 4.3 stems from the stochastic nature of the simulations. The number of samples that is available to compute the mean hopcount is determined by j_{max}. Because j_{max} is a stochastic variable, less samples are available at the tail, yielding a higher variance.

The reason for the strong agreement of the simulation results with (4.4) lies in the ability to establish the difference equation (4.3). The difference equation describes the "state" of the network in terms of the link density. When a flow is allocated the state is changed by decreasing the link density proportional to the hopcount. For the full mesh network, this is possible since the expected hopcount can be fully expressed in terms of the link density. For other classes of networks, such as lattice networks, the expected hopcount is different from (4.6) and (4.4). An exception is the class of Erdös-Renyí random graphs, as will be shown in Section 4.3.

The maximum number of paths, j_{max}, that can be removed from $\hat{G}(j,N)$ before becoming disconnected, is a stochastic variable and may differ for each realization of $\hat{G}(j,N)$. Each realization of $\hat{G}(j,N)$ is independent and identical to the other realizations. The randomness is introduced by the node-selection process and the routing (when multiple paths are found with the same length, one path is chosen arbitrarily). We have plotted $\Pr\left[j_{max} = j\right]$ in Figure 4.4 and fitted the result with the Gumbel distribution density

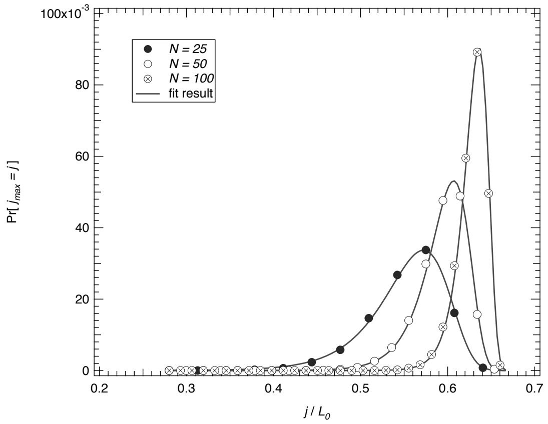


Figure 4.4: The probability distribution function of the maximum number allocated paths $\Pr[j_{max} = j]$ in $\hat{G}(j,N)$. The simulations (dots) have been fitted with the Gumbel distribution from equation (4.7) (lines). The number of paths on the x-axis is normalized with respect to the number of links in the network. The results of the fits for $N = 25$ are $a \approx 11, b \approx 172$, for $N = 50$ we obtained $a \approx 28, b \approx 744$ and finally for $N = 100$ we have $a \approx 66$ and $b \approx 3148$.

function [43, pp. 54],

$$\Pr[j_{max} = j] = \frac{1}{a} e^{\frac{j-b}{a}} e^{-e^{\frac{j-b}{a}}} \tag{4.7}$$

The mean equals $\mathrm{E}[j_{max}] = b + \gamma a$, where $\gamma \simeq 0.5772$ is Euler's constant, and the variance follows as $\mathrm{Var}[j_{max}] = \frac{1}{6}(a\pi)^2$. Figure 4.4 demonstrates that the Gumbel distribution fits the simulations very well. The result of the fits is contained in the caption. The Gumbel distribution is an extreme value distribution, which arise as a limiting distribution for maximums or minimums of a sample of independent, identically distributed random variables. Although mathematically we cannot account for the Gumbel distribution entirely, the fit is very good and intuitively correct as j_{max} is also an "extreme value" that measures the maximum number of allocated paths. For $N = 25$, the fitted mean $\mathrm{E}[j_{max}] \approx 178$, in comparison with relation (4.5), $j_{max} \approx 187$. The standard deviation is very small compared to the mean and, therefore, the mean $\mathrm{E}[j_{max}]$ sufficiently describes the random variable j_{max}. Additionally, Figure 4.4 confirms that (4.5) improves as the network size increases. While for $N = 25$ nodes the deviation is still quite large, the approximation for $N = 100$ nodes is already fairly accurate.

4.2.2 Connectivity of $\hat{G}(j,N)$

In this section, we study the connectivity of $\hat{G}(j,N)$ and compare it with $G_p(N)$. We use the distribution of j_{max} from relation (4.7), and formulate an expression for $\Pr[p[j_{max}] \leq p[j]]$.

Then, we explain the cause of the discrepancy between $G_p(N)$ and $\hat{G}(j,N)$ by means of the degree distribution. The value j_{max} corresponds to the largest value of j for which $\hat{G}(j,N)$ is precisely connected. Hence, we can write,

$$\begin{aligned}\Pr\left[\hat{G}(j,N) \text{ is connected}\right] &\approx \Pr\left[j \le j_{max}\right] \\ &\approx 1 - \Pr\left[j_{max} \le j\right] \\ &\approx e^{-e^{\frac{j-b}{a}}} \end{aligned} \tag{4.8}$$

To obtain an expression for $\Pr[p[j_{max}] \le p[j]]$, substitute the inverse of (4.4) into (4.8). The link density threshold $p[j_{max}]$ corresponds to the smallest link density at which $\hat{G}(j_{max},N)$ is precisely disconnected. Hence, $p[j_{max}]$ in $\hat{G}(j_{max},N)$ can be regarded as the equivalent of p_c in $G_p(N)$. Thus,

$$\begin{aligned}\Pr\left[\hat{G}(j,N) \text{ is connected}\right] &\approx \Pr\left[p\left[j_{max}\right] \le p[j]\right] \\ &\approx \Pr\left[j_{max} \le \frac{\ln(2-p[j])}{\ln(1+1/L_0)}\right] \\ &\approx e^{-\left(\frac{2-p[j]}{(1+1/L_0)^b}\right)^{-a\ln(1+1/L_0)}} \end{aligned} \tag{4.9}$$

Relation (4.9) corresponds to a shifted Weibull distribution [43, pp. 55] with mean $\mathrm{E}\left[p\left[j_{max}\right]\right] = 2 - (1+1/L_0)^b\,\Gamma(1+a\ln(1+1/L_0))$, where $\Gamma(x)$ represents the Gamma function. In comparison, the connectivity of $G_p(N)$ is obtained from [43, pp. 334–337],

$$\begin{aligned}\Pr\left[G_p(N) \text{ is connected}\right] &\approx \left(1-(1-p)^{N-1}\right)^N \\ &\approx e^{-Ne^{-(N-1)p}} \quad \text{, for small } p \end{aligned} \tag{4.10}$$

Relation (4.10) also approximates an extreme value distribution, yet it does not obey the Weibull distribution from (4.9). Figure 4.5 compares computations of (4.9), taken from the simulation results, with computations of (4.10). The distribution in (4.9) exhibits a wider transition region and a larger mean as compared to (4.10), which yields that $p[j_{max}] \ge p_c$. The cause of the different behavior becomes apparent when the nodal degree distribution is considered.

The nodal degree is measured by sampling one uniformly chosen node after each flow allocation. Figure 4.6 presents the degree distribution for $\hat{G}(j,N)$ with $N = 50$ nodes and several values of j. For comparison, the degree distribution of $G_p(N)$ has been computed with the same mean and added to Figure 4.6. The degree distribution of $\hat{G}(j,N)$ deviates from the binomial distribution of $G_p(N)$ because the links in $\hat{G}(j,N)$ are not independent. The dependencies emerge when paths are removed that span multiple links. The links that belong to a path are correlated, because they must be connected to form the path. In Appendix C the effect of the dependencies on the degree distribution is further explained and computed. The degree distribution of $\hat{G}(j,N)$ exhibits a larger variance w.r.t. the binomial distribution corresponding to $G_p(N)$. The larger variance then explains the wider transition region in the connectivity functions displayed in Figure 4.5.

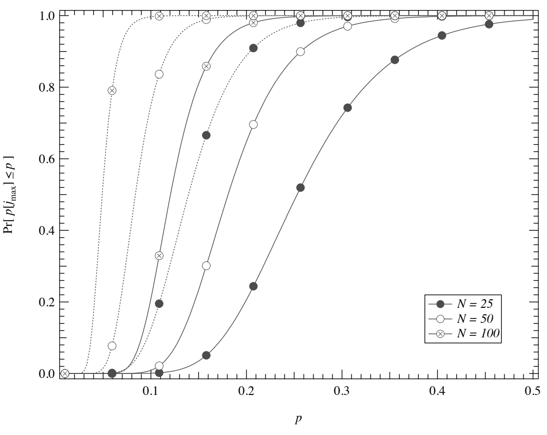


Figure 4.5: Connectivity of $\hat{G}(j,N)$ (solid lines) and $G_p(N)$ (dotted lines) as a function of p, for $N = 25, 50$ and 100. The probability $\Pr\left[\hat{G}(j,N) \text{ is connected}\right]$ is computed with (4.9), where the coefficients are obtained through fitting the results from Figure 4.4. The connectivity of the random graph $\Pr\left[G_p(N) \text{ is connected}\right]$ is computed via (4.10).

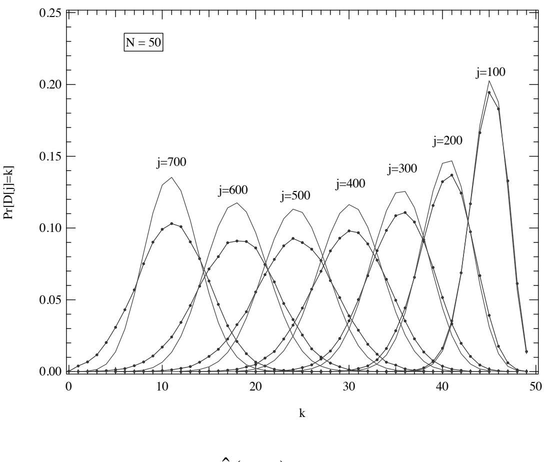


Figure 4.6: Degree distribution of $\hat{G}(j,N)$ for $N = 50$ and various j. The simulation results (dotted lines) are compared with the binomial degree distribution of $G_p(N)$ with $p = \mathrm{E}[L[j]]/L_0$ (plain lines).

4.3 Densely connected networks

In Section 4.2, we have obtained an upper-bound on the number of flows for the case of the fully connected network. Because each graph is a subgraph of the complete graph, the upper-bound applies to all networks. In this section, we demonstrate that we can also apply (4.4) to other densely connected networks and show that the network does not necessarily have to be a full mesh. Dense networks sometimes exhibit a nodal degree distribution which, by approximation, approaches a binomial distribution. In that case, the network may resemble a graph of the class of Erdös-Renyí random graphs and we can apply (4.4) with slight modifications to compute j_{max}. The important differences between the densely connected network and the full mesh are the initial conditions. For the densely connected network, the initial link density is smaller than one, which will yield a smaller j_{max}. We will denote the densely connected network with initial link density p_0 by $\hat{G}_{p_0}(j,N)$. We obtain the link density for $\hat{G}_{p_0}(j,N)$ as,

$$p[j] \leq 2-(2-p_0)\left(1+\frac{1}{L_0}\right)^j \tag{4.11}$$

Similarly, the upper-bound on the number of flows can now be obtained via,

$$j_{max} \leq \frac{\ln(2-p_c+\varepsilon)-\ln(2-p_0)}{\ln\left(1+\frac{1}{L_0}\right)} \approx L_0 \ln\left(\frac{2-\varepsilon-\frac{\ln N}{N}}{2-p_0}\right) \tag{4.12}$$

We have performed simulations for $N = 25, 50$ and $N = 100$ and $p_0 = 0.4, 0.5, \ldots, 1.0$. Figure 4.7 compares the link density from the simulation results with (4.12). Again, the simulation results agree well with the computations.

In Figure 4.8 we have plotted $\mathrm{E}[j_{max}]/L_0$ against p_0. The case where $N \to \infty$ is added to Figure 4.8 and forms the asymptotic upper-bound on the number of flows in the network.

4.4 The fully connected graph with multiple channels per link

In this section we study networks that store multiple flows per link. The capacity is equal for each link and can store up to C_0 flows. This implies that a link is available for allocation if at least one channel is free. If we denote the probability that a link is available by p and assume a weak dependence in the channel usage per link, then we can express $p[j]$ in terms of the "channel density" $q[j]$ as,

$$p[j] = 1-(1-q[j])^{C_0} \tag{4.13}$$

and we may approximate the expected hopcount, similarly to (4.2), as

$$\begin{aligned} \mathrm{E}[H(p[j])] &\geq 2-p[j] \\ &\geq 1+(1-q[j])^{C_0} \end{aligned} \tag{4.14}$$

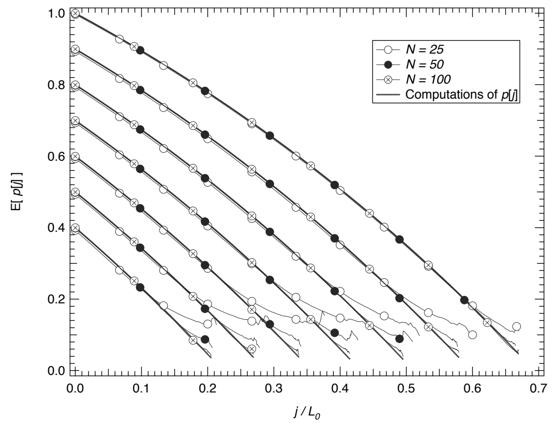


Figure 4.7: The link density in $\hat{G}_{p_0}(j,N)$ for $N = 50$ and $N = 100$ with $p_0 = 0.4, 0.5, \ldots, 1.0$ after allocating j paths. The simulations (lines) are compared with (4.11) (markers). The number of paths on the x-axis is normalized with respect to the number of links in the full-mesh network of corresponding size.

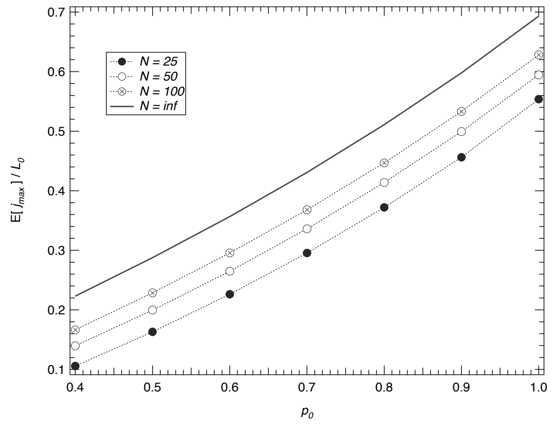


Figure 4.8: The average number of maximally allocatable flows in an Erdos-Renyi graph with $N = 25, 50, 100$ and $p_0 = 0.4, 0.5, \ldots 1.0$. The x-axis shows p_0, while the y-axis shows the average number of flows that can be allocated before disconnection occurs. The lines with markers show the simulation results, while the solid line shows the asymptotic result of (4.12) for $N \to \infty$.

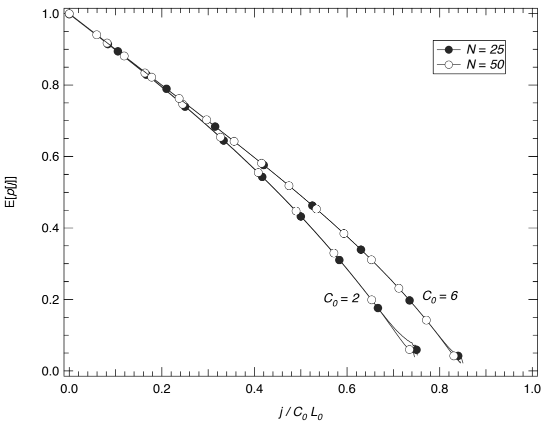


Figure 4.9: Average “link density” in a network where multiple flows can be allocated per link. Networks of $N = 25$ and $N = 50$ nodes are compared, both with $C_0 = 2$ and $C_0 = 6$ channels. The lines with the markers are simulation results, while the plain curves represent computations of (4.15). The number of paths on the x-axis is normalized with respect to the number of channels in the full-mesh network of corresponding size.

The “channel density” now corresponds to the average number of available “channels” $C[j]$ in the network,

$$\begin{aligned} q[j] &= \mathrm{E}[C[j]]/C_0L_0 \\ &= 1 - \sum_j \mathrm{E}[H[j]]/C_0L_0 \end{aligned}$$

The difference equation (4.3) is expressed in the available channels after j allocations,

$$\begin{aligned} \mathrm{E}[C[j+1]] &= C_0L_0q[j] - \mathrm{E}[H(p[j])] \\ &\leq C_0L_0q[j] - \left(1 + (1-q[j])^{C_0}\right) \end{aligned} \tag{4.15}$$

Difference equation (4.15) is non-linear and an analytic solution is hard to obtain. We have performed simulations to examine the upper-bound for these classes of networks and understand the differences with the single-channel case. Figure 4.9 shows the “link density” as a function of the number of flows.

Figure 4.9 shows the simulation results for networks with $N = 25$ and $N = 50$ and capacities $C_0 = 2$ and $C_0 = 6$. An interesting observation from Figure 4.9 is that when a link can carry multiple flows, the curves for the different network sizes still overlap like for $C_0 = 2$. Another observation is that the computations of (4.15) increasingly deviate from the simulations as j increases. The differences are more pronounced for large C_0. One cause is the accuracy of expected hopcount in (4.15). The approximation for the average hopcount in relation (4.14) ignores the dependencies between the

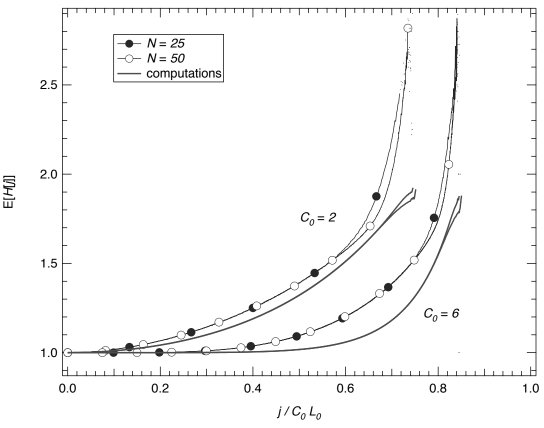


Figure 4.10: Average hopcount in a network where multiple flows can be allocated per link. Networks of $N = 25$ and $N = 50$ nodes are compared, both with $C_0 = 2$ and $C_0 = 6$ channels. The number of paths on the x-axis is normalized with respect to the number of channels in the full-mesh network of corresponding size. The simulations (lines with markers) are compared with computations of the hopcount, where the simulation results from Figure 4.9 have been used for $p[j]$.

channels of a link and assumes that all channels are occupied with same probability. Figure 4.10 shows the simulation results on the average hopcount and compares them with computations of (4.14), where we have used the link density from the simulation results to evaluate (4.14). The computations increasingly underestimate the simulation results with increasing j and C_0. This implies that the probability that the link is not available is higher than (4.13) suggests.

4.5 Rejection rate and disconnectedness

Section 4.2 has focused on connectivity of $\hat{G}(j,N)$ and its relation with the nodal degree. In this section we study the blocking probability of $\hat{G}(j,N)$ in more detail. The blocking probability, or rejection ratio, is closely related to the connectivity of a network, but they are not the same. The difference is that when the network is disconnected, two nodes may still be able to reach each other as long as they belong to the same cluster. The graph $\hat{G}(j,N)$ is defined as the graph that evolves after j random, minimum-hop paths have been removed from the complete graph K_N. In addition to $\hat{G}(j,N)$, we define $\hat{G}(p[j],N)$ as the graph that evolves after an arbitrary number of random, minimum-hop paths have been removed, such that the final link density is, approximately, p. Figure 4.11 presents the meta-code for the construction of $\hat{G}(p[j],N)$. The difference between $\hat{G}(j,N)$ and $\hat{G}(p[j],N)$ is that the latter does not require connectedness.

When a graph is disconnected, it is fragmented into groups of nodes, which are referred to as “clusters”. Clustering in the Erdös-Renyí random graph has been studied in detail by Janson *et al.* [30]. The transition from a group of solitary nodes to a connected cluster is very steep for increasing link density. The majority of the nodes belongs to a single cluster, which is called the “giant component” (GC). The remaining nodes are clustered in groups of size $O(\log N)$. The size of GC (S_{GC}) in the Erdös-Renyí random graph, is given by, (see [45]),

$$S_{GC}(\overline{D}_{G_p(N)}) = 1 - e^{-\overline{D}_{G_p(N)}} \sum_{n=0}^{\infty} \frac{(n+1)^n}{(n+1)!} (\overline{D}_{G_p(N)} e^{-\overline{D}_{G_p(N)}})^n \tag{4.16}$$

where $\overline{D}_{G_p(N)} = p(N-1)$ is the mean nodal degree of $G_p(N)$. For two nodes to be connected in any graph, they must belong to the same cluster. If the nodes are uniformly chosen from a set of N nodes, then the probability that one node belongs to a cluster with size N', equals

$$\Pr[\text{A node belongs to a cluster of } N' \text{ nodes}] = S = \frac{N'}{N} \tag{4.17}$$

The probability that both nodes belong to the same cluster, is then S^2. For N sufficiently large, we can assume that two random nodes in the network are connected if they belong to the GC. The rejection rate then follows as the probability that a source and destination node are not a part of the same cluster,

$$\begin{aligned} r &= 1 - \Pr[\text{Source node in GC}]\Pr[\text{Destination node in GC}] - \varepsilon \\ &= 1 - S_{GC}^2 - \varepsilon \end{aligned} \tag{4.18}$$

where $\varepsilon \geq 0$ is the correction for the probability that the source and destination nodes lie in the same cluster other than the giant component.

4.5.1 Simulations

We have computed the blocking probability for $G_p(N)$ and $\hat{G}(p[j],N)$ for $N = 25$ and $N = 50$ by simulation. For each combination of N and p we have generated $1,000,000$ different realizations of $G_p(N)$ and $\hat{G}(p[j],N)$. After realization of $G_p(N)$ and $\hat{G}(p[j],N)$,

```
GENERATE $\hat{G}(p[j],N)$
1  INITIALIZE($K_N$)
2  **while** $linkdensity[\hat{G}(p[j],N)] > p$
3    **do** node $v \leftarrow$ RANDOM-NODE($\hat{G}(p[j],N)$)
4       node $u \leftarrow$ RANDOM-NODE($\hat{G}(p[j],N)$)
5       path $P \leftarrow$ DIJKSTRA($\hat{G}(p[j],N), v, u$)
6       if $length[P] > 0$
7          remove path $P$ from $\hat{G}(p[j],N)$
```

Figure 4.11: Meta-code of GENERATE for the construction of $\hat{G}(p[j],N)$.

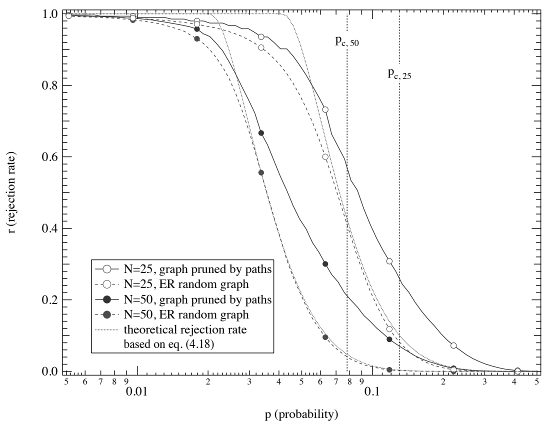


Figure 4.12: Average blocking probability compared with computations of (4.18). The dashed lines are results for the Erdös-Renyí random graph, while the solid lines show the simulation results for $\hat{G}(p[j],N)$. In addition, computations of (4.18) and the density threshold values for the E.-R. random graph have been added.

a pair of nodes is chosen uniformly and an attempt is made to route a path between the two nodes. The ratio of average failed attempts by the total number of attempts gives us the rejection rate,

$$r = \frac{\mathrm{E}\left[\sum \text{failed attempts}\right]}{\sum \text{total attempts}}$$

The rejection rate of $G_p(N)$ and $\hat{G}(p[j],N)$ is presented and compared in Figure 4.12. In addition, computations of (4.18) have been added, where S_{GC} is computed via (4.16).

Figure 4.12 demonstrates that the rejection rate of $\hat{G}(p[j],N)$ is higher than that of $G_p(N)$ for equal values of p. This implies that the probability that two nodes belong to the same cluster is smaller and, therefore, the average cluster size should be smaller. A decrease in average cluster size is established by (i) an increase in the average size of the clusters that separate from the GC, or, (ii) an increased probability of disconnectedness. Especially, the latter observation is in agreement with our findings from Section 4. The computations on the size of the GC with (4.18) are very accurate for $G_p(N)$, except when the link density becomes too small and the GC vanishes. To study the GC and the average cluster size, we have recorded the cluster size distribution for $\hat{G}(p[j],N)$ during the simulations. The results are shown in Figure 4.13. Figure 4.13 illustrates how the size of clusters is affected by the link density. Each curve shows the portion of nodes that belong to a cluster of size s. Several lines have been omitted for clarity. Note that the curve $s = 25$ actually corresponds to the probability that $\hat{G}(p[j],N)$ is connected, see Figure 4.5, while the curve $s = 1$ corresponds to the probability that a node is isolated. We can observe that the breakdown of the GC is initiated by the emergence of an isolated node, around $p \approx 0.3$. Further decreasing the link density results in the emergence of

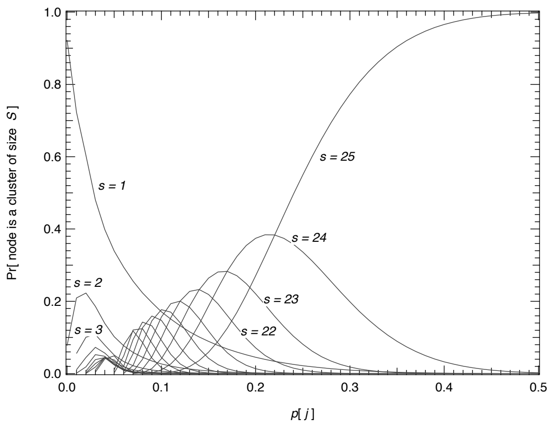


Figure 4.13: The probability that a node belongs to a cluster of size s is plotted against the link density for $\hat{G}(p[j],N)$, where $N = 25$. Several traces for different s have been omitted from the figure to enhance the readability.

more isolated nodes, until small clusters arise consisting of several nodes, indicating the breakdown of the giant component. Summarizing, Figure 4.13 indicates that the network becomes disconnected because solitary nodes separate from the GC. It appears unlikely that the network actually splits up into a set of clusters of comparable size. Even more, the co-existence of small clusters seems unlikely in the presence of the GC. Hence, we argue that the difference between the rejection rates between $\hat{G}(j,N)$ and $G_p(N)$ can be primarily associated with the increased probability of disconnectedness of $\hat{G}(j,N)$.

4.6 Chapter summary

We have studied the number of flows that can be allocated in a network and found the upper bound, given by (4.5), on the number of flows before the network becomes disconnected. In comparison to Chapter 3.1, where we have introduced the term "network capacity" as the number of flows the network can sustain, this chapter only considered the static case where flows are allocated for an infinite duration. The allocation of paths can be considered as a sequence of i.i.d. random variables, such that the maximum number of flows follows a Gumbel distribution [43, pp. 107]. with a mean of approximately $0.7L_0$. Furthermore, relation (4.4) accurately approximates the link density in the network during the allocation process. We have also studied the connectivity of the network as flows are allocated and found that the probability that the network is connected follows the cumulative distribution function of a shifted Weibull distribution. In the case of densely connected networks, where the degree distribution approximates a binomial distribution, the link density and maximum number of flows are found as (4.11) and (4.12), respectively. For other classes of networks, e.g. lattice, Barabási-Albert [13] or power-law graphs, the calculation of the maximum number of flows is limited by the construction of the difference equation (4.3). In particular, we have examined the multiple-channel graph and showed that when links contain more than one channel, dependencies are introduced that result in a non-linear difference equation. However, when the link capacities are increased to carry multiple flows, the multiplexing gain that is obtained by the increased capacity is independent of the network size.

Finally, we have studied the connectivity of $\hat{G}(j,N)$ and compared it with the Erdös-Renyí random graph. The removal of paths from the network results in a wider degree distribution as compared the Erdös-Renyí random graph. The difference is introduced by the correlation that exists between the links of each path. The degree of intermediate nodes in a path, is decreased by two, while the degree of the endpoints is only decreased by one. The nodes are treated differently from each other and the degree distribution increasingly deviates from the binomial distribution of the Erdös-Renyí random graph. The increased variance in the degree distribution manifests itself when considering the rejection rate and connectivity: the transition from no rejection (full connectivity) to full rejection (no connectivity) as a function of the link density is less steep for $G(p[j],N)$ as compared to $G_p(N)$.

Chapter 5

DeSiNe: A flow-level simulator with QoS extensions

5.1 Introduction

The growth in both complexity and size of data communication networks makes the tasks of testing and measuring very complex. Testing networks through realistic test-beds requires many nodes, which is rather costly and involves many practical limitations with respect to e.g. network size, and configuration. Measuring is difficult, since typically one cannot arbitrarily sample the topology or the traffic in large networks like the Internet [4]. Moreover, the "repeatability", or trustworthiness, of test scenarios in real networks is difficult to guarantee. The most accessible method to study network performance and to test new protocols and algorithms is via simulation. In this chapter we present a network simulator that was developed within the Network Architectures and Services group at the TU Delft and is tailored for studying Quality-of-Service in networks at the flow-level. Simulating networks at the flow-level presents several important advantages as compared to the traditional packet-level simulation. Section 5.2 elaborates on the differences between the packet- and flow-level simulation methods. Section 5.3 discusses the notion of Quality-of-Service and the various components of the Quality-of-Service framework that have been implemented in the simulator. The remainder of the chapter presents the functional and architectural design of DeSiNe, followed by several examples.

5.2 Flow-level simulation versus packet-level simulation

Based on the abstraction level at which the traffic is modeled, simulators can be roughly categorized as packet-level and flow-level simulators. A third category, hybrid simulators, combines packets and flows. Packet-level simulation has gained most attention in the past with tools such as the popular open-source ns-2[1] and the SSFNet[2] simulators. In packet-level discrete event simulators, the arrival and departure of each packet triggers an event in the simulator. These simulators aim at an accurate computation with high level of detail and high fidelity. However, this high level of detail comes at the expense of high computation time and memory usage. Packet-level simulators are mainly applicable to small networks and are often used for studies where a high level of detail is required, e.g. when monitoring signaling messages of a new protocol, or the effect of a new buffer management scheme in a router. The major advantage of this approach is that the protocol model does not differ largely from the real protocol implementation. At the same time, this accuracy comes at a cost. The main limitation of packet-level simulators is their lack of abstraction as considerable amounts of computer resources are needed to maintain the complete state of network protocols, especially for large network simulations.

There are two main causes to the lack of scalability of packet-level simulators. Firstly, packet-level simulators rely on discrete-event simulation (DES). The principle of DES consists in keeping all the network protocol events ordered according to their time of occurrence, using a priority queue. Typical events in this context are the transmission of a data unit by a protocol or the expiration of a protocol timer. Additionally, intermediate events might also be created in order to model the CPU workload or queueing delays at various levels of the protocol stack. Traditional DES implementations relied for their operation on priority queues which have a time complexity of $O(log(n))$ to insert new events. The advent of calendar queues in modern schedulers has reduced this complexity to $O(1)$ on average.

Secondly, packet-level simulators lack a level of abstraction in the protocol stack. Each protocol model is extremely accurate, often including the complete protocol finite state machine. This requires significant time and space to hold on the computer. Moreover, heavy traffic conditions increase the computational cost of packet-level simulations. Indeed, each packet-flow is composed of several thousands of packets that must go through the network stack of several nodes, generating a lot of events and increasing the running time of the simulation. One can say that for a single flow, the running time increases linearly with the number of packets in the flow. To circumvent these fundamental problems of packet-level simulation, various approaches have been proposed. Several methods rely on exact solvable models instead of simulation. For example, Anick *et al.* [5] proposed an exact solvable queuing model that describes the queuing behavior at the queue level in their seminal paper. Although their system allows for high accuracy, their hypothesis of independent, continuous on-off sources is too restrictive. The majority of proposed solutions however seek to enhance the scalability and performance of network simulators by lowering the granularity of the simulation or raising the level of abstraction, specifically the level of traffic abstraction.

One approach is to consider the traffic at the level of flows. We regard a flow as a connection established between a source and a destination node in a network. All the packets of that flow follow the same path from the source to the destination during the life-time

[1]http://www.isi.edu/nsnam/ns/

[2]http://www.ssfnet.org/

of that flow. For example, Flowsim [3] uses DES, but coarsens the granularity of the network traffic by aggregating individual packets into packet-trains. The total ensemble of packets then only generates a single event, namely the flow. Hence, the number of events is reduced at the expense of the accuracy of the simulation. Consequently, flow-level simulations are more suitable for scenarios with larger networks and a large number of flows. Moreover, in the context of QoS routing [42, 44], resources should be reserved, i.e. flows set-up, in order to guarantee QoS. Understanding the behavior of flows in a network is invaluable in designing QoS-aware networks.

A second approach, known as the fluid-flow model [35,46], consists in keeping track of the sending rate of sources. This is in contrast with DES which keeps track of any single packet transmission. The fluid-flow model is particularly suitable for studying the impact of congestion on the traffic sending rate, as is the case with TCP. Under light or moderate traffic conditions, the number of events raised in flow-level simulators is typically much less compared to packet-level simulators. However, it has been shown that under heavy traffic conditions and when several flows share the same available resources, one change in the sending rate of a single flow may influence the sending rate of many other flows. This can cause an avalanche of sending rate updates with a dramatic impact on the running time of the simulation. This effect is known as the *"ripple effect"* [35,39], and may lead to drastic performance degradation, such that packet-level simulation will outperform flow-level simulation.

Another scalability improvement can be obtained by switching from a packet-level event-driven simulation to a packet-level time-stepped simulation. In a time-driven fluid simulation, the continuous traffic flow is discretized into time intervals. The time-driven nature relieves the problem of the ripple effect, since the network is only sampled at fixed intervals. This is the approach exposed in [34, 73]. By using coarser time scales it is possible to further speed up packet-level time-stepped simulators. Hybrid packet/fluid-flow simulators use packets for foreground traffic, but model the background traffic at fluid-level. Examples of hybrid simulators have been proposed by Nicol *et al.* [49] and Yung *et al.* [75]. The Hybrid Discrete-Continuous Flow Network Simulator (HDCF-NS) [41] is another example, but little information is given about the actual model and how flows and packets interact. The IP-TN simulator [36] defines hybrid nodes that are capable of mixing both traffic models. The hybrid nodes estimate the aggregate input rate of the packets and the capacity is shared with the flows proportional to the combined input rates.

5.3 Quality-of-Service

Quality-of-Service comprises advanced packet handling that facilitates performance guarantees on an end-to-end connection. QoS-aware routing algorithms do not only consider "static" topology information during path computation, such as the distance in hops, but route the traffic based on the actual traffic conditions, such as the link utilization. Accurate knowledge about the actual state of the network is a crucial factor in QoS and traffic engineering. Each router in the network maintains a table with link-state information advertised by the other routers. Routing algorithms with traffic engineering extensions rely on trustworthy information on the actual available network resources in the path computation process, which is of vital importance to guarantee the QoS requirements of each flow. The link-state information consists of a set of metrics that describe the (transient) state of the link. To keep the link-state information up to date, routers advertise link state updates by flooding messages across the network. Stale link-state information may compromise the dynamic routing process and increase the likeliness of finding a suboptimal route. The acquisition and distribution of network information is a task of the *link-state update policy* (LSUP). As real-time link-state monitoring and link-state advertisements are costly in terms of network resources, a trade-off must be made between the accuracy and overhead associated with the link-state updates. The update policies can be classified into three categories [6, 38]: time-based, trigger-based and window-based. With time-based policies updates on the link-state information are triggered by the actual time. The time-based update policy implemented in DeSiNe is the *periodic update policy*, where updates are sent at fixed time intervals.

Trigger-based policies monitor the real link-state information and send updates when some conditions are met. Two trigger-based policies are implemented: threshold-based and class-based. With threshold-based policies, updates are triggered when the relative difference between the current link state and the advertised link state exceeds a certain threshold. A class-based policy divides the link capacity into a set of classes. When a class boundary is crossed, an update is triggered.

Finally, the window-based policies are used in combination with trigger-based policies and circumvent the problem when the link utilization oscillates around a class boundary or when the traffic is very bursty (threshold-based). Two window-based policies are currently available: hold-down timer and moving-average. The hold-down timer is used to set a minimal spacing between two successive updates. The moving-average takes the average link utilization over a fixed-size time-window and based on this average an update event may be triggered.

The *routing algorithm* utilizes the link-state information provided by the link-state update policy to find the optimal path. The best path is computed according to an optimality criterion. In QoS-aware networks, the path must obey a set of QoS requirements and can thereby include or exclude a certain objective function. QoS routing algorithms solve the Multi-constrained Path Problem (MCP) [44]:

> *Given a network $G(N,L)$ in which each link $u \longrightarrow v \in L$ is specified by a link weight vector with as components m additive QoS link weights $w_i(u \longrightarrow v) \geq 0$ for all $1 \leq i \leq m$. Given m constraints L_i, where $1 \leq i \leq m$, the problem is to find a path $\mathcal{P}$ from a source node A to a destination node B such that, for all $1 \leq i \leq m$, $w_i(P) \overset{def}{=} \sum_{(u \longrightarrow v)\in\mathcal{P}} w_i(u \longrightarrow v) \leq L_i$.*

The MCP problem consists of finding a path that satisfies m constraints. When the path meets all the constraints, it is said to be feasible. The problem of finding the shortest length path, given a definition of path length, among the feasible ones is known as the Multi-constrained Optimal Path Problem (MCOP). Traffic Engineering algorithms can be considered a sub-class of QoS routing algorithms, without multiple constraints. In traffic engineering, the paths assigned to flows try to optimize a local or global objective, i.e. prevent congestion on links or minimize the blocking of future flows. We refer to [42,44] for a detailed discussion on QoS routing and to [11,31,68] for on-line traffic engineering algorithms.

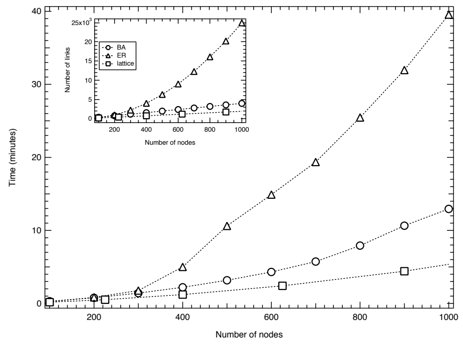
5.4 DeSiNe

In the remainder of this chapter we present DeSiNe. DeSiNe stands for Delft Simulator of Networks and is a scalable flow-level QoS simulator. DeSiNe incorporates QoS routing and traffic engineering algorithms. The purpose of DeSiNe is to study and compare the performance of various QoS routing and traffic engineering implementations at the network level. In particular, DeSiNe supports constraint-based routing and dynamic QoS routing, as well as several on-line traffic engineering algorithms. The good scalability permits the simulation of large networks and heavy-traffic scenarios. DeSiNe models the traffic streams as flows and assumes fixed flow rates and independence between flows, i.e. the arrival or departure of a flow has no effect on the already existing flows. The motivation for taking fixed flow rates is two-fold. Firstly, one of the purposes of DeSiNe is to investigate QoS technology, which implies resource reservations. When the required resources are reserved for a particular flow, they remain unchanged during its lifetime. Secondly, fixed flow rates prevent from running into the scalability problems encountered from variable flow rates arising from feedback-based protocols like TCP.

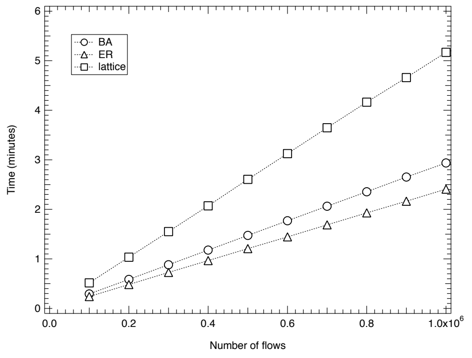
Additionally, the homogeneity and high level of abstraction make DeSiNe easy to use and configure, and suitable for simulation of large networks with many flows. The processing and memory requirements of DeSiNe scale linearly with the number of flows. Figures 5.1a and 5.1b show the CPU time required for simulations with a given number of nodes and flows, respectively. Several types of topologies have been used (Barabási-Albert [13], Erdös–Renyí random graphs [17], and lattices), with varying numbers of nodes, up to 1000. We also vary the number of flows, up to 100,000. We observe that the simulation time scales linearly with the number of flows. The scaling of the simulation time as a function of nodes is not linear because of different link densities of the topologies, as well as different computation times for the Dijkstra algorithm on different types of topologies. Note that unit weights are used on the links.

Moreover, DeSiNe is useful when examining “classes” of networks with specific properties, e.g. Erdös–Renyí random graphs, lattices and scale-free graphs. Such studies require automated generation of many randomized graph realizations. Implementation and incorporation of additional routing protocols, topology generators and link-state update policies is easy and straightforward.

Most simulation studies on QoS routing have created a simulation environment dedicated to the evaluation of their proposed QoS algorithm or policy. When we consider the field of QoS routing algorithms, often only the algorithm is implemented and run on several static networks, without considering traffic and flow dynamics. Several QoS simulators have been published and we will briefly mention them, here. Zhang *et al.* [76]



(a)



(b)

Figure 5.1: Scalability of DeSiNe. (a) Scalability with respect to the network size. (b) Scalability with respect to the number of flows.

have developed the packet-level QoS Routing Simulator (QRS), which was later extended to EQRS [77] to capture DiffServ MPLS networks. A. Shaikh [56] has developed a flow-level event-driven QoS simulator called *routesim*, which focusses on the evaluation of link-state update policies. Sivasankar *et al.* [60] developed a flow-level simulator, called MuSDyR. DeSiNe differs from the previous simulators, because it contains all, instead of only some, of the following features: (a) dedicated to evaluate all aspects of QoS routing, including traffic engineering, (b) flow-level abstraction, for scalability, but also as a natural consequence of QoS routing, (c) many built-in QoS mechanisms, and (d) modular design, such that it can easily be extended with new QoS mechanisms.

The multi-constrained routing algorithms that have been implemented in DeSiNe are: SAMCRA, the self-adaptive multiple constraints routing algorithm [44] and TE-DB [11]. SAMCRA is proposed by Van Mieghem *et al.* [44] and exactly solves the MCOP problem. Banjeree *et al.* [11] proposed the TE-DB algorithm which uses TAMCRA [47], a predecessor of SAMCRA, and the max-flow concept used in MIRA. Various traffic engineering (TE) algorithms have been implemented in DeSiNe: several variations of MIRA [31], New MIRA [68], and SMIRA, the simple minimum interference routing algorithm [26]. MIRA takes into account the information of ingress-egress pairs (S_i, D_i) and weights them by their importance α_i. To minimize the interference between source-destination pairs, these algorithms maximize the sum of the residual weighted max-flows[3] between all ingress-egress pairs. Upon the arrival of a connection request from S_i to D_i, the algorithms compute the sets of *critical* links L_j for all other pairs $(S_j, D_j)(j \neq i)$. A link l is called *critical* for an ingress-egress pair (S, D), if the reduction in l's capacity leads to the reduction in (S, D)'s max-flow. Then link costs are set according to the link *criticality*, and the shortest path is applied on this "properly" weighted topology. Different weighting methods lead to variations of MIRA. The resulting path is called the *minimum interference path*. For further details about MIRA we refer to [31]. The New MIRA and SMIRA algorithms seek to improve MIRA in computation complexity or weighting methods. For further details about New MIRA and SMIRA, we refer to [68] and [26] respectively. The above routing algorithms have all been evaluated in [8].

5.4.1 Network model

A network is modeled as a graph $G(N, L)$, where N denotes the number of nodes and L the number of links connecting the nodes. A subset of the nodes can serve as source or destination nodes, while other nodes only act as internal nodes or intermediate hops. Each link is assigned a positive value: the *maximum capacity*, which expresses the maximum amount of traffic the link can carry. The actual utilization of the link is maintained by two values. The first value always contains the true *available capacity* of the link. The second value is the *reservable capacity* that is advertised by the routers and used in the routing process. The link-state update policy synchronizes these values. In case the available and reservable capacity are not synchronized, the link-state information is inaccurate or *stale*.

Each link is also associated with a number of QoS weights related to additive QoS measures (e.g., delay and jitter). Additive measures are such that their value along a path is the sum of the values associated with each constituting link [42, Chap. 12]. Multiplicative QoS measures such as the probability of packet loss can be dealt with by taking the logarithm. The QoS link weights are used by QoS routing algorithms to compute feasible paths, as described in Section 5.4. We allow for static and dynamic link weights. In case of dynamic link weights, the weight can be a function of the throughput of the link. All links are either directed, allowing traffic in a single direction, or undirected, in which case the traffic flows in both directions.

The traffic in the network is modeled by flows. A flow f represents a packet-stream flowing from one source to one destination. To each flow, a set of QoS requirements can be assigned that must be met and guaranteed by the network during the life-time of the

[3]The max-flow for an ingress-egress pair (S, D) is the maximum amount of traffic that can be pushed between this pair. Multiple link-disjoint paths may be used.

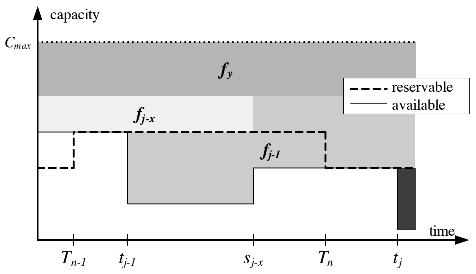


Figure 5.2: Example of the link state, showing the evolution of the available and reservable capacity subject to flow-arrivals and departures. Each colored block corresponds to one flow. First, the advertised link-state value is updated at T_{n-1}. Subsequently, flow f_{j-1} arrives at t_{j-1}. Next, flow f_{j-x} expires at time s_{j-x}. Then, the next periodical update takes place at T_n. Finally, flow f_j arrives at t_j. Before flow f_j is processed, the departure at s_{j-x} and the link-state update at T_n are processed.

flow. The QoS requirements may consist of a single metric, e.g. capacity, or a set of metrics, e.g. delay, jitter, packet loss, etc. When the flow is set up, the required capacity along the path is reserved exclusively to the flow. The resources remain reserved during the full life-time of the flow to guarantee the QoS requirements.

5.4.2 Event triggering

DeSiNe is triggered by the arrival of a flow. When a flow f_j arrives at the network at time t_j, all events in the network that have occurred after the arrival of flow f_{j-1} and prior to t_j are evaluated and processed at the epoch of the new arrival. Figure 5.2 schematically illustrates the arrival of flow f_j at time t_j. The last arrival was at t_{j-1}. All events between t_{j-1} and t_j, such as flow departures and link-state updates, must be processed before flow f_j is routed and allocated in the network. Figure 5.2, provides an example where the link state is displayed schematically from time t_{j-1} to t_j. Between t_{j-1} and t_j one flow departs at time s_{j-x} and the value of the reservable capacity, i.e. the capacity that is advertised by the routers, is updated at T_n.

Figure 5.3 depicts the flowchart diagram of DeSiNe. The simulation begins with the initialization of the network. The network topology can be generated, e.g. random graph, or read from a file. After the network is created, the simulation enters a loop that is repeated for each flow arrival, until the last flow has been generated. The loop begins in step 1 with the generation of a flow. The distribution of each property, such as the flow duration or inter-arrival time, is configured at the start of the simulation. The source and destination nodes are chosen uniformly from the set of valid endpoints. In step 2 the network is scanned for expired flows and time-based (see Section 5.3) link-state updates. If a time-based link-state update policy is used, then the reservable capacity is updated with the available capacity value, as illustrated in Figure 5.2. Next, the flow is routed in step 3. If no link-state update policy is used, and all link-state values are

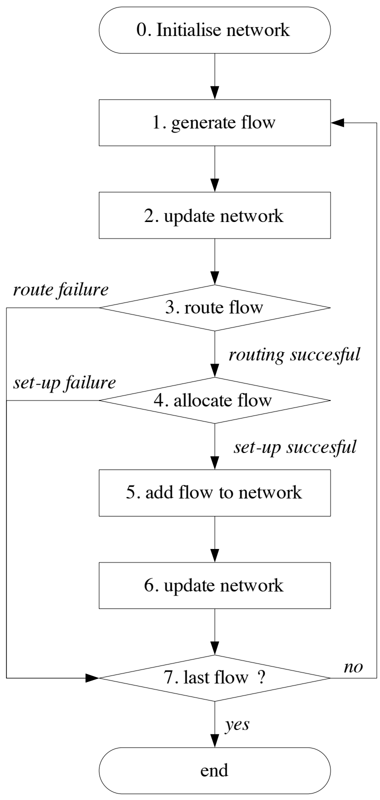


Figure 5.3: Functional view of the DeSine.

updated instantaneously, the reservable capacity equals the actual capacity. The links with insufficient capacity are pruned from the topology. The routing algorithms described hereafter may assign a cost to each link and select the shortest length path, the length of a path being the sum of the costs of the constituting links. Link costs may be specified as a function of the capacity and of the QoS weights. Several different functions are predefined in DeSiNe. If the routing is successful and a feasible path is found, the resources are reserved along the path (step 4). In case the routing fails, or when the routing is successful, but the reservation fails, the flow is rejected by the network and discarded. The reservation may fail due to stale link-state information, such that the advertised link capacity is more than the actual available capacity. If the reservation is successful, the flow is allocated in the network in step 5. The advertised link weights are consequently updated in step 6. Finally, in step 7, the program returns to step 2 for the next flow arrival. If no new arrivals are queued, the simulation ends.

5.4.3 Architecture

To enhance the extensibility of DeSiNe, a modular approach has been used. The modules are grouped based on their function and new modules can be added and removed independently of each other. The modules can be classified according to Figure 5.4, where we have omitted a few utility modules. The module *Main* controls the program flow and performs the data collection, which is written to disk with use of the *IO* (Input/Output) module. Based on the parameters, the *Network* is initialized, holding the actual topology with the traffic information. If the topology is stored on disk, the *IO* module is used to read the corresponding file. The built-in topology generator in *Network* produces (randomized) topologies on demand. Based on a topology class and a matching set of parameters, custom randomized topologies are constructed. In addition to the topological structure, the topology generator sets the link properties based on a specified distribution. The topology representation is implemented as an adjacency list. The *Network* module provides complete control over the network, including functionality to browse the topology and access and manage the individual elements. Since the network traffic is modeled at the flow level, we do not consider nodal properties as queueing or processing delay. Therefore, all nodes are identical and considered propertyless. The properties of the links, e.g. the capacity and QoS-metrics, are determined at generation time and remain constant during the lifetime of the link. The *Network* module uses the *LinkStateUpdate* and *LinkCostFunction* to perform link-state advertisements and compute actual link cost, respectively. A single link-state update policy can be used at a time in the network. In absence of a link-state update policy, the reservable capacity equals the available capacity. The *LinkCostFunction* module computes the weight of a link through a custom function. In general, the weight is a function of the reservable bandwidth. However, some functions may use other measures, such as delay or hopcount. The weight is computed on demand during the routing process. A single link cost function can be active at a time in the network. If the link cost function is not set, the hopcount is used, such that all links have unit weight.

The *Network* module maintains a list containing the active flows, sorted on their departure time. When a flow is requested, a random source and destination node are requested from the topology. Then the arrival time and departure time are determined, based on the inter-arrival time distribution and flow-duration distribution, respectively. Finally, the requested capacity and QoS-constraints are determined. After successful routing, the flow is allocated in the network. The *Network* component is unaware of the concept of flows: it only sees a reduction in capacity as flows are allocated. If the allocation is successful, the flow is inserted into the list. When an update takes place, the "expired" flows are released from the network and removed from the list.

The routing is performed by the *Algorithms* module. DeSiNe uses source routing, for which a full view of the topology is required. The routing algorithms do not check whether the path is feasible: if the link-state information is stale and the path that was found by the routing process is not feasible, then a setup error will occur during the *allocation* of the flow.

The module *Random* implements random-number generator including several well-

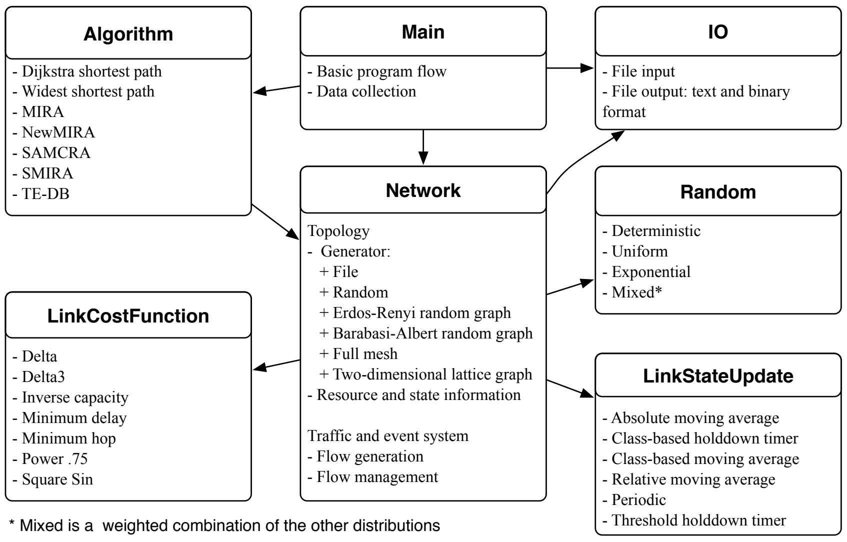


Figure 5.4: The modular design of DeSine. For each module, its core functionality and features are summarized.

known distributions. The underlying pseudo-random number generator is based on the ran4 generator [54]. The *Random* module is used in generating random topologies and in the generation of flows.

5.5 Applications of DeSiNe

DeSiNe is able to simulate the performance of various routing algorithms and link-state update policies under different network and traffic scenarios. It can be used to evaluate new routing algorithms or link-state update policies by comparing them with the existing ones under the same network and traffic conditions. Due to its modular design, custom routing algorithms and link-state update policies can be easily added to the simulator and evaluated thereafter. Network and traffic, under which routing algorithms or link-state update policies will be evaluated, can be configured in flexible ways. Users can define their own performance metrics, e.g. the maximum link utilization, the sum of max-flows, etc., and set them as output.

In this section, we illustrate the use of the simulator on different kinds of scenarios with selected routing algorithms, link-state update policies, network and traffic scenarios, and performance metrics, to show some of the features of DeSiNe. In Section 5.5.1, we compare the performance of different link-state update policies and routing algorithms, where the flows come and leave according to certain processes. We use a number of flows to reach a steady state, and then take statistical results. In Section 5.5.2, we show the performance of different traffic engineering algorithms. Moreover, we study the performance of classes of networks by using randomized samples of such a class. The results are presented in Section 5.5.3, where we study the Erdös-Renyí and Barabási-Albert classes of networks.

5.5.1 Link state updates

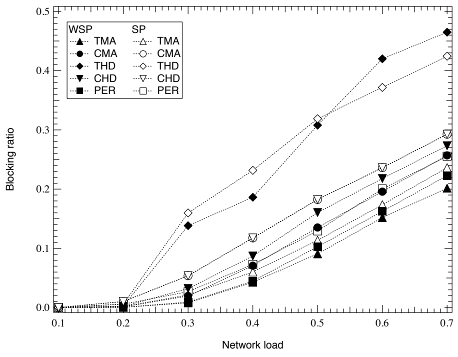
We compare, for a given topology, the performance of different link-state update policies and routing algorithms. We use the MCI topology. The MCI topology consists of 19 nodes, out of which 11 nodes are edge nodes and the remaining 8 are core nodes, connected by 33 links. Each link is bidirectional and assigned 600MB/s capacity. A bidirectional link acts as two unidirectional links both with the assigned capacity. The setting of the scenario is the same as in [57].

The arrival process of the incoming traffic flows is modeled as a Poisson process with rate λ flows per second. Source-destination pairs are uniformly selected among the set of edge nodes. The service time of flows, i.e. the flow duration, is described by an exponentially distributed random variable d with mean 10 seconds. We denote by C_r the capacity requirement of each flow, which is uniformly distributed within $[15,45]$ MB/s. Following [57], the network load is defined as:

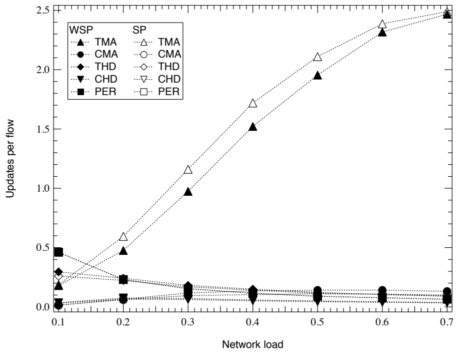
$$\rho = \lambda \mathrm{E}[D]\,\mathrm{E}[C_r]\,\mathrm{E}[H]/L,$$

where $\mathrm{E}[D]$ is the mean flow duration, $\mathrm{E}[C_r]$ is the mean capacity requirement, $\mathrm{E}[H]$ is the mean hop count of the shortest paths between all pairs of source and destination nodes, and L is the number of links in the network.

The comparison of several LSUPs and routing algorithms is given in Figures 5.5a and 5.5b. Dijkstra's shortest path and widest shortest path [23,42] routing algorithms are selected both with the Min-Hop link weight function. For each LSUP, we choose one set of parameters for the simulations. For example, for the threshold based moving average LSUP, we set the threshold to be 0.2 and the window size to be 5. Figure 5.5a shows how each combination of LSUP and routing algorithm behaves as a function of the network load. Most of the evaluated LSUPs give a better behavior under the widest shortest path routing algorithm than under the shortest path routing algorithm. Widest shortest path is known to be better in load-balancing traffic across the network than shortest-path routing. The updates per flow shown in Figure 5.5b estimate the protocol overhead. The moving average LSUPs adapts to the traffic density. As the network load increases, the update rate for the moving average LSUPs increases as well. The time sensitive LSUPs set limitations on the period between two updates. As we only tune λ in order to get different network loads, the higher the network load, the smaller the flow arrival interval. The periodical LSUP generates less updates with a higher network load, because under a higher network load there are more flows in a period. The other time sensitive LSUPs take both the time and trigger conditions into account, and give smooth curves for the update rate.



(a) Performance measured as the ratio of blocked flow requests with respect to the total requests.



(b) The average number link updates due to the arrival and departure of one flow.

Figure 5.5: Simulation results for the MCI topology under varying network load, where the shortest-path (SP) and widest-shortest-path (WSP) routing algorithms are used in conjunction with the threshold-based moving average (TMA), class-based moving average (CMA), threshold-based hold-down timer (THD), class-based hold-down timer (CHD) and periodical (PER) link-state update policies.

5.5.2 Routing with traffic engineering extensions

The scenario in this section illustrates the performance of different routing algorithms with traffic engineering extensions. We implemented the scenario of [31]. The network consists of 15 nodes and 28 bidirectional links. The capacity of the core links is 4800 units, and that of the other links 1200 units. Traffic transits between 4 pairs of ingress-egress nodes. Flows with their capacity requirements following the same uniform distribution between 1 unit and 3 units, arrive in a Poissonian manner with the same mean rate for all these 4 pairs. Once a flow is routed and setup successfully in the network, it will never leave the network. We computed after each request the residual max-flow of each

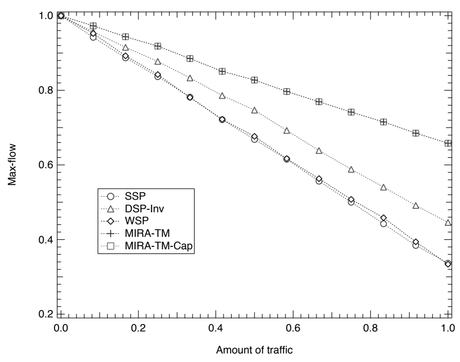


Figure 5.6: Residual max-flow for pair (S_1, D_1)

ingress-egress pair. Figure 5.6 shows the results of 1 of the 4 ingress-egress pairs used in [31], (S_1, D_1) in their topology. We have selected the following routing algorithms in Figure 5.6: static shortest path (SSP), widest shortest path (WSP), dynamic shortest path with link weights being the inverse of residual link capacity (DSP-Inv), and two variants of MIRA (MIRA-TM and MIRA-TM-Cap). MIRA-TM considers in its weighting the traffic between ingress-egress pairs. As all the ingress-egress pairs have the same amount of traffic, MIRA-TM in this scenario is identical to SMIRA used in [31]. MIRA-TM-Cap takes into account not only the traffic between ingress-egress pairs, but also the residual capacity of the critical links.

The x-axis of Figure 5.6 gives the sum of the connection requests that have been successfully routed in the network so far (total traffic accepted). We scaled this sum to quantify the traffic demands. The y-axis gives the normalized residual max-flow for (S_1, D_1). We observe on Figure 5.6 that the two MIRA variants perform better than the other routing algorithms, because MIRA aims at maximizing the residual max-flow between any pair of source-destination pairs. SSP and WSP perform as badly. DSP-Inv that computes dynamic shortest paths depending on the inverse of the residual capacity performs better than both SSP and WSP. The two MIRA variants perform best as they are designed to maximize the residual max-flow on all source-destination pairs. We also observe that the two MIRA variants lead to identical residual max-flow, as in this case the min-cut of this pair does not allow to load-balance traffic for pair (S_1, D_1) without impeding on the max-flow of other source-destination pairs. The ability of traffic engineering traffic differs among the source-destination pairs. Sometimes, alternate paths that do not interfere with other pairs do not exist in the network. Then, traffic engineering cannot help much in load-balancing traffic between specific source-destination pairs since it will impede on other source-destination pairs in the network. Note that the results we obtain are consistent with those from [31].

5.5.3 Random networks

Online generation of random networks is useful when comparing the performance of a particular routing algorithm or link-state update strategy between different classes of networks. Measures of interest here can be the average hopcount, flow blocking or number of updates. DeSiNe features several built-in topology generators (see Figure 5.4). Tables 5.1 and 5.2 show the results of several example scenarios using Erdös–Renyí and Barabási-Albert graphs. Table 5.1 presents the results when using periodic link-state updates. As expected, the number of updates grows with the network size. Furthermore, Table 5.1 shows that Barabási-Albert graphs cannot sustain as much traffic as Erdös–Renyí graphs, for a comparable link density. Blocked flows are those that cannot be routed due to a lack of resources in the network. The rejected flows are the flows for which routing is successful, but the setup fails due to stale link-state information. In Table 5.2 the same networks are used, but now the time-based moving average link-state update policy is used. Clearly, the number of updates is drastically decreased while throughput is improved at the same time.

	N	$\mathrm{E}[H]$	$\mathrm{E}[F_{acc}]$	$\mathrm{E}[F_{bl}]$	$\mathrm{E}[F_{rej}]$	$\mathrm{E}[u]/F_{tot}$
ER	100	3.77	391621	551	7828	120
	200	4.16	396352	189	3459	240
	300	4.37	397917	86	1996	363
	400	4.55	398554	64	1381	482
	500	4.68	398942	38	1019	601
BA	100	3.61	391164	51	8784	117
	200	3.89	395601	5	4393	237
	300	4.04	397035	1	2963	357
	400	4.16	397795	0	2204	478
	500	4.23	398188	0	1811	498

Table 5.1: Simulation results for Erdös-Renyí (*ER*) and Barabási-Albert (*BA*) random graphs with varying N nodes, and comparable link densities; for *BA* $m = m_0 = 3$ and for *ER* $p = \frac{2m}{N}$. The results are the averages computed over 100 random graph realizations and $F_{tot} = 400.000$ flow arrivals. The widest-shortest path routing algorithm with periodic LSUP has been used with window size $w = 50$. In the table, H is the hopcount, F_{acc}, F_{bl}, F_{rej} the accepted, blocked and rejected number of flows, respectively, and u is the number of link-state updates.

5.6 Chapter summary

This chapter discusses several aspects of network simulation. In particular we have focussed on the traffic granularity, i.e. packet- versus flow-level simulation, and on the Quality-of-Service framework, specifically multi-constrained routing and traffic engineering. Furthermore, we have presented DeSiNe, a modular flow-level network simu-

	N	$\mathrm{E}[H]$	$\mathrm{E}[F_{acc}]$	$\mathrm{E}[F_{bl}]$	$\mathrm{E}[F_{rej}]$	$\mathrm{E}[u]/F_{tot}$
	100	3.76	399438	369	192	9.7
	200	4.16	399837	98	65	10.6
ER	300	4.39	399915	47	36	11.1
	400	4.54	399946	30	23	11.4
	500	4.67	399963	20	16	11.7
	100	3.11	399972	11	17	6.85
	200	3.36	399992	2	6	7.43
BA	300	3.49	399996	0	4	7.80
	400	3.58	399997	0	3	8.06
	500	3.65	399997	0	3	8.28

Table 5.2: Simulation results for Erdös-Renyí (*ER*) and Barabási-Albert (*BA*) random graphs with varying N nodes, and comparable link densities; for *BA* $m = m_0 = 3$ and for *ER* $p = \frac{2m}{N}$. The results are the averages computed over 100 random graph realizations and $F_{tot} = 400.000$ flow arrivals. The widest-shortest path routing algorithm has been used with time-based moving average LSUP with window size $w = 50$ and threshold 0.1. The symbols correspond to those of Table 5.1.

lator. DeSiNe is aimed at performance analysis and benchmarking of QoS routing algorithms and traffic engineering extensions. Compared to existing simulators, DeSiNe incorporates in a unique fashion Quality-of-Service routing, traffic engineering and scalability. It provides an end-to-end solution from topology generation to simulation and data collection. This makes DeSiNe well-suited for the performance analysis at system-level of new QoS routing and traffic engineering algorithms in any network topology. DeSiNe has been written in C++ and is modularly built, with extensibility as a major design goal. New algorithms, protocols or other techniques can easily be incorporated into the existing simulator. Moreover, DeSiNe is not built on top of any simulation framework or libraries, which minimizes undefined behavior and dependency issues. The source code is publicly available at the Networking, Architectures and Services group website[4] under the section *Research*.

[4]http://www.nas.ewi.tudelft.nl/

Chapter 6

Conclusions

The Internet has evolved from a simple research network between a handful of hosts towards a vast, worldwide network serving and connecting billions of users. The Internet has become indispensable in modern societies and an increasing number of markets and sectors of society rely on the well-functioning of the Internet. The Internet has altered the art of communication, providing a cheap, fast and global connectivity. It serves as an unlimited source of information and is rapidly taking over the classic telecommunication services of telephony and television. In addition, the Internet offers tremendous opportunities for new markets and services that were not possible or existent before, such as online gaming, telemedicine and e-government. However, the nature of these services often conflicts with the art at which the Internet transports the messages. These services demand stringent requirements on the performance of the network, such that latencies, loss or insufficient capacity can severely deteriorate the quality of the service. Where the conventional telephony system was able to provide service guarantees towards customers, the Internet can only offer a best-effort guarantee. Quality-of-Service provides a framework and a set of solutions to establish service guarantees and reliable communications.

In this thesis we have focussed on flow-based communications. because we believe that when Quality-of-Service will be implemented, the solution will be based on a flow-based approach. In a connection-less environment, the packets can literally travel across the entire Internet. As a consequence, it is very hard to predict where the packets may travel, yielding tremendous uncertainties regarding packet-delays and loss. When packets belonging to the same stream are forced to travel along the same path, the connection is easier to monitor and control, yielding less unpredictable behavior and possibly a better performance.

6.1 Measurement

In Chapter 2 we have studied an extensive data set with traceroute measurements for a period of five years. In particular, we have focussed on the lifetime of routes between a fixed set of source-destination pairs and on the diversity of the different routes between these pairs as time progresses. The measurements indicate that

the average number of unique routes that is discovered over time grows constant in time and a new IP route is found approximately every 15 *days.*

Another important finding in Chapter 2 is the power-law behavior that is found for the lifetime distribution of routes. The lifetime is the time between the first observation of a particular route and the last observation of that route, without seeing another route in between. The power-law exponent indicates that the lifetime distribution has no first or higher order moments, which implies that the lifetime of the route is highly unpredictable and has no expected value. In other words, once a route is established, it can change at any moment in time. A closer inspection of the lifetime distribution revealed that a possible explanation for the unpredictable behavior can be Self-Organizing Critical (SOC, see Section 2.2) behavior in the routing plane of the Internet. SOC is characterized by power-law distributions and $1/f$ noise, both of which are found for the route lifetime. Based on our empirical findings, we argue that the routing dynamics are the source of this behavior. In particular, at the Intra-domain level, where BGP is the de facto routing protocol, routing dynamics are often found and well documented in literature. Route updates can cause long-term dynamic effects, lasting up to many minutes and affecting many systems.

The distribution of the lifetime of Internet paths appears to follow a power-law, yielding unpredictable behavior. This behavior can be a manifestation of self-organized criticality in the routing plane.

Based on the measurements presented in Chapter 2, we argue that the best-effort, connection-less environment that the Internet offers, may hamper the development of real-time applications. The uncertainties in the routing plane can cause dispersion of packets and introduce intolerable latencies and possibly loss. The results serve as a strong motivation for connection-oriented communication for real-time applications and provide a justification to continue research in the field of Quality-of-Service.

6.2 Modeling

In Chapter 3 we have studied the performance of networks where traffic is modeled at the flow level and we defined the network capacity K as the number of flows that the network can sustain. A flow represents a connection between a source and destination that consumes one unit capacity of the links along the path. The path is considered fixed during the lifetime of the flow and in the simplest case, each link has one unit capacity, such that the allocation one flow consumes the full capacity of the link. The network performance is modeled using an analogy with queueing theory, where the network capacity can be considered as the equivalent of the buffer size of a queue. Unlike queueing systems, where the buffer size is a given parameter, the capacity is an emergent property that is the outcome of many network properties and the interaction of processes during the network operation, e.g. the routing process, the selection of source–destination pairs and of course topological properties. In Chapter 3 we have computed the network capacity for several classes of networks and we have found a remarkable relation between the network capacity and the number of links in the network.

The network capacity K scales linearly with the number of links L for several classes of networks, with a maximum of 42 *percent in the case of the fully connected graph:*

$$K = \beta L \qquad , \beta \leq 0.42$$

The scaling factor can be regarded as the efficiency at which the network is used; it presents an upper-bound on the network usage under the model's assumptions. The underlying principle that causes the linear relation and magnitude of the scaling factor is at present an open problem. The primary obstacles are the dependencies that are introduced by the allocation of *flows* instead of *links*: since each flow may span multiple links, and links must be connected to form a path. The allocation of a flow adds dependency between the links. These dependencies make the analysis prohibitively complex.

A rigorous explanation for the dynamic network behavior is still an open issue, however we have found tractable models that give an upper- and a lower-bound on the network capacity. The lower bound is found by assuming that flows cannot span more than one link. Hence the arrival of a flow consists of the allocation of a random link from the network. Blocking occurs when the new arrival tries to allocate an already occupied link. Due to the fixed length of the paths, the estimated capacity is significantly lower, because, if the direct link is not available there may still exist another path of two or more hops. Hence, blocking occurs more often in this model, yielding a lower network capacity. In the case of the the upper-bound, the model assumes an empty network as the initial condition and infinite flow durations. The average hopcount computed over *all* the flows allocated in the network is significantly lower than in the case of a dynamic network, where flows arrive and depart, because the first allocated flows find the network nearly empty. We have studied the dependency between the scaling factor and the number of channels per link and the link density. In the cases of the class of Erdös-Renyí random graphs and the full-mesh graphs with multiple channels per link, the network capacity can be written as presented in Table 3.1.

We have found that the network capacity is independent of the traffic intensity and demonstrated that when the service process is changed from exponentially distributed service times to deterministic servicing, the capacity remains unchanged.

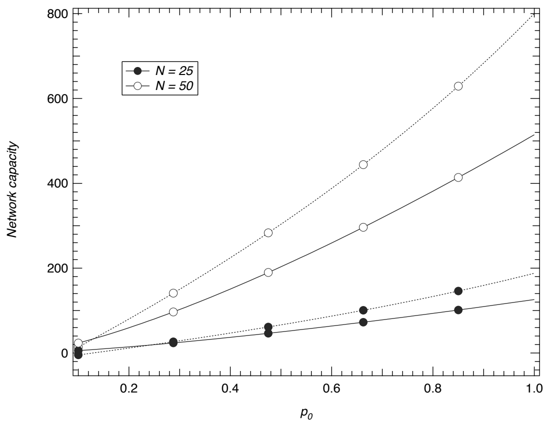


Figure 6.1: Comparison of the network capacity in the dynamic case (Chapter 3, solid lines) and the static case (Chapter 4, dotted lines). The network capacity is computed for the case of the random graph with $N = 25$ and $N = 50$ nodes, the link probability p_0 is presented on the x-axis.

Figure 3.15 illustrates the presence of strong fluctuations, which underlines the notion that dynamics play an important role and the “average” behavior does not sufficiently describe the system.

In Chapter 4 we have investigated an alternate definition for network capacity. Here, we start from the empty network and measure the number of flows the network can support until the network becomes disconnected. We have found an expression that accurately describes the state of the network in terms of available resources as a function of the number of allocated flows at that time. We find that the network capacity is approximately 70 percent of the number of links. The difference between the network capacity from Chapter 3 and Chapter 4 is illustrated in Figure 6.1.

In the static case where flows are allocated for infinite duration, which can be considered as an upper-bound on the dynamic case, the network capacity is approximately 70 *percent of the links.*

The difference between the two scenarios can be explained as follows. In Chapter 3 we consider a dynamic environment where flows arrive and depart from the network, while in Chapter 4 the flows are static and do not leave the network. The scenarios also have a common element, which is that the warm-up phase of the dynamic network scenario is precisely the same as the static scenario. The difference arises as soon as the first flow leaves the network in the dynamic case. The first flows in the dynamic network will see a practically empty network and the hopcount of these flows will be very close to 1, as is discussed in Chapter 4. As these flows leave the network, they release only a single link. The flows that arrive at that time, will find the network occupied and the direct link between the source and destination is likely to be allocated by a previous flow. As a result, the new flow will exhibit an increased hopcount as compared to the earlier flows, which eventually will decrease the network capacity. Hence, the results on the network capacity in the static case can be considered as an upper-bound on the dynamic network scenario.

6.3 Simulation

Chapter 5 presents a new simulation tool, called DeSiNe, that uses flows to model network traffic. DeSiNe features various QoS mechanisms that allow simulation of next generation networks and asses the effect of the difference QoS mechanisms on the network performance. The flow-level approach significantly reduces the computation times, enabling the simulation of large networks and heavy traffic conditions. DeSiNe comprises several built-in functions, such as a topology generation, random number generator and a selection of the well-known (QoS) routing algorithms and update strategies. DeSiNe is particularly suited to test algorithms and protocols on randomized networks of different classes, such that insight can be obtained in the general performance of the algorithm or protocol and not just for one particular network. To illustrate the use of DeSiNe, we have studied several scenarios where we have compared the performance of two network classes using similar routing and update strategies.

Appendix A

Snapshots of a 10-node network

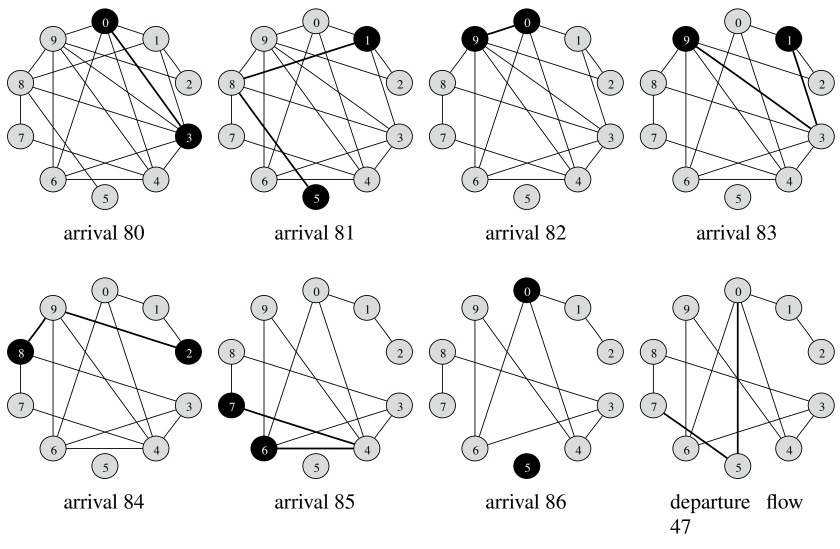
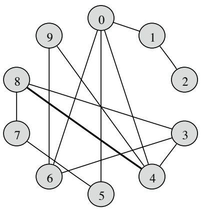
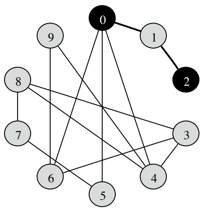
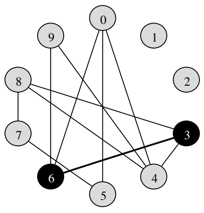
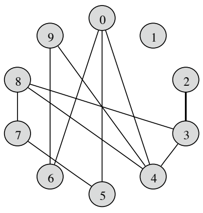


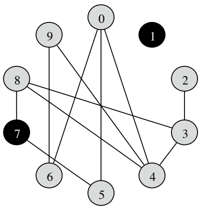
Figure A.1: The arrivals and departures of flows in a 10-node network with a single channel per link. Each diagram depicts a single arrival or departure. The diagrams are printed in chronological order, starting from the 80^{th} flow arrival. For the flow arrivals, the source and destination nodes are colored black. The links corresponding to the path of each flow arrival are highlighted in bold. In the case of rejection, none of the links are highlighted. When a flow is released, the released links are highlighted but not the nodes. — *Figure continues on next page.*

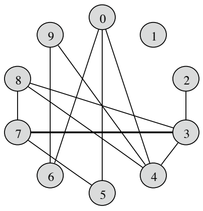

departure flow 62

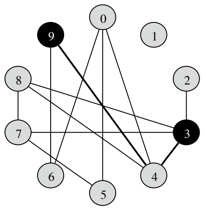

arrival 87

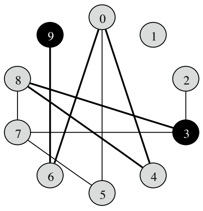

arrival 88

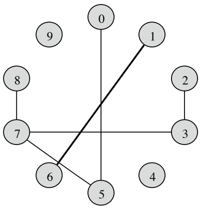

departure flow 61

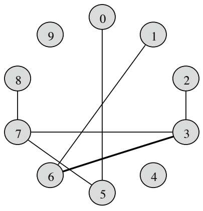

arrival 89

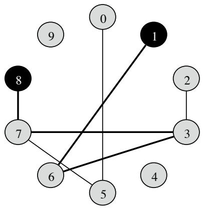

departure flow 72

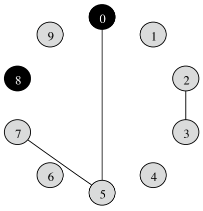

arrival 90

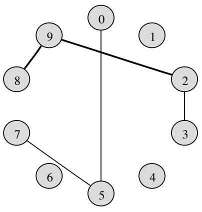

arrival 91

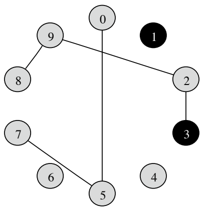

departure flow 68

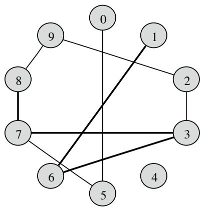

departure flow 88

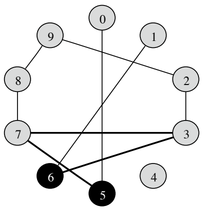

arrival 92

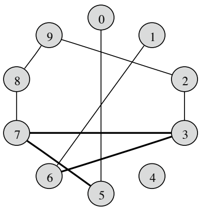

arrival 93

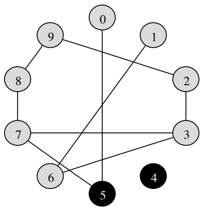

departure flow 84

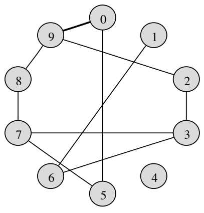

arrival 94


departure flow 92


arrival 95


departure flow 95


arrival 96


departure flow 82

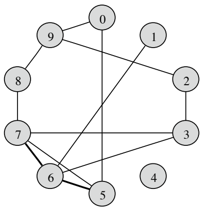
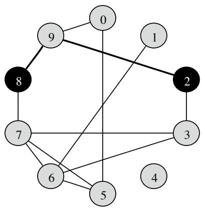
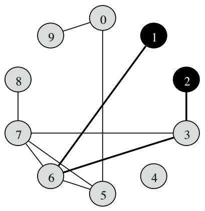
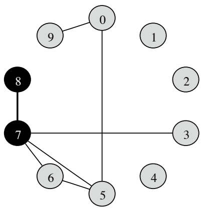
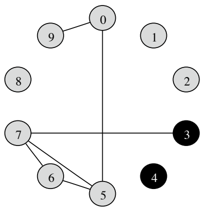

departure flow 60

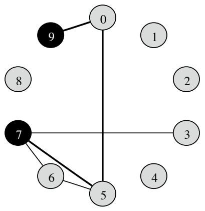
Figure A.1: *– continued –*

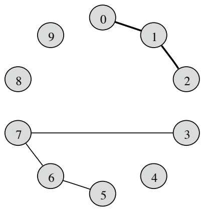

arrival 97

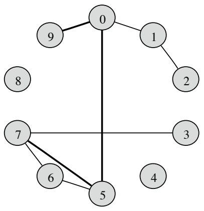

arrival 98

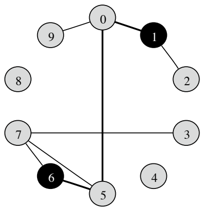

arrival 99

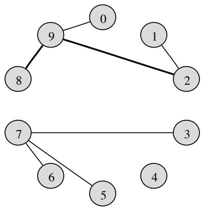

arrival 100

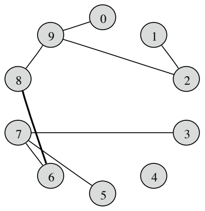

arrival 101

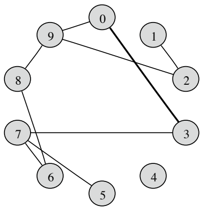

departure flow 87

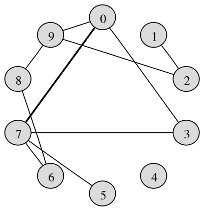

departure flow 101

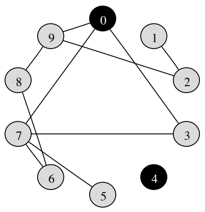

arrival 102

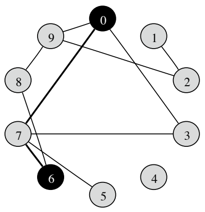

departure flow 97

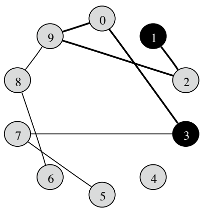

departure flow 77

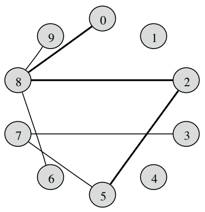

departure flow 80

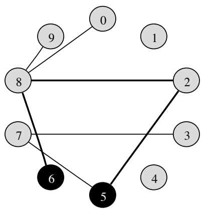

departure flow 65

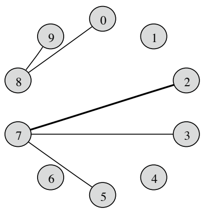

arrival 103

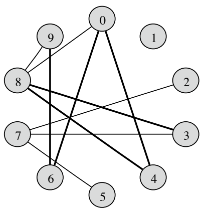

arrival 104


arrival 105


departure flow 63


arrival 106


departure flow 79


departure flow 91

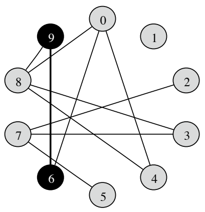
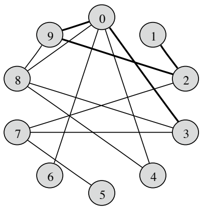
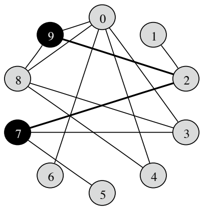
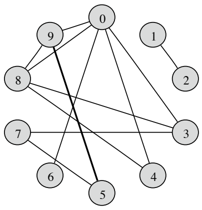
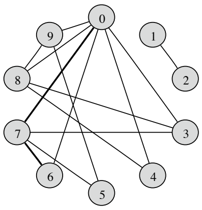

arrival 107

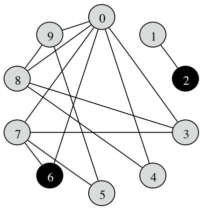
Figure A.1: *– continued –*

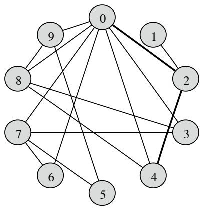

departure flow 105

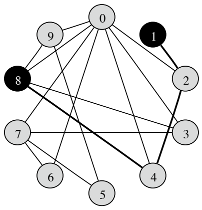

arrival 108

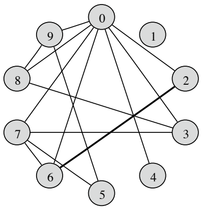

departure flow 70

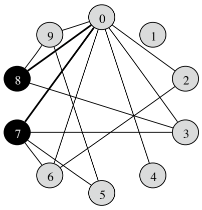

departure flow 104

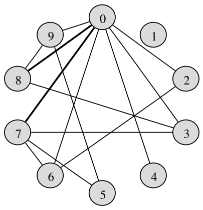

arrival 109

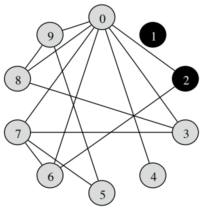

departure flow 38

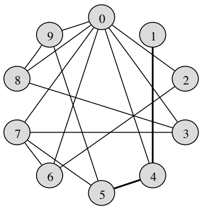

arrival 110

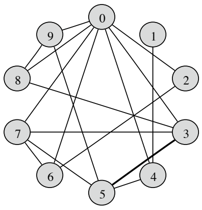

departure flow 52

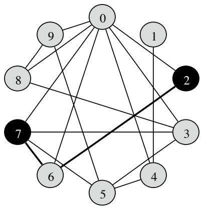

arrival 111

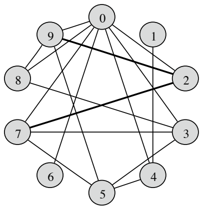

departure flow 111


arrival 112


departure flow 59


departure flow 78


arrival 113


departure flow 108

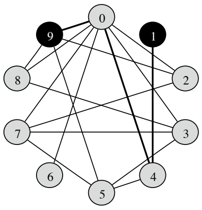
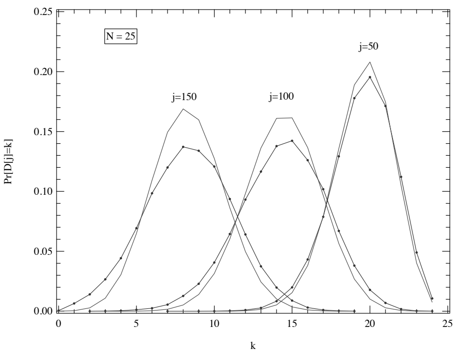

arrival 114

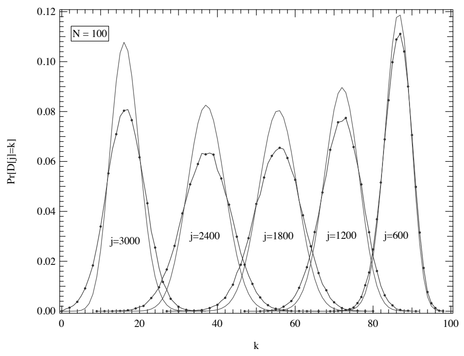
Figure A.1: *– continued –*

Appendix B

Appendix: Degree Distribution of $\hat{G}_j(N)$



(a) $N = 25$



(b) $N = 100$

Figure B.1: Degree distribution of $\hat{G}(j,N)$ for various N and j. As a reference, the degree distribution for the random graph has been added for each result with mean identical to the simulation. The dotted lines are simulation results, the normal lines are computations of the Erdös-Renyí random graph degree distribution.

Appendix C

The degree distribution in K_N after removing links

In this section we examine the effect of removing correlated links on the nodal degree distribution. To understand why a small deviation of the Erdös-Renyí model may lead to large differences observed in Section 4, we assume that *any path between two random nodes consists of two links*. We consider the process where, at each stage j, precisely two links of an arbitrary node in the complete graph K_N are removed. We compute the degree $D[j]$ of an arbitrary node at stage j in the thinned complete graph K_N. From that process, we derive a recursion relation

$$D[j] = D[j-1]\,1_{\text{no link removed}} + (D[j-1]-1)\,1_{\text{1 link removed}} + (D[j-1]-2)\,1_{\text{2 links removed}} \tag{C.1}$$

and $D[0] = N-1$. Ignoring the boundary restrictions[1] when $D[j] = 1$ and $D[j] = 0$, the probability density function of $D[j]$ obeys the equation

$$\begin{aligned}\Pr[D[j]=k] = {} & \Pr[D[j-1]=k]\Pr[\text{no link removed}] + \\ & + \Pr[D[j-1]=k+1]\Pr[\text{1 link removed}] + \\ & + \Pr[D[j-1]=k+2]\Pr[\text{2 links removed}]\end{aligned}$$

with $p_1 = \Pr[\text{1 link removed}] = \frac{2}{N}$, $p_2 = \Pr[\text{2 links removed}] = \frac{1}{N}$ and $\Pr[\text{no link removed}] = 1-\frac{3}{N}$. The corresponding probability generating function

$$\varphi_{D[j]}(z) = \sum_{k=0}^{N-1} \Pr[D[j]=k]\,z^k$$

[1] These seriously complicate the analysis and prevent the derivation of analytic expressions

obeys the functional equation

$$\varphi_{D[j]}(z) = \left(1 - \frac{3}{N}\right)\varphi_{D[j-1]}(z) + \frac{2}{N}\sum_{k=0}^{N-1}\Pr[D[j-1] = k+1]\, z^k + \frac{1}{N}\sum_{k=0}^{N-1}\Pr[D[j-1] = k+2]\, z^k$$

After simplification, we obtain

$$\varphi_{D[j]}(z) = \left(\left(1 - \frac{3}{N}\right) + \frac{2}{Nz} + \frac{1}{Nz^2}\right)\varphi_{D[j-1]}(z) - \left(\frac{2}{Nz} + \frac{1}{Nz^2}\right)\varphi_{D[j-1]}(0) - \frac{\Pr[D[j-1] = 1]}{Nz^2}$$

Since $\varphi_{D[j]}(1) = 1$, it follows that both $\Pr[D[j-1] = 1] = \varphi'_{D[j-1]}(0) = 0$ and $\Pr[D[j-1] = 0] = \varphi_{D[j-1]}(0) = 0$. The functional equation reduces to

$$\varphi_{D[j]}(z) = \frac{(N-3)z^2 + 2z + 1}{Nz^2}\varphi_{D[j-1]}(z)$$

With the initial condition $\varphi_{D[0]}(z) = E\left[z^{D[0]}\right] = z^{N-1}$, the solution is

$$\varphi_{D[j]}(z) = \left(\frac{(N-3)z^2 + 2z + 1}{Nz^2}\right)^j z^{N-1} \tag{C.2}$$

The mean and the variance of $D[j]$ are most efficiently computed from $L_{D[j]}(z) = \log\varphi_{D[j]}(z)$ as (see [43])

$$E[D[j]] = L'_{D[j]}(1) = N - 1 - \frac{4j}{N}$$
$$\mathrm{Var}[D[j]] = L''_{D[j]}(1) + L'_{D[j]}(1) = \frac{2j(3N-8)}{N^2}$$

The mean $E[D[j]]$ follows from the general law for the degree

$$\sum_{n=1}^{N} d_n[j] = 2L[j] = 2\left(\binom{N}{2} - 2j\right) \tag{C.3}$$

and the definition of the mean $E[D[j]] = \frac{1}{N}\sum_{n=1}^{N} d_n[j]$ resulting again in $E[D[j]] = N - 1 - \frac{4j}{N}$. However, the requirement that $\varphi'_{D[j-1]}(0) = 0$ and $\varphi_{D[j-1]}(0) = 0$ implies that $N - 2 - 2j > 0$ or that $\frac{N-2}{2} > j$. The major reason for this artifact is the neglect of the boundary equations.

By expanding the probability generating function in a Taylor series around $z = 0$, the probability density function $\Pr[D[j] = k]$ can be obtained. In general, the power series of $\left(z^2 + bz + c\right)^n$ with $n \in \mathbb{N}$ is derived as,

$$\left(z^2 + bz + c\right)^n = \sum_{j=0}^{n}\binom{n}{j}c^{n-j}\left(z^2 + bz\right)^j = \sum_{j=0}^{n}\binom{n}{j}c^{n-j}\sum_{k=0}^{j}\binom{j}{k}b^{j-k}z^{k+j}$$

```
1  INITIALIZE($K_N$)
2  while $K_N$ is connected
3    do node $v \leftarrow$ RANDOM-NODE($K_N$)
4      link $l_1 \leftarrow$ RANDOM-LINK($v$)
5      remove $l_1$ from $K_N$
6      if $degree[v] > 0$
7        then link $l_2 \leftarrow$ RANDOM-LINK($v$)
8          remove $l_2$ from $K_N$
```

Figure C.1: Meta-code for simulating effect of path removal on graph properties.

Let $m = k + j$, then $0 \le m \le 2n$ and, from $0 \le j = m - k \le n$, it follows that $k \le m$. Hence,

$$\left(z^2 + bz + c\right)^n = \sum_{m=0}^{2n} \sum_{k=0}^{m} \binom{n}{m-k} \binom{m-k}{k} c^{n-m+k} b^{m-2k} z^m \tag{C.4}$$

Using (C.4) with $b = \frac{2}{N-3}$, $c = \frac{1}{N-3}$ and $n = j$ leads to

$$\begin{aligned}
\varphi_{D[j]}(z) &= \left(\frac{N-3}{N}\right)^j \left(z^2 + \frac{2}{N-3} z + \frac{1}{N-3}\right)^j z^{N-1-2j} \\
&= \frac{1}{N^j} \sum_{m=0}^{2j} \left(2^m \sum_{l=0}^{m} \binom{j}{m-l} \binom{m-l}{l} \left(\frac{N-3}{4}\right)^l \right) z^{m+N-1-2j} \\
&= \frac{1}{N^j} \sum_{m=N-1-2j}^{N-1} \left(2^{m-N+1+2j} \sum_{l=0}^{m-N+1+2j} \binom{j}{m-N+1+2j-l} \times \right. \\
&\quad \left. \times \binom{m-N+1+2j-l}{l} \left(\frac{N-3}{4}\right)^l \right) z^m
\end{aligned}$$

from which it follows that

$$\Pr[D[j] = k] = \frac{2^{k-N+1+2j}}{N^j} \sum_{l=0}^{k-N+1+2j} \binom{j}{k-N+1+2j-l} \binom{k-N+1+2j-l}{l} \left(\frac{N-3}{4}\right)^l \tag{C.5}$$

Unfortunately, we cannot evaluate the above sum. We have performed simulations to test the validity of (C.5).

Figure C.1 presents the meta-code of the simulation. First the complete graph K_N is initialized. While K_N remains connected, a random node is chosen uniformly and two of its links are removed from the graph. The simulation results for $N = 50$ are presented in Figure C.2, which illustrates that deviations appear for increasing k due to the fact that the analysis above only applies for small $j < \frac{N-2}{2}$. Figure C.2 is “similar” to Figure 4.6.

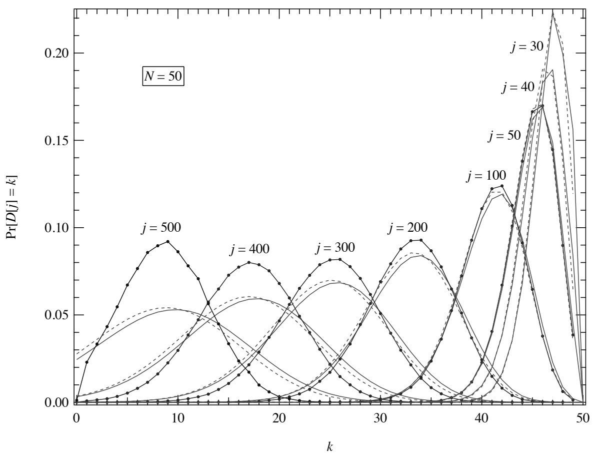


Figure C.2: The probability density function of $D[j]$ for $N = 50$ and various j. The line that connects dots are simulations, the solid lines are computations of (C.5) and the dashed lines correspond to computations of (C.9).

Since (C.5) is awkward and provides little insight, we now approximate $\Pr[D[j] = k]$. Let q denote the probability that a node belongs to a path between a uniformly chosen source-destination pair. Each path spans exactly two hops and we assume that a path can always be formed between the source-destination pair, which is a reasonable assumption when the graph is densely connected. Then, we can write,

$$q \approx \frac{3}{N} \tag{C.6}$$

The next step is to differentiate between the nodes based on the number of times they belonged to a path. The degree of a node is proportional to the number of paths through it. The average degree of the nodes containing m paths can be approximated by,

$$\begin{aligned} E[\mathcal{D}[m]] = k &\approx N - 1 - \frac{2}{3} \cdot m - \frac{1}{3} \cdot 2 \cdot m \\ &\approx N - 1 - \frac{4}{3} m \end{aligned} \tag{C.7}$$

Relation (C.7) provides a direct relation between the number of paths through a node and the expectation of the degree. The degree distribution of the entire network is found by considering each allocation as an i.i.d. event. The probability that a node belongs to m of the j total allocated paths is then independent and can be considered binomially distributed,

$$\Pr[\text{A node belongs to } m \text{ paths}] = \binom{j}{m} q^m (1-q)^{j-m} \tag{C.8}$$

The probability that a node belongs to m paths is written as a function of the node's expected degree by substitution of the inverse of (C.7) for m in (C.8),

$$\Pr\left[\text{A node belongs to } m \text{ paths}\right] \approx \Pr\left[\text{A node belongs to } \frac{3}{4}\left(N-1-k\right) \text{ paths}\right] \cdot \left|\frac{dm}{dk}\right|$$

and

$$\Pr\left[D[j]=k\right] \approx \Pr\left[\text{A node belongs to } m \text{ paths}\right]$$

so that,

$$\Pr\left[D[j]=k\right] \approx \left(\frac{3}{4}\right)\binom{n}{\frac{3}{4}\left(N-1-k\right)} q^{\frac{3}{4}(N-1-k)}\left(1-q\right)^{n-\frac{3}{4}(N-1-k)} \tag{C.9}$$

Figure C.2 shows the goodness of both approximations (C.5) and the simpler (C.9). Both approximations are significantly different from the simulated results, which underlines the importance of the dependencies between the links. Hence, the correlation between the links has been shown to be a very hard problem.

Appendix D

An approximate birth-and-death analysis

We present an approximate analysis of the network model of Chapter 3 using a birth-and-death steady-state analysis to compute the number of flows in the complete graph K_N.

D.1 Lower bound

We first assume that a flow between a source-destination pair consists of precisely one link in K_N. This analysis will give a lower bound for the average number of flows $\mathrm{E}[N_S]$ because, if the direct link between a source-destination pair is unavailable, there may still exist a path (with more than one hop). Hence, blocking occurs more often, which equivalently means that the network capacity to allocate flows is lower than in the network model of Chapter 3. The arrival rate of a flow is λ, while the departure rate of a flow is μ (irrespective of how many flows there are in the network). The state j denotes the number of flows in the network.

Since the allocation of a flow in the simplified analysis is equivalent to the removal of a random link, the probability that the chosen random link is already unavailable in state j equals $\frac{j}{L}$, where $L = \binom{N}{2}$, the total number of links in K_N. Hence, the effective arrival rate is $\lambda\left(1-\frac{j}{L}\right)$. Applying the birth-dead steady-state equations [43, p. 209] with effective arrival rate $\lambda\left(1-\frac{j}{L}\right)$ gives, for $j \geq 1$,

$$\begin{aligned}\pi_j &= \pi_0 \prod_{m=0}^{j-1} \frac{\lambda_m}{\mu_{m+1}} = \pi_0 \prod_{m=0}^{j-1} \frac{\lambda\left(1-\frac{m}{L}\right)}{\mu} = \pi_0 \rho^j \prod_{m=0}^{j-1} \left(\frac{L-m}{L}\right) \\ &= \pi_0 \left(\frac{\rho}{L}\right)^j \frac{L!}{(L-j)!}\end{aligned}$$

where $\rho = \frac{\lambda}{\mu}$ and

$$\pi_0 = \frac{1}{1+\sum_{j=1}^{\infty}\prod_{m=0}^{j-1}\frac{\lambda_m}{\mu_{m+1}}} = \frac{1}{1+L!\sum_{j=1}^{L}\frac{\left(\frac{\rho}{L}\right)^j}{(L-j)!}}$$
$$= \frac{1}{L!\sum_{k=0}^{L}\frac{\left(\frac{\rho}{L}\right)^{L-k}}{k!}} = \frac{1}{L!\left(\frac{\rho}{L}\right)^L\sum_{k=0}^{L}\frac{\left(\frac{L}{\rho}\right)^k}{k!}}$$

The average number of flows is

$$\mathrm{E}[N_S] = \sum_{j=0}^{L} j\pi_j = L!\pi_0\sum_{j=0}^{L}\frac{j\left(\frac{\rho}{L}\right)^j}{(L-j)!}$$
$$= L!\pi_0\left(\frac{\rho}{L}\right)^L\sum_{k=0}^{L}\frac{(L-k)\left(\frac{L}{\rho}\right)^k}{k!} = L - L!\pi_0\left(\frac{\rho}{L}\right)^{L-1}\sum_{k=0}^{L-1}\frac{\left(\frac{L}{\rho}\right)^k}{k!}$$

Hence,

$$\mathrm{E}[N_S] = L - \frac{\frac{L}{\rho}\sum_{k=0}^{L-1}\frac{\left(\frac{L}{\rho}\right)^k}{k!}}{\sum_{k=0}^{L}\frac{\left(\frac{L}{\rho}\right)^k}{k!}} = L\left(1-\frac{1}{\rho}\right) + \frac{L}{\rho}\frac{\left(\frac{L}{\rho}\right)^L}{L!\sum_{k=0}^{L}\frac{\left(\frac{L}{\rho}\right)^k}{k!}}$$

Using $\frac{\sum_{k=0}^{L}\frac{\left(\frac{L}{\rho}\right)^k}{k!}}{e^{L/\rho}/2} \to 1$ for $L \to \infty$ and Stirling's formula [2, 6.1.38], $L! \simeq \sqrt{2\pi L}L^Le^{-L}$, results in

$$\mathrm{E}[N_S] \sim L\left(1-\frac{1}{\rho}\right) + \frac{L}{\rho^{L+1}}2e^{\frac{L}{\rho}}\frac{L^L}{L!} \sim L\left(1-\frac{1}{\rho}\right) + \sqrt{\frac{2}{\pi}}\frac{\sqrt{L}}{\rho^{L+1}}e^{L\left(1-\frac{1}{\rho}\right)}$$

If $\rho \to 1$, then $\mathrm{E}[N_S] \sim 0.797\sqrt{L}$, while simulations in Figure 3.26 show a different behavior, $\lim_{\rho\to 1}\mathrm{E}[N_S] = \frac{K}{2} = \frac{\beta L}{2}$.

D.2 An upper bound

Rejection does not occur as long as there is a path between the source node and the destination node. A necessary condition to have a path is that the source and destination node belong to a same connected cluster. Since the cluster sizes in $G_p(N)$ are small compared to the giant component GC, and the source and destination nodes are randomly and independently chosen, we may approximate the path condition as $\Pr[\{\text{source} \in GC\} \cap \{\text{destination} \in GC\}] = (\Pr[\text{any node} \in GC])^2 = S_{GC}^2$, where $S_{GC} \in [0,1]$ is derived in [43, p. 339] and S_{GC} obeys the functional equation[1]

$$S_{GC} = 1 - (1 - pS_{GC})^{N-1}$$

[1]Rewriting the functional equation as $p(S_{GC}) = \frac{1}{S_{GC}}\left(1-(1-S_{GC})^{\frac{1}{N-1}}\right)$ gives the extreme values for the link density $p(0) = \frac{1}{N-1}$ and $p(1) = 1$.

that reduces for large N and constant average degree $\overline{D} = p\,(N-1)$ to

$$S_{GC} = 1 - e^{-D_{\mathrm{av}} S_{GC}}$$

Hence, given that j flows are allocated, the probability of blocking is

$$r_j = 1 - S_{GC}^2\,(j)$$

where the dependence on j is via the average degree $\overline{D} = p\,[j]\,(N-1)$. A Lagrange expansion of S_{GC} as a function of $\overline{D}$ is given in [43, p. 339, eq. (15.21)]. Similarly as in Section D.1, the effective rate is $\lambda\,(1-r_j) = \lambda S_{GC}^2\,(j)$ and

$$\pi_j = \pi_0 \prod_{m=0}^{j-1} \frac{\lambda_m}{\mu_{m+1}} = \pi_0 \prod_{m=0}^{j-1} \frac{\lambda S_{GC}^2\,(m)}{\mu} = \pi_0\, \rho^j \prod_{m=0}^{j-1} S_{GC}^2\,(m)$$

From Section 4.2 we have that

$$p[j] \leq 2 - \left(1 + \frac{1}{L}\right)^j$$

Finally, we arrive at the upper bound

$$\mathrm{E}\,[N_S] \leq \sum_{j=0}^{L} j\pi_j = \frac{\sum_{j=1}^{L} j\rho^j \prod_{m=0}^{j-1} S_{GC}^2\,(m)}{1 + \sum_{j=1}^{L} \rho^j \prod_{m=0}^{j-1} S_{GC}^2\,(m)}$$

Numerical simulations show that $\lim_{\rho\to 1} E\,[N_f] = \frac{K}{2} = 0.34L$ or, that $\beta \simeq 0.68$, which indeed upper bounds $\beta_{\mathrm{sim}} \simeq 0.42$. The analysis also shows that improving the computations will be quite difficult.

Appendix E

Abbreviations

AS	Autonomous System
BGP	Border Gateway Protocol
CBR	Call Blocking Rate
CCDF	Complementary Cummulative Distribution Function
DES	Discrete Event Simulation
DiffServ	Differentiated Services
FIFO	First In First Out
FTP	File Transfer Protocol
FttH	Fiber to the Home
GC	Giant Component
GPS	Global Positioning System
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunication Union
LSA	link state Advertisment
LSP	Label Switched Path
LSU	link state Update
LSUP	Link State Update Policy
MCOP	Multi-Constrained Optimal Path
MCP	Multi-Constrained Path
MIRA	Minimum Interference Routing Algorithm
MPLS	Multi-Protocol Label Switching
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
QoS	Quality of Service
RIPE	Réseaux IP Européens
RSVP	Resource Reservation Protocol

SAMCRA	Self-Adaptive Multiple Constraints Routing Algorithm
SLA	Service Level Agreement
SOC	Self-Organized Criticality
SP	Shortest Path
SSP	Static Shortest Path
TAMCRA	Tunable Accuracy Multiple Constraints Routing Algorithm
TCP	Transmission Control Protocol
TE	Traffic Engineering
TTM	Test Traffic Measurement
VoIP	Voice over IP
WSP	Widest-Shortest Path

Appendix F

Acknowledgements

This thesis contains the work that I have performed during the past five years at the Network Architectures and Services group at the Electrical Engineering, Mathematics and Science faculty at Delft University of Technology. If I would ever write my memoirs, which I deem unlikely at this moment considering the energy it has taken me to write this thesis, then my stay at the 19th floor will be remembered as very pleasant and educative. Here I would like to thank the people that helped in creating this valuable and inspiring environment.

My sincere gratitude goes to my advisor, promotor and, occasionally, mental coach Prof. Piet Van Mieghem, for his guidance, patience and confidence. I am very thankful for being given the opportunity to achieve "de hoogsthaalbare academische graad" and for sharing the knowledge and insights. The enjoyment of working at the 19th floor would not have been possible without my colleagues and friends Bingjie, Santpal, Fernando, Huijuan, Jasmina, Anteneh, Javier, Christian, Yue, Siyu, Steve, Almerima, Xiaoming, Milena, Antonio, Rob, Edgar, Eguzki and all the others. I would like to thank Marjon, Wendy, Laura and Rowena for their assistance during these years. Special thanks go to Jos Weber for the inspirational tennismatches we had the past five years.

I am indebted to the committee members for their comments, suggestions, time and effort to serve in the defense committee. In particular I would like to thank Gerard Hooghiemstra for the pleasant cooperation and his efforts on the queueing model. A special thanks go to Stefano Avallone from Naples university for his assistance and cooperation in setting up the network testbed. Furthermore, I would like to thank STW and in particular Wouter Segeth and the user committee members for their advise and useful suggestions during the project meetings. I wish to acknowledge my brother Ernstjan Kleiberg for assisting me in designing the cover of this thesis and helping me out whenever I needed help.

Finally, I would like to thank my friends for their support and friendship. Last, but not least, I want to thank my parents Ad en Cocky and my girlfriend Coby for their unconditional love and confidence. Without your support this achievement would not have been possible.

Appendix G

Curriculum Vitae

Teunis Johannis (Tom) Kleiberg was born in Dordrecht (the Netherlands) at April 2, 1977. He graduated from the Faculty of Electrical Engineering at Delft University of Technology in 2000. His Master of Science graduation project was in the Network Architectures and Services group headed by the then recently appointed professor Piet Van Mieghem. His graduation project consisted in the development of software to visualize graphs and analyze their properties. After his graduation, he joined Ericsson EuroLab Netherlands in Rijen, where he worked on the development of a management system for Interactive Messaging Systems. In 2002, the difficult times in the telecommunications industry forced him to leave Ericsson, after which he made a trip of one year through Australia, New-Zealand and Indonesia. Upon his return to the Netherlands in 2003, he rejoined the group of prof. Piet Van Mieghem, but now as a PhD student working on network dynamics and quality-of-service. He represented the finalist of the TU Delft for the dutch KIvI telecommunication contest held between the three technical universities in the Netherlands in 2007. He is now active as a post-doc researcher within the Network Architectures and Services group, where he is working on implementation of networking testbed and on robustness of complex networks.

Publications

1. J. M. Hernandez, T. Kleiberg, H. Wang and P. Van Mieghem,"A Qualitative Comparison of Power Law Generator", Proc. of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007), July 16-18, San Diego, California, USA.

2. T. Kleiberg, B. Fu, F. Kuipers, P. Van Mieghem, S. Avallone, B. Quoitin, "DeSiNe: a flow-level QoS Simulator of Networks", First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems, 2007, Marseille, France

3. D. K. Agrawal, T. Kleiberg, S. Papp, R. E. Kooij and P. Van Mieghem,"Quantifying the Quality of Service of Streaming Media in Differentiated Services Network", 2007 International Conference on Software, Telecommunications and Computer Networks (IEEE SoftCom 2007), September 27-29, Split-Dubrovnik, Croatia.

Bibliography

[1] J. Abate and W. Whitt. Calculating time-dependent performance measures for the M/M/1 queue. *IEEE/ACM Transactions on Communications*, 37(10):1102–1104, Oct 1989.

[2] M. Abramowitz and J. A. Stegun. *Handbook of Mathematical Functions*. Dover Publications Inc., New York, 1968.

[3] J. S. Ahn and P. B. Danzig. Packet network simulation: Speedup and accuracy versus timing granularity. *IEEE/ACM Transactions on Networking*, 4(5):743–757, 1996.

[4] D. Alderson, H. Chang, M. Roughan, S. Uhlig, and W. Willinger. The many facets of Internet topology and traffic. *Networks and Heterogenous Media*, 1(4):569–600, 2006.

[5] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system and multiple sources. *The Bell System Technical Journal*, 61(8):1871–1894, 1982.

[6] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi. Quality of service based routing: A performance perspective. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 17–28, Vancouver, BC, Canada, September 1998. ACM.

[7] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In J. M. Almeida, V. A. F. Almeida, and P. Barford, editors, *Internet Measurement Conference (IMC)*, pages 153–158, Rio de Janeriro, Brazil, October 2006. ACM.

[8] S. Avallone, F.A. Kuipers, G. Ventre, and P. Van Mieghem. Dynamic routing in QoS-Traffic Engineered networks. In *EUNICE IFIP WG 6.6, WG 6.4 and WG 6.9 Workshop*, pages 222–228, Colmenarejo, Spain, Jul 2005.

[9] P. Bak. *How Nature Works*. Copernicus, Springer-Verlag New York Inc., New York, NY, USA, 1996.

[10] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: An explanation of the $1/f$ noise. *Physical Review Letters*, 59(4):381–384, July 1987.

[11] G. Banerjee and D. Sidhu. Comparative analysis of path computation techniques for MPLS traffic engineering. *Computer Networks*, 40(1):149–165, 2002.

[12] S. Banerjee, T. G. Griffin, and M. Pias. The interdomain connectivity of PlanetLab nodes. In *Proceedings of the Passive and Active Network Measurement, International Workshop, PAM*, Antibes Juan-les-Pins, France, April 2004. Springer.

[13] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[14] O. Brun and J.-M. Garcia. Analytical solution of finite capacity M/D/1 queues. *Journal of Applied Probability*, 37(4):1092–1098, December 2000.

[15] K. Christensen, Z. Olami, and P. Bak. Deterministic $1/f$ noise in nonconserative models of self-organized criticality. *Physical Review Letters*, 68(16):2417–2420, April 1992.

[16] I. Csabai. $1/f$ noise in computer network traffic. *Journal of Physics A: Mathematical and General*, 27(12):L417–L421, 1994.

[17] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[18] A. J. Field, U. Harder, and P. G. Harrison. Measurement and modelling of self-similar traffic in computer networks. *IEE Proceedings Communications*, 151(4):355–363, August 2004.

[19] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[20] E. Gavron. NANOG traceroute, 1995. for latest version, see ftp://ftp.login.com/pub/software/traceroute/.

[21] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. Susanj, H. Uijterwaal, and R. Wilhelm. Providing active measurements as a regular service for ISPs. In *Proceedings of the Passive and Active Network Measurement, International Workshop, PAM*, Amsterdam, Netherlands, April 2001. Springer.

[22] G. Grimmett. *Percolation*. Springer-Verlag, New York, NY, USA, 1989.

[23] R. A. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proceedings of the Global Communications Conference GLOBECOM '97*, Phoenix, AZ, USA, November 1997. IEEE.

[24] Y. Hyun, A. Broido, and kc Claffy. On third-party addresses in traceroute paths. In *Passive and Active Measurement Workshop*, La Jolla, CA, USA, April 2003.

[25] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *Internet Measurement Conference (IMC)*, pages 237–242, Marseille, France, November 2002. ACM.

[26] I. Iliadis and D. Bauer. A new class of online minimum-interference routing algorithms. In *NETWORKING '02: Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, volume 2345 of *Lecture Notes in Computer Science*, pages 959–971, London, UK, May 2002. Springer-Verlag.

[27] ITU-T. Recommendation E.800: Terms and definitions related to quality of service and network performance including dependability, August 1994.

[28] ITU-T. Recommendation X.200: Open systems interconnection - basic reference model, July 1994.

[29] V. Jacobson. traceroute, 1989. see ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[30] S. Janson, D. E. Knuth, T. Luczak, and B. Pittel. The birth of the giant component. *Random Structures Algorithms 4*, 3:231–358, 1993.

[31] K. Kar, M. Kodialam, and T. V. Lakshman. Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. *IEEE/ACM Journal of Selected Areas in Communications*, 18:2566–2579, December 2000.

[32] B. Kaulakys, V. Gontis, and M. Alaburda. Point process model of $1/f$ noise vs a sum of Lorentzians. *Physical Review E*, 71(5):051105, 2005.

[33] B. Kaulakys and T. Meškauskas. Modeling $1/f$ noise. *Physical Review E*, 58(6):7013–7019, December 1998.

[34] A. Kavimandan, W. Lee, M. Thottan, A. Gokhale, and R. Viswanathan. Network simulation via hybrid system modeling: A time-stepped approach. In *14th International Conference on Computer Communications And Networks (ICCCN)*, San Diego, CA, 2005.

[35] G. Kesidis, A. Singh, D. Cheung, and W. Kwok. Feasibility of fluid event-driven simulation for ATM networks. In *IEEE Globecom*, London, 1996.

[36] C. Kiddle, R. Simmonds, C. L. Williamson, and B. Unger. Hybrid packet/fluid flow network simulation. In *17th Workshop on Parallel and Distributed Simulation, PADS*, pages 143–152, San Diego, CA, USA, 2003. ACM.

[37] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of Internet stability and backbone failures. In *FCTS*, pages 278–285, Madison, Wisconsin, USA, June 1999. IEEE Computer Society.

[38] B. Lekovic and P. Van Mieghem. Link state update policies for quality of service routing. In *IEEE 8th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, pages 123–128, Delft, The Netherlands, October 2001.

[39] B. Liu, Y. Guo, J. F. Kurose, D. F. Towsley, and W. Gong. Fluid simulation of large scale networks: Issues and tradeoffs. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 2136–2142, Las Vegas, Nevada, USA, 1999. CSREA Press.

[40] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of failures in an IP backbone network. In *INFOCOM*, Hong Kong, China, March 2004. IEEE.

[41] B. Melamed, S. Pan, and Y. Wardi. Hybrid discrete-continuous fluid-flow simulation. In *SPIE International Symposium on Information Technologies and Communications (ITCOM 01), Scalability and Traffic Control in IP Networks*, pages 263–270, Denver, CO, 2001.

[42] P. Van Mieghem. *Data Communications Networking*. Techne Press, Amsterdam, The Netherlands, 2006.

[43] P. Van Mieghem. *Performance Analysis of Communications Networks and Systems*. Cambridge University Press, 2006.

[44] P. Van Mieghem and F. A. Kuipers. Concepts of exact QoS routing algorithms. *IEEE/ACM Transactions on Networking*, 12(5):851–864, October 2004.

[45] P. Van Mieghem and S. van Langen. Influence of the link weight structure on the shortest path. *Physical Review E*, 71(5):056113, May 2005.

[46] V. Misra, W.-B. Gong, and D. F. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *SIGCOMM*, pages 151–160, 2000.

[47] H. De Neve and P. Van Mieghem. TAMCRA: A tunable accuracy multiple constraints routing algorithm. *Computer Communications*, 23:667–679, 2000.

[48] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, September 2005.

[49] D. M. Nicol, J. Liu, M. Liljenstam, and G. Yan. Simulation of large scale networks i: Simulation of large-scale networks using SSF. In *Winter Simulation Conference*, pages 650–657, 2003.

[50] T. Ohira and R. Sawatari. Phase transition in computer network traffic model. *Physical Review E*, 58(1):193–195, 1998.

[51] R. V. Oliveira, B. Zhang, and L. Zhang. Observing the evolution of Internet AS topology. In J. Murai and K. Cho, editors, *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 313–324, Kyoto, Japan, August 2007. ACM.

[52] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[53] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD dissertation, University of California, Lawrence Berkeley National Laboratory, April 1997.

[54] W. H. Press, S. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[55] H. Pucha, Y. Zhang, Z. Morley Mao, and Y. Charlie Hu. Understanding network delay changes caused by routing events. In L. Golubchik, M. H. Ammar, and M. Harchol-Balter, editors, *SIGMETRICS*, San Diego, California, USA, June 2007. ACM.

[56] A. Shaikh. *Management of Routing Protocols in IP Networks*. PhD thesis, University of California, Santa Cruz (UCSC), December 2003.

[57] A. Shaikh, J. Redford, and K. G. Shin. Evaluating the impact of stale link state on quality-of-service routing. *IEEE/ACM Transactions on Networking*, 06(02):162–176, April 2001.

[58] O. P. Sharma. *Markovian Queues*. Series in Mathematics and its Applications. Ellis Horwood, 1990.

[59] O. P. Sharma and A. M. K. Tarabia. A simple transient analysis of an M/M/1/N queue. *Sankhyā: The Indian Journal of Statistics*, 62(A, Pt. 2):273–281, 2000.

[60] R. J. Sivasankar, S. Ramam, S. P. Subrahmaniam, T. Srinivasa Rao, and D. Medhi. Some studies on the impact of dynamic traffic in QoS-based dynamic routing environment. In *Proceedings of the 2000 IEEE Internation Conference of Communications (ICC)*, pages 959–963, New Orleans, LA, USA, June 2000. IEEE.

[61] R. V. Solé and S. Valverde. Information transfer and phase transitions in a model of Internet traffic. *Physica A*, 289(3):595–605, January 2001.

[62] N. T. Spring, R. Mahajan, D. Wetherall, and T. E. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[63] L. Takács. *Introduction to the Theory of Queues*. Oxford University Press, New York, 1962.

[64] A. Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 2002.

[65] A. M. K. Tarabia. Transient analysis of an M/M/1/N queue - an alternative approach. *Tamkang Journal of Science and Engineering*, 3(4):263–266, 2000.

[66] R. Teixeira and J. Rexford. Managing routing disruptions in Internet service provider networks. *IEEE Communications Magazine*, 44(3):160–165, March 2006.

[67] M. Toren. Tcptraceroute, 2001. for latest version, see http://michael.toren.net/code/tcptraceroute/.

[68] B. Wang, X. Su, and C. L. P. Chen. A new bandwidth guaranteed routing algorithm for MPLS traffic engineering. In *Proceedings of the 2002 IEEE Internation Conference of Communications (ICC)*, New York City, NY, USA, April 2002. IEEE.

[69] F. Wang, N. Feamster, and L. Gao. Measuring the contributions of routing dynamics to prolonged end-to-end Internet path failures. In *Proceedings of the Global Telecommunications Conference GLOBECOM '07*, Washington, DC, USA, November 2007. IEEE.

[70] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end Internet path performance. In L. Rizzo, T. E. Anderson, and N. McKeown, editors, *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 375–386, Pisa, Italy, September 2006. ACM.

[71] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, February 1997.

[72] T. De Wolf and T. Holvoet. Emergence versus self-organisation: Different concepts but promising when combined. In *Engineering Self-Organising Systems: Methodologies and Applications*, volume 3464 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, 2005.

[73] A. Yan and W.-B. Gong. Time-driven fluid simulation for high-speed networks. *IEEE Transactions on Information Theory*, 45(5):1588–1599, 1999.

[74] J. Yuan, Y. Ren, and X. Shan. Self-organized criticality in a computer network model. *Physical Review E*, 61(2):1067–1071, February 2000.

[75] T. Yung, J. Martin, M. Takai, and R. Bagrodia. Integration of fluid-based analytical model with packet-level simulation for analysis of computer networks. In R. D. van der Mei and F. Huebner-Szabo de Bucs, editors, *SPIE*, volume 4523 of *Internet Performance and Control of Network Systems II*, pages 130–143, 2001.

[76] P. Zhang, R. Kantola, and Z. Ma. Design and implementation of a new routing simulator. In *Proceedings of the 2000 SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), July 16-20, 2000 - Vancouver, BC, Canada*, Vancouver, BC, Canada, July 2000. SCS.

[77] P. Zhang, Z. Ma, and R. Kantola. Designing a new routing simulator for DiffServ MPLS networks. In *Proceedings of the 2001 SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), July 15-19, 2000 - Orlando, Florida, USA*, Orlando, Florida, USA, July 2000. SCS.

[78] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. ACIRI technical report, AT&T Centre for Internet Research at ICSI, 2000.