Date approved Grade

Reviewer

Routing in anonymous networks as a means to prevent traffic analysis

Sami Lunnamo

Helsinki 02/05/2016

University of Helsinki
Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion – Faculty/Section Faculty of Science	Laitos – Institution – Department Department of Computer Science	
Tekijä – Författare – Author Sami Petteri Lunnamo		
Työn nimi – Arbetets titel – Title Routing in anonymous networks as a means to prevent traffic analysis		
Oppiaine – Läroämne – Subject Computer Science		
Työn laji – Arbetets art – Level M. Sc. Thesis	Aika – Datum – Month and year 28.4.2016	Sivumäärä – Sidoantal – Number of pages 67 pages + 10 pages of attachments

Tiivistelmä – Referat – Abstract

Traditionally, traffic analysis is something that has been used to measure and keep track of a network's situation regarding network congestion, networking hardware failures, etc. However, largely due to commercial interests such as targeted advertisement, traffic analysis techniques can also be used to identify and track a single user's movements within the Internet.

To counteract this perceived breach of privacy and anonymity, several counters have been developed over time, e.g. proxies used to obfuscate the true source of traffic, making it harder for others to pinpoint your location. Another approach has been the development of so called anonymous overlay networks, application-level virtual networks running on top of the physical IP network. The core concept is that by the way of encryption and obfuscation of traffic patterns, the users of such anonymous networks will gain anonymity and protection against traffic analysis techniques.

In this master's thesis we will be taking a look at how message forwarding or packet routing in IP networks functions and how this is exploited in different analysis techniques to single out a visitor to a website or just someone with a message being forwarded through a network device used for traffic analysis. After that we will discuss some examples of anonymous overlay networks and see how well they protect their users from traffic analysis, and how do their respective models hold up against traffic analysis attacks from a malicious entity. Finally, we will present a case study about Tor network's popularity by running a Tor relay node and gathering information on how much data the relay transmits and from where does the traffic originate.

CCS-concepts:
• **Security and privacy~Privacy protections**
• **Networks~Overlay and other logical network structures**
• *Information systems~Traffic analysis*

Avainsanat – Nyckelord – Keywords
Routing, Overlay networks, Tor, Freenet, GNUnet, privacy

Säilytyspaikka – Förvaringställe – Where deposited
Kumpula Science Library, serial number C-

Muita tietoja – Övriga uppgifter – Additional information

Table of contents

1 Introduction

As has been the trend for quite a few years now, the significance of the Internet is still steadily growing, whether in the personal, occupational or even national context. More and more data is being saved to cloud services, personal information is being made available over social media networks and all of that data is being collected, analysed and unified with other bits of information by numerous different operators for their own purposes. The most common use for a person's personal information has to do with more customized advertising, and it is a very common occurrence nowadays. Google shows customized advertisements based on your search, several sites have advertisements based on how you landed on the site, even video games display customized advertisements to you as a part of their scenic backgrounds. Based on this we know there is a market for the information of what content you as an internet user search for.

At the same time as we are wilfully publishing more and more personal information about ourselves in several different social media platforms, concerns about privacy issues are being brought up more regularly. A typical internet service which requires registration and offers users the opportunity to buy goods online is required to have some information about its customers. This blob of information includes but is usually not limited to real name, living address, email address and telephone number. Additionally the user is most often given the option to save their credit card information to the service, in order to speed up the checkout process within the service by removing the need to input lengthy serial numbers. Considering a service like Sony's Playstation Network (PSN), servicing millions of users daily, it should come as no surprise that it is a high priority target for a malicious attacker and successful hacks into a service like the Sony PSN can have huge repercussions [Wil11].

So we know that an internet user's personal information is something that different agents, malicious or otherwise, want to get their hands on. Additionally we know that a lot of advertising platforms and companies utilize data on what content internet users search for. What is not as often brought up is the fact that even the information of who you are talking with is perceived as valuable information for some. For example, the

National Security Agency (NSA) in America has collected this so-called “metadata” for years after the terrorist attacks of 9/11, until very recently when that right expired and was not renewed [Mac15]. Encryption allows two communicating parties to be reasonably assured that the contents of their communication stay private, but the problem is that the parties themselves are still easily discernible. It is fairly easy to see why this would be a problem, examples include a reporter talking to a source wishing for anonymity on a volatile subject that might garner the notice of intelligence agencies, such as the Snowden case, or a network of political activists against the current political ideology in a conservative and censorship-prone nation like China. The examples are extreme and deal with certain types of people, but there are also scenarios that concern the general population as well, such as a married man or woman who talks to a lawyer about divorce. Basically any dialogue with someone where one or both parties want to keep the dialogue itself a secret is in jeopardy via this collection of metadata, or more accurately, traffic analysis. The parties may of course try to circumvent this by utilizing a link to the internet which is not directly linkable to him or herself, like using a computer in an internet café. However, it may not always be feasible to search for such an internet connection and obviously it negatively impacts the ease of use of the internet as a communication medium [PfW86]. It should also be noted, that it is theoretically possible for a malicious operator to hold control over all the components needed for communication, including the internet service providers, routers and autonomous systems on the path of the messages, even make himself seem like the intended target of your messages. In these cases it is virtually impossible to have a technical implementation that would protect the user from losing information to parties he didn't intend the information for [PfW86].

In time, several different technologies have emerged that all try to somehow hide the participants of any given message exchange. Together they are called anonymous overlay networks, overlay network meaning that they are networks build over the traditional IP network, but utilizing their own routing protocols. Some of these anonymous overlay networks include GNUnet, Freenet and the popular Tor network. Each of them have different motivations and views on what is the most important aspect of anonymisation for their users. GNUnet and Freenet are mainly anonymous networks used for anonymous sharing of data and files, while the Tor network utilizes onion routing, a way of emphasizing anonymisation of communicating parties.

1.1 IP network routing

In the internet, two machines communicate with each other by sending each other a number of datagrams or packets. In theory one packet sent by a web server to a client computer could contain one web page the client's web server would then render as a page for the user, but nowadays the size of one complete message is regularly far greater than what one packet can contain. Therefore for the delivery of one complete web page, for example, several packets have to be sent between the machines [KuR08].

The maximum size of a packet is not a constant, but can vary depending on the physical quality of the wires. This is important to note, because through this we come to understand that because the whole physical network for the internet actually consists of separate links of physical wires between two routers, the maximum supported packet size could theoretically vary between every two router [KuR08, Com13].

Earlier we briefly mentioned routers and links. Routers are physical machines equipped with the capability to receive a packet and determine where to send it next. A router is typically connected to at least two different networks, A and B, so we say that it has a link to network B. We call this connection to network B a link because it is thus implied that there may be more routers on the way from our router to this target network [KuR08]. The router can receive a packet belonging to a bigger message from one machine on network A and determine if the packet needs to be forwarded to network B or back to network A. This determination is done via a routing table, a logical construct which has instructions about where to forward the packet based on the destination IP address in the packet headers [KuR08].

1.2 Traffic analysis and measurement

The act of traffic analysis actually governs a wide array of methods and tools. In the widest sense of the term, it means the measurement, categorization and analysis of transmission of packets, their timing, contents, frequency and destinations. The aim is to gather relevant information regarding e.g. possible problems within the network – like congestion – and performance evaluation for services and points of improvement, as well as for accounting purposes [Wil01, KWH05].

The methods for this analysis vary greatly and they incorporate both software and hardware approaches. The hardware approach to traffic analysis usually requires that a special-purpose hardware tool is installed within the network somewhere close to a network hub, multiple routers under heavy load. These hardware solutions are often quite reliable and accurate but often very expensive. By contrast, software solutions usually tend to be cheaper, but might not provide as clear of a perspective by comparison. Software solutions might be either modified network drivers with packet-capture ability on several different workstations, or it might be some traffic classification algorithms running on the server-side [Wil01, KWH05].

1.3 Overlay networks

In the most general sense, an overlay network is a group of devices forming a connection between each other utilizing some other protocol or logic than what is provided by the underlying network infrastructure. Most usually the underlying network infrastructure is the normal IP network, but in some cases it might also be another overlay network. Whatever the case, the nodes or devices in the overlay network are connected through a logical or virtual link. What is notable is that the logical link is not concerned with the distance of the nodes through the underlying network infrastructure [JGJ00]. Therefore, overlay networks are distributed networks in nature, no single authoritative server is in charge of keeping the network topology in memory and knowing how to route messages, but rather they are self-organizing [LCP05]. Curiously, it should be noted, that the IP network itself started as an overlay network for the telecommunication network, the logical link there being the IP address itself.

The motivation for building an overlay network is to strive for a better quality of service (QoS) in an internet service. Traditionally overlay networks have been constructed to provide some form of efficient multicasting – a way of sending one message to multiple recipients – or to provide effective content distribution [JGJ00, LCP05]. Nowadays these overlay networks have been built in multitudes, mostly building on top of some of the most known and effective overlay network schemes, like Chord or Pastry [LCP05].

1.4 Anonymous overlay networks

Anonymous overlay networks or anonymous networks are a specialized sub-category of overlay networks, with focus on masking and obfuscating as much information about its users as possible. Encrypting messages is possible and encouraged even with communication over the traditional IP network, therefore anonymous overlay networks focus on protecting sender and recipient anonymity as well as their separation, meaning that the sender of a message cannot be linked to a recipient of a message [PfW86].

Over time, several different types of anonymous overlay networks have risen. Most notable types are file-sharing overlay networks like Freenet, and anonymous web browsing and communication overlays like the Tor network.

It is prudent to note, that even though this master's thesis focuses on the effect of anonymous overlay networks on traffic analysis and classification methods and their ability to protect user privacy, user privacy is not completely guaranteed by successfully masking communication partners or the used applications, which is what traffic analysis and classification aims to achieve. It is enough for a malicious agents to successfully set out any user from a crowd for future tracking. This act is called fingerprinting and an example how this might be achieved is by following a user's mouse scrolling patterns [Nor16]. Another approach is requesting HTTP headers to specify browser plugins and their versions, which might be enough to single any user out. So even though anonymisation of traffic is an important aspect of user privacy, it is important to keep in mind that the whole problem domain of user privacy protection is far larger.

1.5 Structure and research question of this thesis

In order for us to properly understand the motivation for these anonymous overlay networks, we need to take a closer look at the motivation for building these networks. In this master's thesis we will familiarize ourselves with the workings of routing in traditional IP networks, how it enables traffic analysis, what is traffic analysis and what are the potential implications for personal information security, and how three different anonymous networks handle these threats.

In the chapter IP network routing, we will start with the most common knowledge, the way traffic is routed in a normal IP network. We will cover areas related to path selection for messages, like how routing tables are constructed, how delivery semantics affect path selection, and what are autonomous systems and how does having to traverse through multiple autonomous systems affect path selection choices. Different routing protocols are also discussed, and the importance of different routing metrics in regards to the choice of your routing protocol is brought up.

In chapter three, we will involve ourselves with the act of traffic analysis. Different modes of analysis will be introduced and the possible repercussions on privacy will be discussed.

After we've learnt why we would want to hide our communication from traffic analysis, we will then take a look at different tools that enable us to do so. Chapter four, “Overlay networks”, will familiarize us with the concept of an overlay network and what it means to have a network built on top of the IP network. The concept of overlay networks will then be expanded with the concept of anonymous overlay networks, overlay networks with the single focus of masking their traffic within the underlying IP network. Three different such anonymous overlay networks will be presented, the GNUnet, Freenet and the Tor network. For each overlay network we will discuss how they accomplish their anonymisation and compare it to what we already know about different means of traffic analysis.

In chapter five we will present results of running a Tor relay node as a case study for this master’s thesis. After running a relay node for a few months, we will take a look at how much traffic has been routed through it, in how many routing groups has it been a part of and from where in the world has it received data. This is interesting as a validation for the popularity of the Tor network, and it will also imply something about the current needs for such traffic anonymisation around the globe.

Finally, in the Discussion section we will pull together what we've learned and present final thoughts about the effects of analysing internet traffic in bulk, and how overlay networks help combat these problems.

For our thesis, the main points of interests are the effect of traffic analysis on privacy, and the ability of a given anonymous network to protect its users from this breach of privacy through methods of traffic analysis. Additionally, we want to know if the Tor's perceived popularity in part due to claims of strong anonymity is translatable to anecdotal evidence from running our own relay node within it. We can therefore form our primary research question thusly: “How does traffic analysis degrade an internet user's anonymity and how do GNUnet, Freenet and the Tor project offer effective protection against traffic analysis methods?” Our secondary research question has to deal with the popularity of Tor, and is as follows: “Can Tor's perceived popularity be witnessed from the viewpoint of a single relay operator as high traffic volume and a large number of connections?”

2 IP network routing in IPv4 networks

As previously stated in the Introduction section, the act of routing means forwarding a datagram or message composed of multiple packets over the network to its destination. This routing and forwarding is performed by the router devices within the network, which is useful to keep in mind throughout this section. Routers are connected to different networks – and therefore other routers – through multiple network interfaces built within the routers.

When a client wishes to send a message to another specific device, it must know the IP address of the recipient and record it into the header information of the message being sent. The sender will then submit the message to the network by figuring out the correct router to pass the message to, which will then receive the message, review the destination IP address and figure out whether it is in direct connection with the destination or whether it needs to forward the message onwards to the next router [KuR08].

When performing routing, the router usually is not in direct contact with the destination device, and therefore can not directly forward the message to the right destination. However, all devices connected to the internet belong to a network, which might be a part of a little bit bigger network, which all together comprise the internet [KuR08, Com13]. That is not to say however, that all networks are contained within another, but they are all connected to each other. The IP address of a device is comprised of the subnet address and the device address parts, meaning that a router will always route packets towards the correct subnet and the router for the correct subnet usually is usually the first one to be interested about knowing the exact device address.

It should be noted that as part of the TCP protocol, each received packet is acknowledged by the destination machine to the sender. There are also other minute details related to the protocol of sending and receiving messages, as well as relating to the physical limitations of the network used [KuR08]. However, as we are interested in the act of routing, we will omit those details in this section. Additionally omitted is the process of resolving a name against a destination IP, which is the domain of Domain Name Services (DNS) but which can be for the purposes of this thesis thought of as a fairly straightforward transaction where the client contacts a DNS server with a URL and the DNS sever responds with an IP address.

Although the exhaustion of IPv4 addresses is currently one of the biggest problems of the IP network and the switch to IPv6 addresses is ongoing, this document will mention only IPv4 addresses where applicable. This is due to the fact that the version four addresses are easier to keep track of due to their more compact form, and because no matter the used IP address space, the mechanics of routing will remain the same.

2.1 Path selection

Routers all have the common purpose of routing received packets to their intended destination. Towards this goal they have to figure out the correct direction, that is, the correct next router closer to the subnet that houses the destination device. The determination process is possible because of the hierarchical nature of IP addresses and more specifically the subnets. We already know that IP addresses are made up of a subnet or network number part and the device address or host number part. Additionally, IP addresses are classified to one of three different classes, which determines how big of a portion belongs to either part [KuR08, Com13].

As previously mentioned, a router receiving a packet checks the destination address from the headers. If the address matches a network the router is directly connected to, the router will broadcast to all the devices on that network a request for the physical or MAC address of the device at the destination IP. The destination device should respond with its own MAC address, at which point the router will forward the packet to its intended destination.

If the destination address does not match any networks the router is directly connected to, it will compare the address against its routing table which contains directions for packet forwarding. The entry with the longest matching prefix in common with the packet's destination address will be chosen and the router will forward the packet according to the instructions in that entry [Com13]. If no entry matches any part of the destination IP of the packet, most routers also have a default entry as a fallback. Default entries might most commonly be seen in household routers that are usually connected to just the household's network and to the internet service provider's (ISP's) network through its single network interface. In these routers the routing table might be just the default entry for all packets to be forwarded to the ISP's network. In cases where there is no default entry and there is no match in the routing table, the router simply drops the packet and informs the sender.

Through the aforementioned forwarding or routing logic, it is theoretically possible that a packet could be lost indefinitely in the internet, for example due to erroneous routing. To counteract this, IP packets have a Time To Live (TTL) information set in the headers. The TTL is a number ranging from 1 to 255 and represents the number of times a packet can be forwarded before it must be dropped. Each receiving router must decrease the TTL by one and drop the packet if the TTL reaches zero. When a packet gets dropped due to its expiry, the dropping router sends back to the source IP address a so called Internet Control Message Protocol (ICMP) message informing the sender of the packet's expiry [KuR08, Com13]. Because a sender can not directly affect the route its datagrams will flow through, there is not much the message originator can do but retry sending in the hopes that the error was due to some temporary problem in the network, perhaps additionally setting the TTL value to be higher than it was on the initial send operation.

Through the expiry notification system, an interested party can get some knowledge of routing done on the way to his intended destination. A common network diagnostics tool called traceroute uses this feature in packet forwarding protocol and sends success-ive packets to specified destination IP with increasing TTLs starting from 1. Thus, because the first router, then the second router, etc. will send back a notification of packet expiry, and the notification will carry information about the sender – the router that dropped the initial packet – the user will gain knowledge of the chain of routers his packets will travel through on their way to the destination device.

2.1.1 Routing table

Like we already know, every packet has a destination towards which all routers forward said packet, based on the IP address in the headers. A routing table is a logical construct in each router the device consults for instructions on through which physical link it needs to forward the packet through.

A single entry on the table consists of at least three things: the network IP, a gateway IP and a cost metric. The network IP is self-evidently the intended destination of the packet, and the gateway IP is the address of the next router the packet needs to be for-warded to. The cost metric is some assigned value representing the gravity of using that specific route. The basis of the assigned value is dependant on the used emphases but could for example represent the latency of that route, or the number of hops as a multi-plier compared to other routes [Com13]. Routers might also have more data associated with an entry, like the physical interface the network is configured to. Table 1 shows an example routing table for some machine. The netmask information in the table repres-ents how many bits of each 8-bit block the address has used for the network part of an address. For example on row 2 the network 192.168.0.0 has a netmask of 255.255.255.0, meaning that the network part consists of the first 24 bits and therefore the last 8 bits are reserved for host addresses.

The gateway information in table 1 leads us to two bits of information about the device and where it is connected to. The next hop destination for default and the subnet 209.85.128.0 is 192.168.0.1 but for the subnet 192.168.0.0 itself it is set as 192.168.0.100. Since our device has two hosts known from within that specific network, we can easily conclude that our device must actually reside in that network. Further-more, the second bit of information would be to conclude that since our device communicates into that network through the address of 192.168.0.100, it probably is the IP address of our own device.

Network Destination	*Netmask*	*Gateway*	*Metric*
0.0.0.0	0.0.0.0	192.168.0.1	10
192.168.0.0	255.255.255.0	192.168.0.100	1
209.85.128.0	255.255.255.0	192.168.0.1	1

Table 1. An example routing table

The way routers determine which interface to use is called the longest matching prefix system and involves the device going through its whole routing table and checking the destination address against each of the table's entries. Now, because of the explosive growth in the number of devices connected to the internet, and the routing tables nature as a small logical construct preserved most often in a router's very limited memory, it has become a challenge for routers to contain all of the required information to all of the reachable networks [KuR08, Com13].

2.1.2 Subnets

The most simple explanation for a subnet is that it is an isolated part of a normal net-work. A subnet behaves like any other network, in that it usually reachable only through a gateway and that it can be governed by its own firewall rules. This mechanic is most often employed by sites where more fine-grained security on the network level is re-quired or when a hierarchical structure of networks is desirable for routing purposes. A subnet can be formed on an arbitrary point in an IP address's bit position, through Class-less Inter-Domain Routing (CIDR, which, as the name suggests, is designed to allow more efficient routing within a single domain [Com13].

Historically IP addresses were classified and handed out in one of three standard classes, A, B or C. Each of the classes corresponded to 8-bit boundaries of an IP address, so that a class A address used the first 8 bits of the whole 32 bit IP address space for representing the network, and the rest for representing hosts. Respectively, a class B address reserved 16 bits for networks and 16 bits for hosts, etc. [KuR08]. However, this scheme was quickly found non-scalable because of its inflexible nature – many sites required more hosts than a class C network would allow with its 256 host addresses, but far less than a class B block's 65 536. These sites were however being handed class B address spaces, which then started to quickly run out, due to there being only roughly 16 000 of them. This is the IPv4 address exhaustion problem in a nutshell that we are dealing with even today.

The solution CIDR introduced, was the ability for sites to decide on an arbitrary bit in their address space, from which to split their own network into two subnets. As such, a subnet then is a logical but visible splitting of network into two. This split is notated by using the normal decimal notation of an IP address to denote the network part of the IP address and concatenating a forward slash and the number of bits used for the network part of the address. Therefore, a normal class C address in CIDR notation could be e.g. 156.28.243.0/24 and a subnet for that network could be 156.28.243.64/26. In that subnet, the two most significant bits of the last 8-bit block are now used for the network address, namely the 8th (2^7) bit is set at 0 and the 6th (2^7) bit at 1. Furthermore, because these two bits are now used for network addresses, we have actually created a total of four subnets: 156.28.243.0/26, 156.28.243.128/26, 156.28.243.64/26 and 156.28.243.192/26. The number of hosts for each network is the number of bits left in the 32-bit address space of the IPv4, that is 6 bits or 64 devices with host address parts between 0 and 63 [FuL06].

Earlier we shortly mentioned that the growing size of a routing table is an increasing problem as well. CIDR partially helps combat the problem with a mechanism called prefix aggregation. In this aggregation system, if some part of the network address parts match, they can all be stored in the routing table grouped under the matching part of the prefix. E.g. if fifteen network addresses with the network address part length of 24 bits share 20 of those bits, they all can be stored as one entry with a network prefix of /20 [FuL06].

What CIDR enables is for IANA and other authorities in charge of handing out IP addresses to hand out smaller blocks of IP addresses and in that way combat the problem of IPv4 address space exhaustion. The other solution is the emergence of the IPv6 addresses, where addresses are comprised in the same way, but the address space is 128 bits. How subnetting works is important for the purposes of our thesis, since it deepens our understanding of the structure of networks and how it affects routing.

2.1.3 Delivery semantics (unicast, broadcast, multicast, anycast, geocast)

Thus far, when we have spoken about routing, it has been in the context of routing one message or datagram to one recipient based on its network location. This is the most basic scenario of routing and is actually called unicasting, one of several addressing methodologies used in the internet. When talking about addressing methodologies, it is important to keep in mind that what we are discussing is how to select recipients for messages and how to route the messages to them [Com13, KuR08].

Broadcasting, by contrast to unicasting, is the act of sending one message to all devices in some network or subnet. It does not differ greatly from unicast, the source device sends a packet to a networks so-called broadcast address, which has been decided to be the all-ones of the host part [Mog84, Kur08]. That means that for any network, the broadcast address is the address where all bits left for the host part of the IP address are set as ones. So, for example a 192.168.0.0/24 network, the broadcast address would be 192.168.0.255. Upon receiving a message for the broadcast address, the router will replicate the packet for all devices connected to the destination network.

Multicasting is a method of sending from one host to many receivers, using a so-called multicast address or group address. Since an IP packet header still needs to contain only one destination address, a special group address is constructed, that interested clients then specifically join using Internet Group Management Protocol (IGMP) [Com13]. Multicasting scales pretty well, since the source device only needs to send one packet and the necessary replication for all the needed recipients is done by the packet-forwarding devices. For routing purposes, multicasting determines the *direction* of the packet flow by taking a look at the source IP address, which should be a simple unicast address, and routing the packet away from that, towards the appropriate networks where there are devices that have joined the network using IGMP [Com13, Dee88].

Anycast is an addressing method where multiple servers are assigned the same network IP and advertise that address to routers. When a client wishes to send a message to that IP address, the routers compare possible routes against their attached metrics values. Most usually the packet is then forwarded towards the network with the lowest cost attached to it, meaning in IP network routing the network which can be reached with the least amount of hops. Anycast is mainly used to provide increased quality of service for decentralized services, most well-known example being the current Domain Name System (DNS) [Har02].

Last common addressing method is the method of geocasting. Geocasting requires specific geo routers, whose job it is to determine if a packet destined for a specific geographic location is within its service area. A geocast packet's destination is defined either as a point, a circle with a center point and a radius, or a polygon defined through a list of points. Geo routers define their own coverage areas as a union of all the geographic areas covered by the networks attached to the router. The routers share this information amongst themselves and build routing tables from that information [Rüh09].

2.1.4 Autonomous systems

A big part of the current routing schemes is the concept of autonomous systems (AS). As the name suggests, they are self-governed collections of connected network addresses that define a common routing policy to the internet. An AS number can be requested from an Internet Service Provider (ISP).

The autonomous system serves as a kind of encapsulation towards the outside world. For the purposes of the outside world, they only see the routing policy of the ISP and don't really have to care about the internal workings of the AS. The AS, by contrast, can have its own routing policies within its own domain [Com13]. The term domain here refers to the grouped networks governed by the entity in charge of the AS.

Autonomous systems have also played a big part in the development of some of the features of today's internet and network services. For example, in order to provide global connectivity without laying down wildly redundant physical networking, ISPs sometimes loan or buy a part of the communication bandwidth of a part of network owned by their competitor. Understandably this carries a cost for the buyer, and therefore they want to avoid utilizing their competitor's network wherever possible. Through autonomous systems and the ability to define their own routing protocols and routing metrics, the ISP is able to assign a cost in their routing metrics based on who owns the network the route would connect to, or some other metric like the cost of bandwidth, latency, etc.

2.2 Routing protocols

The act of routing or packet forwarding should not be understood as a protocol for routing. The act of a router forwarding some packets is the execution of a well-defined algorithm, like we have seen in previous chapters. However, a routing protocol actually means the logic of constructing the routing tables for routers. It has to deal with the act of routers announcing their routes within a network and collecting received information of other routes and filtering them into a complete and most optimal routing table [Com13].

In this chapter we will discuss different models of routing protocols and where and how are they used and implemented. In this chapter, the role of an AS will also become more prominent when we discuss the need for different types of routing protocols depending on the routing needs in relation to autonomous systems.

2.2.1 Routing metrics

We have already talked about how a metric value is attached to each route in the routing table. In a scenario where multiple possible routes are found in the routing table, the option with the lowest metric value is chosen [Com13]. Most usually this metric is a representation of the path length, meaning the number of hops along the path to the destination network, but other simple possibilities also exist. For example, the metric might represent the maximum transfer unit (MTU) of the path to destination. MTU is the maximum size of a packet the network can transfer, so it directly dictates the number of packets a message has to be split into. The greater the number of packets, understandably the longer it takes for the message to be completely transferred to its destination. Therefore, if the MTU for a particular route is really low – meaning the message needs to be split into a greater number of packets – the cost for that route will be set higher to reflect the greater time needed for the message to travel [Kur08, Com13]. In addition to hop count or MTU, other simple metrics might include latency, bandwidth or the path's reliability, meaning how much packet loss has been experienced on average on that particular route.

Due to the costs of maintaining the physical aspects of networking, it stands to reason that ownership is split between multiple ISPs, each having built a part of the network as part of their business. The business of ISPs is to rent out usage of their network to customers, and this applies to personal customers and enterprises alike. What's noteworthy is that ISPs will, in order to offer their own customers the best possible service and connectivity, rent usage of their competitor's network as well. Naturally the ISP renting out its part of the network will want to bill its competitor according to their usage. This is possible, because the IP address blocks the ISP has been assigned are public knowledge. The renting ISP, in turn, will want to avoid using competitor networks when their customer tries to reach a network reachable from the ISP's own network with a reasonable metric. They can assign a modifier to any metric based on how many competitor networks the route passes through, and how much will they be billed in each. The metric of any route is usually not based on just the bill accrued, but it is also possible if an ISP wishes to always avoid using competitor networks.

2.2.2 EGP and IGP

How routing tables are constructed is a matter of choice, made by the network administrators. There are multiple different options available, but before choosing the used algorithm, one must first understand the position and role of the router in question. There are two main uses for routing protocols: intra-AS and inter-AS, that is, the routing done within an autonomous system and the routing done between autonomous systems, respectively [Com13]. For the former purpose, the group of routing protocols are called interior gateway protocols (IGP) and the latter group is called exterior gateway protocols (EGP).

For the IGP family, there are two main ways of approaching the problem. In one scenario, the routers do not have the full picture of the network topology, in the other they do. The former type of routing protocols are called distance-vector routing protocols, while the latter are link-state routing protocols [Com13].

If a router is not allowed to have a full picture of the network topology, then it must receive information about paths in some other manner. In the distance-vector routing protocol most commonly used, the Routing Information Protocol (RIP) the path information is advertised by routers to their neighbours, who are then able to update their own routing tables based on this new information. In the first iteration of the RIP algorithm, this advertisement was done via broadcasts every 30 seconds, but when network sizes began to grow, it quickly became apparent that an advertisement cycle once every 30 seconds would be a heavy burden on the network, especially considering that not every router would even participate on the RIP routing. Therefore, in version 2, the advertisement is done as multicast messages [Com13, Mal98].

In link-state routing protocols, the routers will have complete information of the whole network before they start constructing their routing tables. The information is spread via advertising; each router will send a message to every one of their neighbours, identifying the router itself and all of its neighbours, excluding the one the advertisement was just sent to. These advertisements carry with them a sequence number, which is used to identify the most current information at a receiving node [Com13, ClJ03]. The routers will then construct a routing graph for themselves using some form of Dijkstra's algorithm. An example of link-state routing protocol is the Open Shortest Path First (OSPF) algorithm, which instructs the routers to build their routing table using round-trip time, data throughput, link availability and reliability as metrics for each route [Com13].

When performing routing between autonomous systems, the routing protocol most often used is the Border Gateway Protocol (BGP). A BGP-router receives its neighbouring ASes through manual configuration, after which it will advertise these learned routes to its peers and the IGP-routers within its own AS so that they know of the networks reachable outside of their own AS. This data is also propagated through the AS to possible other BGP-routers on another edge of the network, with different neighbours configured for it. BGP, operating with multiple ASes, has the capacity to take into account multiple different metrics, including the cost of using a specific route through a competitor's network by taking into account its cost in the form of billing [Com13, RLH06].

2.3 Summary

In this chapter we have taken a look at the most important features and aspects of the traditional IP network routing scheme. The section serves as important groundwork for the reader to familiarize himself with the most fundamental of concepts related to traffic forwarding in the internet.

Throughout this thesis it is important to keep in mind, that even though we do not explicitly bring up the IP network when talking about message forwarding in e.g. overlay networks, all overlay messaging is eventually similar IP packets than all other traffic.

3 Traffic analysis

As we have seen in the previous chapters, a lot of the routing information is made public for simplicity. The source and destination IP addresses are public for every router on the way to check, the destination port is public and barring the use of encryption like using the HTTPS protocol, even the contents can be read by the routers.

It is no surprise then, that the data readily available by reading packet headers is utilized by different actors. This act of gathering information from messages is called traffic analysis and is inherently not a bad or unethical process, even though nowadays the term has gathered quite a notorious reputation. Traffic analysis can be, and is, used for network planning, problem detection and usage reporting. For example, we have talked about how different ISPs bill one another for usage of their network. This billing would not be possible without traffic analysis, because the packets originating from a competitor's network are revealed as such because of the source IP address. All packets are inspected and the ones that need to be billed separately from a competitor are tallied for later use [MYH05].

Today's challenges in traffic analysis have to deal with the huge amount of traffic going through the networks in each given moment, and the fact that with the emergence of several P2P platforms, streaming software, online gaming etc., accurate identification has become increasingly more difficult [EMA06, MYH05]. In this chapter we will take a look at what the act of traffic analysis entails and what are the possible approaches used when trying to figure out what are the packets flowing through a given network. Lastly, when these methods are more familiar to us, we will also discuss what this act means for user privacy and the assumed right of anonymity on the internet for all users.

3.1 Traffic classification

In today's internet, traffic classification is done for a lot of other reasons other than simple billing between ISPs. Numerous network administrators want to know what type of traffic goes through their networks for reporting purposes, data collection etc.. Often the publicly available IP packet headers are not enough to identify the traffic, due to the increasing number of protocols and platforms build on top of the normal TCP/IP stack. Trying to identify traffic using only this publicly available information will usually lead to 50-70% accuracy [MoZ05]. However, in 2005, network administrators reported that for example peer-to-peer (P2P) traffic constituted over 50% of their total traffic, which is not usually identifiable based on just the IP packet headers [MYH05].

In this chapter we will discuss the most common traffic classification techniques currently in use. We will focus on the software-based classification techniques, since they are more prevalent and widely deployed. The more traditional techniques, port number and payload analysis, are discussed and shown to be still valid and applicable techniques but ultimately not fully equipped to catch and reliably identify the multitudes of new applications emerging nowadays, especially in the field of P2P traffic [EMA06]. To supplement these traditional techniques, machine learning techniques have been implemented in the field and with good results.

Machine learning techniques are traditionally split into two different sets of algorithms:

Either the supervised or unsupervised approach [EMA06]. Machine learning in itself is at its core a set of algorithms that are learning from example data. This means that the data itself specifies what is a good or right answer to the questions pertaining to the same dataset. Mentioning the dataset previously, both styles of machine learning require some sort of training set to get it going, but the difference between supervised and unsupervised come in the form of the dataset. For a supervised machine learning algorithm, the dataset needs to be classified or labelled. Each data point tells the learning algorithm what are its most meaningful characteristics and values. From there on, the algorithm can with pretty good probability recognize similar new data as belonging to existing classes. An example of a supervised machine learning algorithm is a Naïve Bayesian classifier [MoZ05].

On the other hand, an unsupervised algorithm, while still requiring the starting dataset as learning material, does not require the data to be classified as anything from the start. The algorithm runs over the given data and tries to cluster similar data together. In other words, it will invent classes based on the data it gets and assign the starting set and all new data according to those classes in the future. Additionally, it will fine-tune its classifications based on new information it receives when analysing new data, meaning the classes are not static and often are prone to change.

Hardware-based traffic classification exists as well. It requires specialized machines installed at an operator's server room, close to the network routers. Although the machines are reliable, they are often really expensive and work better as a traffic measurement tool rather than traffic classification tool, since they can be outfitted with traffic classification algorithms, but they would need to be regularly updated as well [Wil01].

No matter the chosen approach, to enumerate a few of the challenges a traffic classification algorithm has to overcome: first, many applications utilize application-layer protocols that might be encrypted; applications use port numbers that are not IANA-assigned and therefore not easily traceable; many new applications, most notable streaming applications, can utilize multiple sessions to transfer different sets of data. In the case of streaming services, one session might transfer control- and other metadata and another session the multimedia data itself. These sessions can also use different transmission control protocols, especially video streaming is traditionally done with the help of user datagram protocol (UDP) [KWH05].

3.1.1 Port number analysis

Traditionally, the Internet Assigned Numbers Authority (IANA) has assigned set port numbers for different protocols to use, and is then subsequently used by a lot of traditional applications [KWH05, EMA06]. This set of well-defined port numbers has made it easy to track applications using those ports historically and is the basis of the port number analysis technique. For example, normal HTTP traffic uses the port 80, and HTTPS uses the port 443 by default and email uses the port 25. When traffic utilizes one of these well-defined port numbers, it is easy to also classify and identify the application in charge of that traffic. Until the emergence of a lot of today's streaming and gaming platforms, this approach was moderately successful in classifying traffic, but recently it has been shown to achieve less than 70% accuracy [KCF08].

The problems traffic classification based on port number analysis has to face is that many applications nowadays don't use the IANA-assigned port numbers, either because they want to hide their identity or for some other reason. They can either use ports of other applications or assign port numbers dynamically per session and deliberately avoid the set of well-defined port numbers. Especially P2P file sharing applications tend to hide their identity from classification, while media streaming platforms utilize many simultaneous sessions, sometimes sharing a port number [MoZ05, KCF08].

Bottom line is, if the IP packet does not specify a port number in the headers that is also present in IANA port number list, then this approach is not applicable and therefore port number analysis is most prominently a statistic on how much traditional traffic is being routed through the network. By traditional we here mean traffic that obeys the standard port numbers. From the above, it follows that the accuracy of port number analysis is inversely proportional to the amount of P2P traffic in the network, but when a link primarily has to deal with traditional applications such as WWW, DNS, Mailing, FTP and SSH flows, this approach can still be highly successful, reaching accuracies of up to 90% and more [KCF08].

3.1.2 Payload-based analysis

Out of all traffic classification methods, payload-based analysis of packet traffic is the one with potentially the most negative impact on user privacy. In this approach, the packet flow is searched for characteristic signatures of known applications. Signatures here mean specific contents of packets or some combination of headers – like source and destination ports – and content. It is based on the fair assumption that applications have similar communication flows each time they are communicating with another application or a client [EMA06, KCF08]. Once the algorithm has been supplied with these known signatures of applications, it will perform extremely well in identifying those applications it has been taught about, even identifying applications otherwise known to be hard to classify like streaming and P2P traffic [KWH05, EMA06].

The aforementioned privacy concerns naturally rise from the fact that a network operator has the power to read contents of packets flowing through his network. While other analysis mechanics avoid using flow-specific data, payload analysis is wholly based on reading and analysing specifically data specific to an instance of communication. This means that every single time payload analysis is performed on packets, it is specifically checking unique information about that communication session in the form of packet headers and its contents.

One of the main downsides of utilizing payload analysis is the fact that it is much more computationally heavy and requires more storage capacity, because those requirements are directly proportional to the amount of traffic the system wants to capture and inspect and that in turn implies greater cost for the analysing system. Other than that, the analysis system needs to have been pre-taught about the applications it is required to identify, meaning that new or otherwise previously unknown application traffic will still go unnoticed by a payload analysis system. Finally, because payload analysis is based on looking for signature characteristics of packet flows, encrypting your traffic understandably interferes with this method, because the flow will no longer match any known signatures. Therefore, protecting your traffic by using HTTPS communication wherever possible seems to be one easy and feasible way of achieving a decree of protection against this classification method [EMA06, KWH05, KCF08.

3.1.3 Machine learning based analysis

The basis for machine learning techniques is to conceptualize a communication session between two devices as a continuous, bi-directional flow of messages between the two endpoints. Because the session is based around some application, it stands to reason that the messages in said flow should share some characteristics, like packet length, number of packets in a message, destination port, etc. These machine learning approaches both rely on this assumption and strive to classify flows under analysis to one of pre-existing clusters the algorithm has formed based on its training data [MoZ05, EMA06].

For both supervised and unsupervised approach, the process for deploying them is similar. First, there is the clustering process, in which the algorithm is given the training data and forms clusters or classes based on common characteristics within that dataset, the execution of which varies depending on the type of clustering used [EMA06]. In this scenario, the dataset is an example set of message flows, either crafted or recorded real examples. For a supervised machine learning algorithm, the training data has been hand-classified has been shown to achieve accuracies of up to 95% [MoZ05]. By contrast, the unsupervised algorithm's dataset has not been touched in any way beforehand, and it performs the clustering by itself. Therefore, the unsupervised algorithm does not have any a priori understanding of the correct or true classification of applications. A good result of clustering process should be clusters in which entities within a single cluster share a lot of similarities, but entities between different clusters have almost nothing in common [EMA06].

3.2 Implications on privacy

One of the cornerstones of the internet has been widely thought to be the inherent anonymity. If you are someone with beliefs, ideas, opinions or data that the authorities would deem unacceptable, the internet has traditionally been the place where self-expression has been possible without fear of persecution. The aforementioned scenario is somewhat naïve for sure, but still very possible and reality in countries where human rights are not as strictly enforced. An easy example from 2016 is Russia's current trend to persecute homosexuals, which understandably leads to such individual's need to hide their identity when discussing the matter. China's Great Firewall is a concept in itself, aiming to filter any and all material that the government deems unfit to be seen by its citizens. Side effects of the Great Firewall and similar policies are imprisoned and house arrested reporters that have been found to have spread sensitive material that contradicts the official news outlet's versions about China's internal state. The great majority of these cases have roots in the government's ability to follow and track internet traffic through ASes within their own borders, enabling them to identify certain actors through packet inspection.

Of course, the whole internet quaked when in 2013 Edward Snowden revealed NSA's mass surveillance operation PRISM. PRISM represents an unprecedented level of surveillance on internet traffic, regardless of the origin of said traffic [GrM13] and understandably raises questions and concerns about any unencrypted traffic being subject to potentially malicious operators listening in. We bring up PRISM as one of the most blatant breaches of internet's anonymity to serve as proof that there is a real possibility of people in places of authority having an interest in secretly following users across the internet.

It can then be argued that the internet is not inherently anonymous and is not meant to be such. After all, IP packet headers are public and they specify IP addresses of both the destination and the source. Additionally, HTTP protocol is not encrypted by default, meaning it is made possible for other parties to listen in on conversations. Lastly, often repeated is the phrase that someone who has done nothing wrong has nothing to hide. And certainly, if we accept these premises, then there is no moral implications to discuss about traffic analysis and what is left is only the academic curiosity of what is the effect of traffic analysis techniques on an abstract thing called privacy. However, it is our belief that while privacy for sure is an abstract concept, by accepting the above premises we essentially deny the right to privacy altogether. Privacy, and the right to thereof, means that you have the right to expect that your communications between any other person stays between you and the person you have chosen to communicate with. If there is always reasonable doubt of someone listening in on your conversations, you arguably never have any privacy or at least always act like it. This act of self-censorship is one of the most heinous results of continuous surveillance that has been realized, because we start modifying our own – completely legal – behaviour according to the belief that someone is listening to everything we say or do, even on the internet [MaT15, Gre14].

The essential aspect of privacy that traffic analysis places under jeopardy is source anonymity. What is meant by this is the ability for the sender of any message to expect that his or her message will not be traced and used to reveal the identity of the sender, and through identification the geographical location of the user is also easily found out [FJZ11]. Even though the aforementioned traffic analysis methods do not explicitly state that they can be used to this effect, it is easy to see that each of them can very easily be outfitted with such capabilities. Additionally, when traffic analysis is performed *en masse*, it is important for the reader to understand that the techniques mentioned above are not deployed separately, but more commonly in unison, enhanced with the ability to read source IP address, perhaps even the ability to connect that IP address to the internet connection currently holding it, and who owns that connection.

It is interesting to note that until Snowden's reveal of the PRISM and other mass surveillance endeavours by the American government, there has not been a chance to gather academic data about the effects of surveillance on the large public. It should be expected, that a revelation of that magnitude would be strictly followed by the same public, because it is expected that people are interested in their own privacy. To some extent this turns out to be true, we know that people tend to start censoring their own searches and online behaviours directly after such a huge exposure, but at the same time it is also noted that continuous exposure around the same subject seems to suffer some diminishing returns on their popularity. Generally it seems that big reveals on surveillance do not seem to garner much interest and do not hold it for long [Pre15].

3.3 Summary

In this chapter, we have taken a look at traffic analysis and its different methodologies. We have elaborated on why IP traffic is inherently easily identifiable and traceable, and what might be the motivations for any entity to do traffic analysis. At the same time, it is important to note that some of the analysis techniques, particularly port analysis, has suffered some falling out because the myriad of different applications and protocols using non-standardized ports make it more difficult to categorize traffic based on used port numbers alone.

After getting familiar with the various normal traffic analysis techniques we have discussed how these might be harmful for a person's anonymity. Even though traffic analysis is not in any way illegal, and indeed has its roots in the effort of quality of service and maintaining the health of a network, we have pointed out that there are multiple scenarios in which an internet-user would want to keep his or her identity secret, for fear of any kind of persecution, that is put in jeopardy by traffic analysis, either done for monetary or political purposes. The current trend of increased surveillance, as we have shown, also begets and cultivates a culture of self-censorship, meaning that a lot of people might not find the help or answers they use the internet for in the first place.

4 Anonymous overlay networks

We have already given an outline on overlay networks and defined them as machines being linked together through a virtual or logical link, running on top of another network, either another overlay or the IP network. The overlay network is most typically an application-level network, meaning that the host computers are running some form of software that enables them to access said overlay. By extension then, an anonymous overlay network has all the same characteristics of a normal overlay network, but additionally it strives for user privacy or anonymity through various different means.

At the same time as we are talking about anonymous communication or anonymous overlay networks, it should be kept in mind that rarely is the aim of an anonymous overlay network to offer anonymous communication in the sense that the communicating parties do not and can not know or identify each other. The option to identify yourself via communication is always present and is actually usually even encouraged in order to establish trust between communicating persons [GRS96].

The presented motivations for building these networks vary, but seem to revolve around the same desire: the desire for stronger privacy, be it in the form of more secure communication, stronger validation of your communication partner's identity, preventing others from listening in on your conversations or making sure that you can share material with other people without any parties finding out where or from whom the material originates from [BGT02, CMH02, GRS96].

There are naturally a lot of different motivations to develop and implement an anonymous overlay network. In the context of anonymous overlay networks, anonymity can be mostly defined in two different scenarios. Anonymity for communication participants we have already covered and defined as source anonymity before [FJZ11]. The other purpose that's common for an anonymous overlay network is to enable users to share data without it being possible to track the origin of the data to a person, namely file sharing.

In the past, most anonymous networks have been developed for the latter purpose, to facilitate anonymous file sharing. But with the rise of increased network surveillance, it has become more popular to use a network which hides your communication and traffic altogether. It is somewhat ironic, that the increased network surveillance is in part due to the illegal side effect of anonymous file sharing – piracy – which the copyright industry wants to keep in check by increased network surveillance.

In this section, we will be covering three different anonymous overlay networks and outlining the methods of how they work and how do they attempt to secure privacy for their users. Since the view we have chosen for this thesis is the way traffic analysis -based privacy problems are handled in different anonymous overlay networks, and since a case study of a Tor network node utilization is a part of this thesis, some special focus has been given to the onion routing paradigm and the Tor network in general.

4.1 Routing in an overlay network

Generally, routing in an overlay network follows a pattern akin to routing in the IP network: if a node has direct knowledge of where in the network a message's recipient is, it will direct the message straight to it. Otherwise, it will attempt to pass the message on towards the correct destination.

Unlike the IP network, overlays are usually not divided into different ASes or subnets, but instead their routing tables are constructed using some form of optimization based on the knowledge of the overlay network's structure [CDG02]. For example, in a distributed hash table (DHT), a type of overlay network designed for easy data input and retrieval, implementation named Chord, the keyspace is thought of looping from the maximum id number back to the minimum id number in the keyspace. Each node's routing table, called finger table in the implementation lingo, is the size of the number of bits in the keyspace. The i^{th} entry in the finger table for node *n* has the closest node's address, whose key is closest to the value of $n + 2^{i-1}$ in the keyspace chosen. This optimization is possible because of the conceptually circular nature of the keyspace, and leads to O(log n) lookup time [SMK01]. Figure 1 shows an example of how the logarithmic search nature leads to skipping a lot of nodes on each early step, until the message is closer and closer to the intended target and the intended recipient key is close to a known node in a node's finger table.

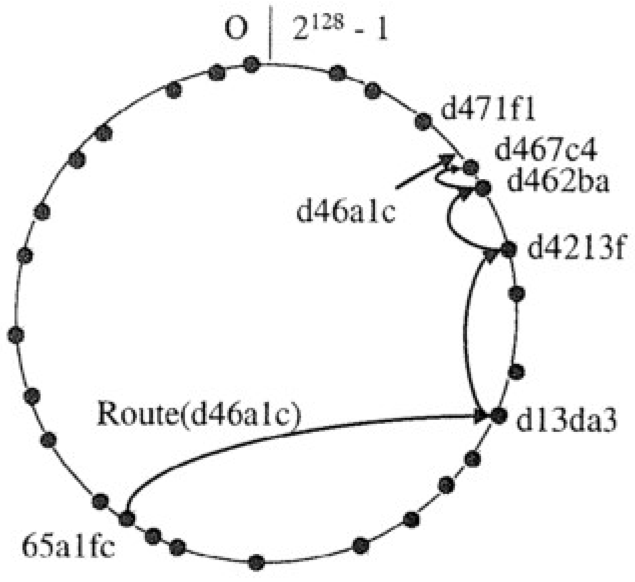

Figure 1. Example of routing in Chord. Node 65a1fc routing a message to d46a1c [CDG02]. Dots on the circle are routing nodes.

Other routing implementations of course exist, but the one presented in Chord is something that is pretty common and exemplifies the idea that due to the exact knowledge of how the network is constructed, it allows the designers to come up with an efficient routing algorithm. Also of note, that since the overlay network runs on the application level, it does not care, and indeed is not supposed to care, how the message is actually routed to its destination on the network layer. Of course that's something that will happen, because IP packet forwarding is still the way messages are actually routed to destinations. However, ignoring that allows the designers to more elegantly design their overlays, and by paying attention to proper encryption and other techniques, it is possible to achieve the wanted level of privacy that is the purpose of our anonymous overlay networks.

4.2 Freenet

The chronologically first anonymous overlay network in our sample set, Freenet is a file-storage application that utilizes free disk space on its client's hard drives that the users decide to allocate to Freenet's use. The authors of Freenet deemed it important to categorize Freenet as a file-storage application rather than a file-sharing application, since one of the core functionalities of Freenet is to replicate inserted data to multiple nodes in order for it to remain accessible even in the case of the original host of the data becoming unavailable. In a normal file-sharing application the case often is that if the file has not yet been downloaded by other nodes, and the original host then goes offline, the file will be unavailable even though demand for it would suddenly emerge. Freenet tries to counteract this by copying the file to multiple nodes initially on file insertion [CMH02, CSW00].

In this section we will take a look at how Freenet functions, and how are some of the basic functions like storing and retrieving a file made possible. Afterwards we will reflect this against the methods of traffic analysis and find out, if there is some sort of method we can use to break Freenet's anonymity.

4.2.1 Overview of functionality

In order for a node to join the network, it must first generate itself a public and a private key. The public key will act as the node's identity within the network, which will work even in the case of a physical address change for the node. The new node has to have knowledge about another, existing node within the network in order for it to be able to send a notification about itself. This knowledge might be gained through some node listing on the internet or other out-of-bands communication. The information notification contains the new node's public key and physical address, which the receiving node will propagate through the network until the TTL on the notification runs out. At that point, each node that has received the notification will participate in forming a random SHA-1 GUID for the new node. The GUID will also dictate the new node's responsibility area within the Freenet keyspace, and using the SHA-1 algorithm all but guarantees that no collisions for objects occur [CMH02].

In order for a user to insert data into the system, he will first have to generate key-hash for the file by hashing a short descriptive string about the file. This key-hash is then considered an insert proposal and sent to the user's own node, accompanied by a TTL value. When a node receives the insertion proposal, it just needs to check its own storage to see if the same key already exists. If no collision is detected, the node routes the notification to a node within its routing table with the GUID closest to the hash value of the file that is going to be inserted. This process is performed until the insertion notifications TTL runs out, at which point, if no collisions have been seen, the insertion is approved and the file can be inserted into the network. In this case, the initiator then sends the file down the same routing path as with the insert notification. In the case that the key already exists, the insertion is denied, because there has been a key collision. Additionally, the original file will be returned with the decline message, making an attack in which a malicious node floods the network with files, infeasible. Along the path of insertion, each node verifies the data against its GUID, stores it and creates a routing table entry associating the data source, who will be named in the insert-message, with the new key. To prevent the obvious security problem, each node can randomly decide to change the insert-messages data source identifier to point to either itself or to another randomly assigned node [CSW00]. If a node cannot forward the insert message to its preferred recipient, it will backtrack to the second-closest GUID within the routing table, and so on.

When retrieving data from the network, the process is almost equivalent to the insertion of data. The query initiator will hash the description of the wanted file, and send it to a node in its routing table. If the key matches a key in the node's local storage, the file is returned along with an indication that it was the source of the data. If the node does not locally possess the file, it will forward the request along to a node with a GUID as close to the file's identifier as is possible with the node's routing table. This process will be repeated until the request's TTL runs out or the file is found and subsequently returned to the original seeker. If the file is found somewhere along the search path, it might get stored locally to some nodes along the search path. As with the insertion, nodes have the option to decide to swap the data source identifier of the return message to point to themselves [CMH02, CSW00]. It should be noted, that this operation does not impede any further queries for the same file, because the node who decided to change the data source identifier still retains the real data source in its own local routing table [CMH02].

Freenet also utilizes signed subspaces for some form of data management. Signed-subspace keys (SSKs) are set up to function as kind of folders that should contain only similar data within them. Because these subspaces need to have their own public and private keys generated for them, every member of the network can read the contents of the subspace, but only the owner of the subspace can write to it. By contrast to normal file insertion, inserting a file to a subspace requires the node to hash the concatenation of the subspace's public key's hash and the hash of the file's description string, then signing it with the subspace's private key. Similarly, retrieving a file from a subspace requires only the public key of the SSK and the descriptive string of the file in question [CMH02].

Through the routing of messages in Freenet, a couple of interesting effects can be identified. First, because queries are forwarded to nodes that have GUIDs similar to the key being inserted or sought after, they will continue to get queries for similar keys. Because of this, they will also store and cache those files more often, over time becoming highly specialized in returning content with specific keys, leading to better performance in queries over time. Second, a node's connectivity increases over time. This is due to the fact that as node's process returning requests, they create new entries in their routing tables, indicating the suppliers of files, thus increasing connectivity for themselves. This is helpful for a new node to gain more insight about the network, but in order for the node itself to be found more often, it needs to insert files or indicate itself as a data source in requests it processes [CSW00].

Due to users sometimes wanting to hide the file's contents itself from prying eyes, it is also possible to encrypt the file's content before inserting it to Freenet. In that case

4.2.2 Effectiveness against traffic analysis

Looking at traditional traffic analysis methods, they seem to offer some valid data about Freenet. Through payload analysis, given that the contents are not necessarily encrypted, an observer might gain insight on what is the data being transferred within the network. Because of local caching of files, however, it is not certain that two parties communicating and transferring files are actually the original requester and original owner of the file in question. Of course these techniques are dependent on the fact that the observer puts up a node within the network, but given that Freenet is open for joining nodes, that will not be an issue for any actor.

Because of the file-sharing nature of Freenet though, a malicious observer is probably mostly interested in finding out the sources of certain files and the requesters for those files. On the subject of finding out the source of a particular file, it would prove extremely difficult, because after a successful insert of a file into the network, the file is actually already hosted by multiple nodes, and the tracking of the original one would not be possible. However, when the file is being inserted, the notification we have talked about earlier is basically just another query with an original source. An attack has been devised that is able to trace a request back to its originator, called the traceback attack [TDB03]. We furthermore note that due to the way requests are processed in Freenet, we can count every node that has seen a particular request, as communicating with the source node. Additionally, because of the requirement for a node to announce its physical address when joining the network, and the address in essence being an IP address, real-world identification for identified originators is a very real possibility.

The traceback attack hinges on the fact that each request in Freenet has a particular UID value attached to them. These UIDs or transaction IDs are not guaranteed to be unique, they are randomly-generated 64-bit identifiers. But because these IDs are only stored at the nodes forwarding or processing them for as long as the processing is still unfinished, they are only important for as long as the query is traversing the network [CSW00, TDB03].

The first step in the attack is to attach a malicious node *m* to a suspect node *n* in Freenet. The problem is that a node in Freenet is allowed 40 neighbours in its routing table, and they are based on how close their GUID is to the node's own and how many requests the node has served. If *n* already has the maximum amount of neighbours, it is allowed to exchange the least recently used node in favour of a new node, if the new node's neighbours have served at least a minimum number of content requests [TDB03]. This is apparently to ensure that the new node has gathered enough information about the network to be an effective part of it. In the event that *m* is not accepted as a neighbour for *n*, and given that the GUID for *n* is known, we can choose to insert a large number of files so that their keys cover a sufficiently large area around the suspect node *n* [TDB03]. Then a different malicious node *a* can be used to request these files, which *m* will serve to *a*, until the minimum number requirement is satisfied. After this, a new attack node *b* can be announced, with a GUID close to both *n* and *m*, who now has neighbours who have served a number of content requests to this point. If this new node is still not accepted as a neighbour for *n*, we can repeat this process until some subsequent node becomes a neighbour for *n*.

After having attached a malicious node to a suspect node, by way of sending special probe messages, used for debugging purposes, with the UID of an interesting content request (interesting here denoting interest in the part of an attacker), and an invalid destination location, which usually identifies the intended recipient for the debugging information, the malicious node can create a message that the suspect node will answer with a specific error in case of a conflicting UID, but will not propagate to other nodes within the network, because the invalid destination location renders the probe message invalid for processing [TBD03]. The malicious node will then wait for a content request message of interest by trying to match the routing keys of requests to the predetermined routing keys of interest to it. When such a request is detected, the node will gain the content request message itself and the set of neighbouring nodes for the suspect node *n* to determine who might have seen the same content request. By following the same protocol as above, the attacking party can then start identifying the whole set of nodes that have seen the same request.

To identify the originator of the request from the whole set of nodes that have seen the request is actually a challenge, because Freenet nodes do not associate any sort of timestamp with the UIDs, so it is not clear in which order nodes have seen the request. Figures 2, 3 and 4 describe some different scenarios on how the same content request might travel between two nodes A and B in the set of nodes that have seen the request. In figure 2, B has seen the request because A has directly forwarded the request to him. However, in figure 3 we see that the path might also become convoluted in the sense that the request has backtracked from B back to A, possibly after B had forwarded the query to other nodes first. Moreover, figure 4 illustrates the case where A and B have had no direct contact with each other. The problem becomes apparent when considering the scenario in which A and B are neighbours, but A hasn't forwarded the request to B, but rather B has seen and stored the UID because some other node X has forwarded the query to it. The traceback is done through the reverse path, that is the path the query has travelled down through the network walked in reverse. Problems arise in scenarios like the one depicted in figure 4, where the traceback path is non-linear, meaning that there are at least two different forks in the path, which would be the case in figure 4, given that A and B are also neighbours. However, it turns out that Freenet has a tendency to produce linear reverse paths [TBD03].

The originating machine can be found utilizing some lemmas about the Freenet net-working. First, when a reverse path exists between nodes *a* and *b* and the path is of length two, then b must always be the originator. Second, on a linear reverse path, a backtrack can occur only once on a corresponding forwarding path, meaning that the same path cannot contain more than one occurrence of a backtrack. Third, given a linear

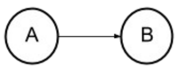

Figure 2. A forwards the content request to B.

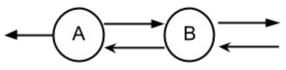

Figure 3. A forwards the request to B, but the request backtracks.

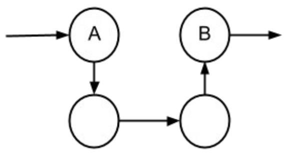

Figure 4. No direct communication happens between A and B.

reverse path $n_m \rightarrow \ldots \rightarrow n_{k+1} \rightarrow n_k \rightarrow n_{k-1} \rightarrow \ldots \rightarrow n_0$ it stands that n_k cannot have initiated a traceback to n_{k-1} if the distance from n_{k+1} to n_m is less than the distance from n_{k-1} to n_m, meaning that on the original forwarding path the distance to n_0, even though backtracking would occur, is still getting smaller by each hop. Fourth lemma states that if for every node n_k on a linear reverse path of $n_m \rightarrow \ldots \rightarrow n_{k+1} \rightarrow n_k \rightarrow n_{k-1} \rightarrow \ldots \rightarrow n_0$ it holds that the distance between n_{k+1} and n_0 is less than the distance between n_{k-1} and n_0, then n_0 must be the originator of the request. Finally, if a linear reverse path $n_m \rightarrow \ldots \rightarrow n_{k+1} \rightarrow n_k \rightarrow n_{k-1} \rightarrow \ldots \rightarrow n_1 \rightarrow n_0$ exists, n_0 is the originator of the request, if n_1 has at least one neighbour n who has not seen the UID of the message and it is true that the distance to n_m is less from the node n than it is from n_0 [TBD03]. Put together, these lemmas enable a malicious observer to identify a request's originator with certainty, although the process is somewhat cumbersome and requires the attacker to put up multiple observation nodes, which could end up being a somewhat lengthy process for even one node.

4.3 GNUnet

One of the examples of anonymous file sharing, and building on Freenet's lessons learned, GNUnet attempts to offer anonymity by making sure that no node within the network knows exactly what they are serving. This is achieved by splitting all data into small, encrypted chunks that are spread over the network and are searchable via queries that incorporate natural language to some degree [BGT02]. Searches pertain to keys that have been defined for the data upon insertion by the inserting user. Distributing file chunks across the whole network also helps with overall load per node, even though it means that retrieving complete data requires cooperation from a multitude of nodes.

The goals of GNUnet are the deniability of participants, fault-tolerant distribution of content and efficient usage of bandwidth and storage space of participating nodes. Here, deniability of participants mean that each node can claim they are not in possession of some piece of data but are merely routing it. The creators of GNUnet have additionally decided to set efficient usage of bandwidth as a goal, because the design of GNUnet means that reconstruction of data that has been input into the network requires a lot of traffic.

4.3.1 Overview of functionality

When joining the network, each node chooses a pair of keys, one for identification and authentication of the node itself, the other for encrypting the communication between nodes. In order to hide which nodes are communicating with each other, the nodes use a MIX-like approach: nodes are treated as intermediaries that send messages to other nodes, but are not allowed to know the contents [Cha81, Kug03].

When data is first inserted into the system, it is split into 1 kilobit (kb) blocks, because GNUnet operates in fixed size files optimized for the transfer protocol of choice, UDP, when compulsory protocol headers are added to the payload the size of 1 kb. The user provides arbitrary keys K, that can be derived from natural language, plus a description of the content [BGT02]. Data is divided into three different block types, the Data Block (DBlock), Indirection Blocks (IBlocks) and Root Blocks (RBlocks). The file under insertion becomes split into multiple DBlocks, to which IBlocks then point as a kind of instruction on how to reconstruct the original file, and finally the RBlocks are the searchable root blocks containing descriptions and metadata of the file [Kug03].

When a file has been split into DBlocks D_i, they are hashed with a one-way hashing function, yielding $H(D_i)$. Then an IBlock-tree is formed recursively, in which every block contains knowledge of how to find the blocks under it in the tree, which will eventually lead into the DBlocks. This information is called a query hash. In addition to the query hashes, each IBlock will also contain superhash, a hash of the concatenation of all query hashes below this IBlock in the tree structure [Kug03]. Every block B_i, including $H(D_i)$ mentioned earlier, is encrypted with its own content's hash value, which yields then $E_{H(B_i)}(B_i)$ and is stored under the name of $H(E_{H(B_i)}(B_i))$. Therefore, one query hash within an IBlock is a pair of $(H(B_i), H(E_{H(B_i)}(B_i)))$. As the final step, the RBlock is generated. The RBlock, as stated before, will contain the metadata, such as file length, data's description, a checksum and a signature value. For every K_j the user has provided, the RBlock R will be encrypted to $E_{H(K_j)}(R)$. Both the encrypted RBlock and $H(H(K_j))$ are then stored under the name of $H(H(H(K_j)))$ [Kug03, BGT02].

The splitting of files is done in order to make sure that no node has to bear unnecessar - ily high burdens in terms of bandwidth or storage space. It is hard to find a node willing to spend lots of storage space to store files that are the size of mega- or gigabytes. Even if they would be willing to spend the storage space, seeing as how cheap storage is nowadays, the worse problem would be encountered at the moment someone would want to download that file. Keeping in mind that GNUnet nodes mostly run in people's private networks, having to commit the bandwidth necessary to transmit a file of that size would certainly deter possible network participants.

Retrieving content that has been inserted into the network utilizes the knowledge of how and under what name the data is stored. We already know that the name the data is stored under is the block's data hashed twice. Because the hash function used should be a one-way function, the nodes transmitting the data cannot obtain knowledge of the data they are transmitting [BGT02, Kug03]. The first step in retrieving data from GNUnet is to discover the RBlocks pertaining to the file, by forming a search query by keywords K_j that are triplehashed to form $H(H(H(K_j)))$. If a node has the RBlock stored under that hash function, it will respond with the encrypted RBlock $E_{H(K_j)}(R)$ and the $H(H(K_j))$ as proof that the node has the RBlock stored under the specific keyword. Note that the node that issued the query is able to decrypt the response by using $H(K_j)$. No other node is able to get knowledge of the data being transmitted except through knowledge of the specific keyword K, or by being in possession of the RBlock and being able to thus match $E_{H(K_j)}(R)$. Although theoretically a dictionary attack could then be possible to quess the value of K, the threat can be mitigated by properly choosing K and basically treating it as the same as a password for the data [Kug03, BGT02]. Querying by boolean queries of the form a AND b is also possible, leading to only those RBlocks that match both keywords to be returned. Storage usage is not impacted by a lot, since this only leads to one extra RBlock per keyword being generated [BGT02].

After the RBlock has been found, the searching node can issue a download order for the root IBlock using the query hash within the RBlock. Decrypting the root IBlock will yield query hashes of the other IBlocks that are also then downloadable and so on, until finally the DBlocks are downloaded and decrypted using the information within the IB-locks.

We now see that there are two different types of queries within the GNUnet: search and download queries. Both of these query types contain additional values that can be considered as headers or metadata for the query itself. These are the return address, the priority of the query and a time-to-live (TTL) value. Each node operates its own, separate message queues for every single neighbouring node it has. When a query has to be sent to one of those nodes, it is instead stored in the queue for that neighbour. Each queue is flushed at random intervals, meaning that all the messages in that queue are sent to the node the queue represents [Kug03].

When a query is received, the node first checks whether the requested block is found locally. If this is the case, a response of the form outlined before is put into the queue corresponding to the return address in the query. If the query was a search query, and if the content was not locally available, the query is sent to other neighbouring nodes by randomly selecting n neighbours, where the size of n is dependant on the priority of the query. The query's priority is then adjusted to be $\frac{p}{n+1}$ where p is the old priority of the message and n is the number of nodes selected to receive the message next. Additionally, the node can exchange the return address to be its own address, in which case it will store the query to its own routing table, making it able to route the message backwards to its source when a response arrives [Kug03].

The anonymous nature of GNUnet comes from the network giving them deniability for each query. Each node within GNUnet has the ability to claim indirection for any query, meaning they can claim that they are not the original source of the query, but rather they have simply exchanged their own address for the return address of the query in question, like it was described previously. Figures 5 and 6 describe two scenarios in which a query is first not indirected by the node B and then is indirected, respectively. Claiming indirection understandably offers anonymity to each node within the network, since all nodes can claim indirection when faced with any queries about the data it has supposedly been downloading. Indirection in GNUnet is based on node's current load and a random factor in order to throw off any guesses about whether indirection has happened [BGT02].

However, like it was stated before, nodes are not required and do not necessarily always indirect queries they receive. This leads to a serious revelation. If a node N never indirects any queries, then any and all queries that have the return address of N are sure to be originated from N and thus there is no deniability for N. From this we come to understand that in order for a node to achieve deniability and thus anonymity within GNUnet it has to indirect at least some sufficient number of queries [BGT02].

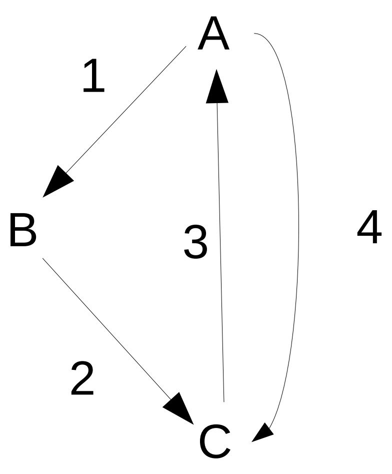

Figure 5. No indirection happens at B.

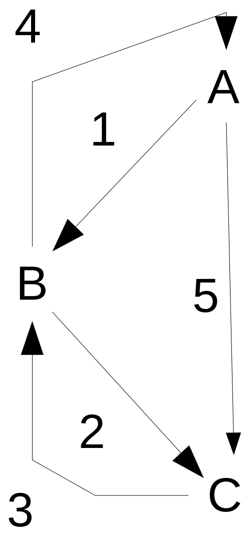

Figure 6. B indirects the request from A.

4.3.2 Effectiveness against traffic analysis

In the case of GNUnet, traffic analysis will require nodes within the network. We will assign them the mission of gaining as much knowledge as possible about the network and the nodes within it, and then see what methods are possible for them to gain this knowledge. Since we know that GNUnet is a file-sharing network and run on application-level, we can discount port analysis techniques because they would not reveal anything, because ports used by GNUnet are not assigned in the IANA lists.

Due to strong encryption of content, and the deniability GNUnet offers to its nodes, machine learning techniques employed on an observer outside of the network would not suffice either, because even though the IP packets would reveal the destination and source addresses of GNUnet nodes, indirection offers the level of anonymity in which no outside observer can be certain about whether an IP packet from A to B actually means A and B are communicating with each other in any meaningful way, or whether in reality another node C wants to communicate with B and A just happened to be the node to reroute the query C sent.

In order for an agent to gain any tactile information about GNUnet's network and its nodes, it would have to have its own node join the network and attempt to somehow gain information about the data moving through there. We call these kinds of nodes that are in the network only to gain information about the network and its participants against their will malicious.

The first step that a malicious node needs to take, is to gain knowledge of some of the content within the network. Due to encryption of everything moving through every node, it is impossible to get information about the data being moved without being able to decrypt the blocks somehow. By having the malicious node query the network for a set of data pertaining to predefined, interesting keywords, the node gains knowledge of the encrypted RBlocks, and is able to decrypt them, and thus can also download all the IBlocks and DBlocks. So far nothing malicious has been done, the malicious node has only used GNUnet in the manner it was meant to be used, as a file sharing network, but the node has now gained the ability to observe all queries containing one of the prepared keywords, all queries for any of the I- or DBlocks it has itself queried for, and any and all responses for those I- or DBlocks, because it has itself calculated the hashes for those keywords and blocks and is thus able to decrypt any messages that contain those blocks or keywords [Kug03, THS05]. From this it follows, that any nodes within GNUnet that do not turn up in the monitored data traffic, can be eliminated from the anonymity set [BPS01]. So only by downloading block data and monitoring the GNUnet traffic for similar data traffic, we can already start limiting the anonymity of nodes.

A method dubbed “The Shortcut Attack” utilizes the fact that even though there is some random variance in when queries get indirected, the process is still somewhat deterministic. When a node has a lot of resources to spend, it will indirect a query, and forward it when it is sufficiently stressed. Queries can also get dropped if a node is extremely busy. This behaviour is what is called the shortcut and it is not supposed to hurt anonymity of other nodes in the network, since we have just shown that a node can only hurt its own anonymity by never indirecting queries. In short, the shortcut attack aims to make a node forward queries instead of indirecting them in order for a malicious node to gain knowledge of the query's initiator [Kug03, THS05]. This is achieved by flooding a node with high priority queries, thus increasing the node's load and making it favour forwarding queries as opposed to indirecting them. By forwarding queries to the malicious node, the node is exposing the initiator of the query to the malicious node doing traffic analysis [Kug03].

The weakness of this shortcut attack is that the attacker has no way of making sure, what queries are being forwarded. Keeping in mind that the node under attack might be connected to at least hundreds of other nodes, the possibility of the malicious node getting a query he's interested in getting forwarded to him, is pretty small. The attack method would be more efficient if routes in the network were more static, but since they are not, the need to flood the attacked node with queries needs to be removed [Kug03].

In the improved shortcut attack the malicious node will approximate that a node A is close to the initiator than the node B. In this case, even if the malicious node receives an indirected query from B, that originates from A, the malicious node can simply respond straight away to A. If the malicious node has quessed correctly, it is providing valid content to A and possibly attracting even more queries straight from A itself. This possibility is even increased in the event that B is unable to respond to A's query without the help of the malicious node, which is never going to come [Kug03]. Now, if A is the initiator, the attack has successfully determined who is communicating queries outward. Of course, this improvement on the shortcut attack requires some knowledge of the network's topology, but when that knowledge is present, the malicious node can find the initiator by next connecting to A's neighbours and waiting for the content it is interested in. By way of statistical testing it can then determine whether a new node is closer to the initiator or whether it is not, which indicates A as the initiator of the query. With the basic routing mechanism of GNUnet – that is, random – the lower bound for this type of attack to find the initiator of the query can be defined to be equal to

$m \geq \frac{1}{P_{rnd}^{2}} + \sum_{i=1}^{l} \frac{1}{P_{rnd}^{i}}$, where P_{rnd} is the equal probability for a node to be selected as the target to forward or indirect a query to, m is the selected set of nodes from k neighbours and l is the distance of the attacker to the initiator [THS05, Kug03].

4.4 Onion routing

When aiming for anonymous connections in the sense that no other party has any knowledge of who is communicating with whom, a technique called onion routing has been regarded as a successful technique. In it, an initiating node, the first node contacted by the sender of the message, determines a route to the recipient node through the network's different nodes and forms an onion that it sends through that route in order to establish a circuit, a kind of semi-permanent communication channel between the two endpoints. The term onion refers to the object being sent, because it is encrypted with layers, with the outmost layer being decryptable by the next receiving node, and the payload being the next encrypted layer and the next node's id in the network [GRS96, GRS99, SGR97]. Only for the receiver's node will the payload be the actual message. From this it follows that after the initiator node, a normal forwarding node only has information about the immediately preceding node and the immediately following node. The process of choosing nodes into the circuit based on their routing to the destination is left out of this chapter, since that process is more implementation-specific and does not have a general model like the process of onion routing does.

In case a malicious agent tries to determine information from the length of the remaining route, a random string the size of the layer being peeled off is added as padding to the payload for the next node at each hop. Alternatively, the padding can be performed for each layer of the onion at the initial node so that for each receiving node, the payload has the size of a full onion, as though the node was the first one to receive it. This method is based on the fact that onions are defined as having constant size [GRS96].

Excluding the details of the actual payload's format, a layer of the onion should have at least this information contained within it: Expiration time, next hop id, cryptographic operations and keys to be performed for the payload when messages are being routed from the original sender to receiver, cryptographic operations and keys to be performed for the payload when messages are being routed from the original receiver to the sender, and the payload in some format. For every routing node, the payload will look like another complete-sized onion, and for the intended receiver the payload will be the actual message with the padding we mentioned earlier [GRS96, GRS99]. The expiration timer is used as a countermeasure for replay attacks, in which someone tries to gain information about the network and the communicators by sending the same onion again through the same circuit. A routing node needs to keep a copy of each onion it receives until the expiration time is met, so that if the node receives a copy of any onion it already has, it simply discards it. Same goes for onions that have expiry timers already met, those are discarded straight away as well [GRS96, SGR97].

It is also possible that the originating node does not know the full route to the recipient. It's not necessary either, and in some cases might even be discouraged in order to add more hops to the circuit's chain. In these cases, or in a case there is a connection change of which the initiator is unaware, a technique called loose routing is used [GRS96]. When using loose routing, a maximum loose routing count variable is added to the information contained in an onion's layer. The router that makes the decision to employ loose routing then acts as a kind of secondary initial node, making the received, part-way routed onion in its entirety a new payload, and forming a new onion with layers at most the amount specified in the maximum loose routing variable mentioned earlier. Of course, this could conflict with the demand for fixed size onions, so the original sender's node needs to take this into account when doing the initial padding and data operations to the onion [GRS96].

4.5 Tor network

Building on the onion routing paradigm introduced in the previous chapter, the Tor network (Tor) is a so-called second generation onion router. Being an anonymous overlay network, it too aims to conceal the identities of people participating in the network, but unlike our other examples which are primarily used for data storage and sharing, Tor is primed for concealing communication itself. That is, Tor specifically tries to provide source anonymity for communicating partners, making it impossible for listeners to find out, who is communicating with whom [DMS04].

When designing Tor, the developers have chosen several different design goals for the network. First, the system needs to be easily deployable, meaning that running a Tor node can not be too expensive for the user, nor be difficult to implement by requiring a lot of custom patching of your operating system. Second, because an anonymous system requires a solid user base in order for it to reliably hide user interactions, Tor needs to be easy to install and use on any operating system, and not require much configuration in order to get it working. It is important to note that the aforementioned large user base is not only a design goal, it is a security requirement as well [ADS03, BMS01]. Third, the platform needs to be independent enough to serve as a potential testbed for future research into open problems in low-latency anonymous networks, like protection from Sybil attacks. Fourth, the end product's design needs to be simple and well understood, in order for Tor to be extensible with additional features without imposing too much of a cost on readability and deployability [DMS04].

4.5.1 Overview of functionality

Like other anonymous overlay networks we have discussed in this section, Tor is a user-level application as well, running with no special privileges. When speaking about the functionality of the platform, two distinct terms need to be made clear from the start: Onion Proxies (OPs), are user-run local softwares that are their connection to the network, used to handle connections from other user applications, fetch directories etc. Onion Routers (ORs) are the processes operating within the network, relaying data and connecting to requested destinations [DMS04].

ORs maintain two keys: one for long-term identification, meaning that it is used to sign TLS-certificates for communicating between different ORs, and signing the OR's own descriptor, containing its keys, address, bandwidth and other such information. The other key is for handling the OR's onions, key negotiation and decrypting circuit requests that are incoming from different users. Additionally, as a part of the TLS communication protocol, a link key is generated, but these are rotated periodically and independently from other keys by the OR, in order to mitigate the damage a compromised key could generate [DMS04, DiA99].

When talking about communication in Tor, it is important to remember that the data transmission adheres to what was discussed in section 4.4 in general. When reading through this chapter, it is therefore important to note that even though we do not explicitly bring up the onion routing scheme or the act of circuit building, the same principles are still in effect.

Next, we are going to take a look at how a circuit is built in Tor. In that discussion we will omit the step wherein an OP decides on the exact nodes to which it will relay messages. Therefore, we deem it appropriate to at this point concisely discuss this fact. Tor uses some small number of known ORs that act as directory servers. These directory servers track network topology and node states, including their keys, and make this information available through their own HTTP servers. The directory servers share this information between themselves, and other ORs periodically publish their own information to these directory servers, in order for each directory server to have a uniform view of the network. What this means for OPs is that they can contact these directory servers in order to get knowledge of the network around them and start building a circuit [DMS04].

ORs communicate with 512 byte fixed-size cells. These cells consist of a header and a payload, where the header carries a circuit identifier, and a command that describes how the receiving OR should treat the cell payload. The circuit identifier is a some form of unique ID that corresponds to one circuit, which is necessary because a single TLS connection can carry data from several different circuits [DMS04]. Additionally, the identifier is connection specific, meaning that each OP/OR or OR/OR -hop carries its own circuit identifier, but nevertheless they will still correspond to a particular circuit. Cells can be either control cells or relay cells, based on the command provided in the cell's headers. Control cells are always intended and interpreted for the receiving node, and they carry one of possible commands: padding commands are used to keep a connection alive or pad a link; create or created commands are used for establishing a new circuit and acknowledging the creation, respectively; destroy, which is used to remove an existing circuit altogether.

By comparison, relay cells are used for end-to-end communication and are decorated with an additional relay header at the beginning the payload. The relay header fulfils a very similar purpose to the cell header, carrying stream identification, checksums for payload integrity checking, payload length information and a relay command. The possible values of a relay command are: relay data, which is used for moving data from the sender to the recipient down the communication stream; relay begin or end which instructs the nodes to begin or shut down a stream, respectively; relay teardown is used when relay end is not possible, meaning the stream has broken and needs to be shut down; relay connected is used to notify the OP that a relay begin command has successfully completed; relay extend and extended are used to bring another node into the communication circuit and to acknowledge the completion of such an operation; relay truncate and truncated are then used correspondingly to drop a part of the circuit's participating nodes; relay sendme, which is used for congestion control; and finally relay drop, used to implement long-range dummy messages [DMS04].

To begin constructing a circuit, a user's OP starts by sending a create command cell to the first chosen node on the path. The payload of the create cell is an encrypted handshake, the key used being the onion key of the receiving OR. The OR then responds with the created command cell, with a payload that contains the negotiated key that is based on the initial handshake. The initiating user now has a circuit with the length of one hop [DMS04].

In order to get a bigger circuit, the OP issues a relay cell with the command relay extend to the first OR. For this relay cell, the payload is a new encrypted handshake and the identity of the next OR. The first receiving OR then issues a create command cell to the OR specified in the initial message, and passes the encrypted handshake along to the control cell. The created response control cell from the newest OR in the circuit is then wrapped to the relay extended relay cell by the first OR and passed back along the circuit to the OP. Every subsequent extension to the circuit follows the same method [DMS04, SGR97]. In the end, the original user has constructed a circuit with perfect unilateral entity authentication, because the OP knows for whom the handshake is made, but since no public key signing is used, the ORs have no knowledge of from where the handshake actually originates. Furthermore, because the encryption key is decided between the first OR and the OP, unilateral key authentication is also established [DMS04, SGR97].

When the OP wants to communicate with another OP, it first needs to select a circuit to use the communication for. It can either choose one of the pre existing ones, or then construct a new one altogether, and then designate one of the nodes on the circuit to be the exit node. The exit node is a crucial node in the circuit, since it is the only one that will gain knowledge of the intended recipient of the message being transmitted, and it can be chosen to be any node within the circuit, though usually the last one [DMS04]. When the circuit and the exit node have been selected, the initiator OP will send a relay begin cell to the exit node, specifying the recipient OP in the payload. Upon the exit node successfully connecting to the recipient, the exit node will send a relay connected notification cell to the OP, and the stream is then ready to transmit data, which is packaged into relay data cells. Closing a stream is analogous closing a TCP stream, the OP sends a relay end cell and gets a relay end cell in response, after which the stream has successfully been closed. In the event of an error or a broken stream, the adjacent node which notices the problem will send a relay teardown cell [DMS04].

4.5.2 Effectiveness against traffic analysis

Like in other anonymous overlay networks, it is not feasible to consider scenarios, where an attacker or adversary has global control over the whole overlay network. It is easy to understand that when somebody has total control over the entirety of a network, that entity can trivially get all the knowledge required to identify all the traffic in the network. Therefore, Tor aims to provide protection against an entity that can gain partial control of the network [DMS04, MDM05].

In Tor’s model of routing data, one OR is of significant value when considering ramifications for anonymity. The exit node from a circuit is always in a special position, because it gains the knowledge of the intended recipient for a particular message. Therefore, if the exit node happens to be malicious, it gains at least some knowledge of the communicating partners. Remember, that because Tor is a volunteer-network, it is trivial for a malicious user to install the required software and join the network with multiple nodes. Even though we cannot assume that this same user will have control over all of the Tor network, we can for the purposes of this analysis assume that he can plausibly have a node in any position of the network or a particular circuit.

Attacks against Tor can be categorized into two different types, which are analogous to the traffic analysis types we have discussed earlier. A traffic analysis attack can be either passive or active. Passive attacks against Tor's anonymity usually mean that malicious nodes gather information about communication patterns and try to single out relevant data from the patterns or usage of certain circuits. However, because several streams can share one circuit, and due to cell contents being encrypted, passive attacks will be hard to enact and would require a lot of processing of encrypted stream data, trying to find the similarities in one OPs outbound traffic and another's inbound traffic over a long period of time [DMS04, LLY09].

By contrast, active attacks focus on watermarking the traffic, somehow making it distinguishable from all other communication flowing through the network. An example of such an attack could be an OR that slightly changes the timing interval of cells when transmitting them forward, and then attempting to correlate that same interval on the receiving end [LLY09].

The above watermarking technique can be used to achieve some form of traffic identification by a modestly equipped attacker. In one effective form of attack, the attacking entity needs a tor node of his own as well as a web server under his control, serving a website somebody connects to. The prerequisites are directly analogous to a situation where a Tor user wants to anonymously connect to some website being controlled by a corporation or some other entity that has an interest in breaking that anonymity and gain information about users connecting to their site.

In the attack, the attacker configures his web server to send packets in a predetermined pattern. The pattern itself is going to be the watermark that he will attempt to follow through the network. The corrupt tor node will be used for tracking. It is possible to configure your own node to construct circuits that are only one hop in length, meaning that they only connect to one other node. By configuring the attacker's node in this way, he gains the ability to follow any one particular node, by flooding the only other node in the circuit with probe traffic that measures the latency of the connection, and therefore gives an estimate on the load of the target Tor node [MDM04]. So the attacker will therefore send data in a particular pattern from his malicious web server, then flood a target Tor node with probe traffic to get information on the latency the probe traffic receives, then try to match that latency pattern with the particular sending pattern of the web server. This attack pattern seems to be reliable and pervasive and clearly breaches some of the anonymity in Tor, because it degrades the anonymity of a node into being basically just a normal proxy server [MDM04].

4.6 Summary

This section has provided some insight into anonymous overlay networks. In the introductory part, we briefly discussed some of the more common ways of routing messages or data along an overlay network, Chord was introduced as an example here. We then introduced some basic functionality of three somewhat popular anonymous overlay networks, GNUnet, Freenet and the Tor network. For each of these networks we showed how they try to ensure anonymity for their users, which was mainly through different encryption schemes and mechanisms for obfuscating the traffic path so that an observer would not be able to tell the real originator of a given message. However, we also found out that there are ways for a persistent attacker to gain some information about the network's users on every platform. Of course, academic papers and the realities of traffic analysis do not necessarily meet at each turn, so even though a network's anonymity model might get breached under certain conditions, it does not automatically mean that the anonymity of all of a network's users is in peril.

On a larger scale, traffic analysis and user identification on the internet is a very large area of different technologies, methods and attacks on a lot of different platforms. Even though an anonymous overlay network might tackle some of the problems and offer an increased level of anonymity against traffic analysis techniques, there are still ways in which a person might get identified, for example through canvassing techniques [Nor16].

5 Case study: Utilization of a Tor relay node

At the moment, one of the foremost anonymous networks on the tongue of people seems to be the Tor network. The Onionview service lists the number of all Tor relays at several thousand all around the globe, which at least heavily suggests that Tor as a project's support base is somewhat wide, keeping in mind that the number of relays does not reveal anything about the number of Onion Proxies, meaning the amount of people using these relays to more anonymously surf the internet [Oni16]. Hidden services, which are sites accessible only from within the Tor network, and their popularity also suggests that even Tor is widely and somewhat regularly used [BPT14].

Because of Tor's perceived popularity, we believed that running a Tor relay and logging the amount of traffic through it would give some insight to the amount of users and their geographical distribution. In this section we will be going through the setup of our tor relay and take a look at the gathered results.

5.1 Setup

We ran the tor relay on a Raspberry Pi, first model type B+, with 512 Megabytes of RAM and a Broadcom ARM processor BCM2835 with integrated GPU. The choice of hardware was based on the empirical knowledge of a Raspberry's stability, and the fact that running a Tor relay was not going to heavily burden the processor or RAM, especially with proper throttling of the relay's bandwidth. Network connectivity was handled through the author's own home internet connection, with a static IP allocation from the ISP.

The operating system installed on the Raspberry was the Raspbian Jessie, a Debian-based linux operating system. The Tor relay package is currently available and was installed through Debian's package manager apt-get, as well as arm, the command-line monitoring and logging tool for the relay itself. Excluding dependencies for the afore-mentioned packages, no other software was installed for the purposes of this data gathering case study.

On the configuration side of the relay, we set it up to advertise a bandwidth of 819,2 kilobytes per second, with a burst bandwidth of 1 megabyte per second. Because the computer was run off the author's own internet connection, which was still in use daily for other purposes as well, the case study was not to interfere unnecessarily with normal life. At the same time, almost a megabyte worth of bandwidth nets the relay a “fast” attribute in the atlas.torproject.org online directory, which in turn increases the amount of clients that want to connect to our relay, meaning that the advertised bandwidth should be enough to guarantee meaningful amounts of traffic.

For every relay, there is the option to specify in the configurations that the relay will also work as an exit node. What it means is that an OP is allowed to designate that node in the circuit as the one that will submit the message to the recipient OP. However, we opted out of this possibility, even though it would offer us insights on what kind of services Tor users might connect to. The reason is, that if the traffic was malicious or illegal, and it was being followed, the traffic would seem to be originating from the author's personal connection, and it could possibly lead to legal ramifications, which was not something we would be willing to risk.

For logging, we deemed that using the logging offered by Tor itself was going to be sufficient. After all, we scoped this as a case study and therefore were not aiming at doing any special inspections of our relay's traffic. Tor offers the ability to log on three different levels of severity, debug, notice and error. Debug logs would log everything the relay was doing, and in so doing expose some aspects of our relay that a malicious entity could perhaps use for its own purposes. The notice level offers the level of logging we felt was the most useful: Through it, the software writes periodically a so-called heartbeat information to disk, containing the uptime, amount of circuits the relay is a part of, the amount of traffic sent and received, and the number of handshakes for circuits received and successfully completed. Tor network currently uses two different protocols for handshakes, the TAP and Ntor protocols, which are both logged separately. In a separate file, the software would track connections by country, based on the geoIP of a packet. These written logs were then backed up through rsyncing to an external hard drive in case the Raspberry Pi would suffer some form of breakdown or otherwise lose the logs.

5.2 Data collection phase

Generally speaking, four different major variables can be identified from the data collected. Those are the number of circuits the relay is operating in, how many handshakes the relay has received, how much data has been transferred and what is the geographical distribution of the connections.

We did not have a lot of set expectations for the case study, mainly because the case study was conducted without much prior knowledge about the inner workings of Tor. Starting the case study, we assumed that the relay would receive and send data somewhere around 500 gigabytes during the data collection phase. Additionally, due to Tor essentially being a peer-to-peer network and torrent networks quickly garnering connections from all over the world, we assumed that such would be the case here as well, and that our data would quickly show that connections had been made from almost every country in the world.

During the data collection phase, there were two unfortunate complications. Firstly, on the 8th of March, the Raspberry suffered a power outage for the day. No data for that day was therefore accumulated and the datapoint closely matches the one from the 7th of March.

Additionally, there is an undocumented feature that you are unable to turn off in Tor software's logging, which causes the program to rotate and overwrite heartbeat logs on a 10-day cycle. Since our redundancy was only for hardware failure and not an unforeseen feature like that, this resulted in the loss of all collected heartbeat data from February to the start of April. The heartbeat logs included the cumulative transferred data for the uptime period, handshakes and circuits for the uptime period. This feature affected the data collection in that data gathering was turned off on the 17th of April and the absence of data was not noticed and consequently logging turned back on until the 19th, resulting in a gap from 17th of April to the 20th of April. However, the entry statistics were intact through the whole data collection period.

5.3 Results

Like previously stated, the Tor software by default offers us four distinct measures by which to gain insight into our relay's popularity and therefore allowing us to extrapolate the popularity of the network to some extent. We will be introducing those results one by one.

For building circuits, Tor sends specific handshake-messages to relays, which are equivalent to the create control messages specified in section 4.5.. These two different types of protocols are the TAP protocol and NTor protocol, of which the latter is the more modern version and the former is deprecated and mostly used by some bots operating within the network [How13]. Our node successfully served just about every handshake request, failing handshakes total being seven for the data collection period. Since this number is so low, it probably indicates that the handshake process was terminated by the client OPs before they could be processed properly by our node. Figure 7 shows the fluctuation of received handshakes during the data collection period, the complete collected data is available as attachment 1. In total, the number of completed handshakes per protocol is 116 983 for TAP protocol and 386 082 for the NTor protocol, out of a total of 116 985 and 386 087 respectively. Considering that on 2013 the number of Tor users was nearing four million, half a million handshakes processed in the year 2016 for a somewhat slow relay in Finland seems to be on par with expectations [How13].

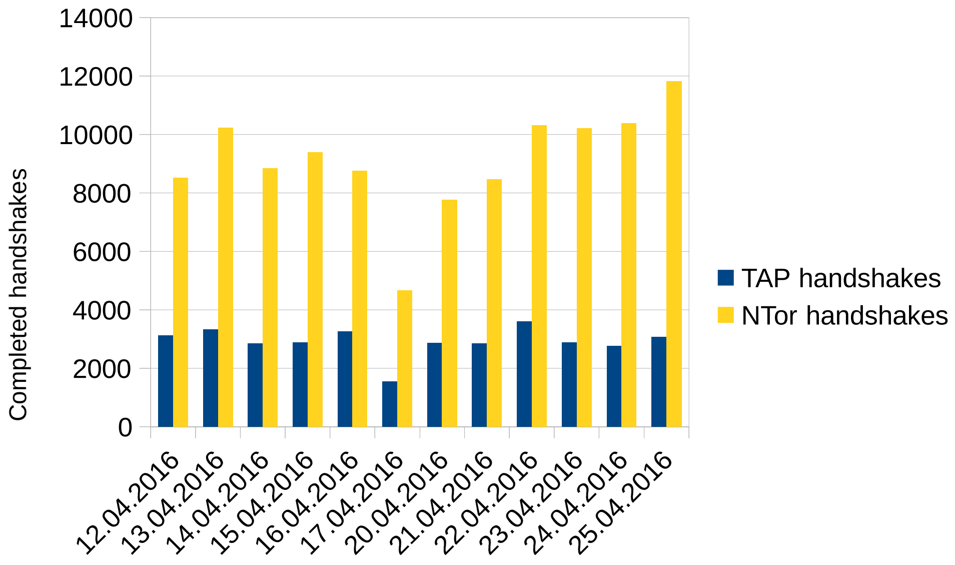

Figure 7. Number of completed handshakes in the two different protocols.

Figure 7 shows the number of completed handshakes, which essentially is the amount of circuit creation requests our node has received. An OP has multiple circuits from which it selects one with sufficiently small amount of load at the time to send a message through. Circuits can be torn down at the behest of OPs at basically any time, meaning that the number of circuits a relay is a part of will never be as high as the amount of completed handshakes. However, based on the number of handshakes, it would stand to reason that active circuits should number in somewhere in the early thousands at the very leasts.

In actuality, the number of circuits our node was handling simultaneously seems to revolve around the 300 circuits mark. Figure 8 shows that the maximum number of circuits achieved has been 400, and the minimum 186. The complete data is available as attachment 2. The low number of circuits compared to the extremely high number of handshakes does seem to indicate that either there is a great number of circuit teardowns happening all the time as well. However, because all the source material has led us to

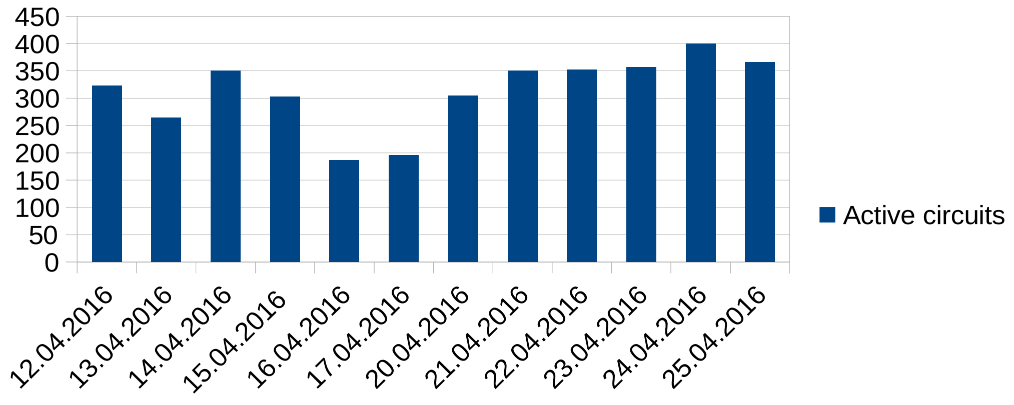

Figure 8. Number of active circuits at the time of writing heartbeat information to disk.

believe that circuits are at least semi-valuable to form due to the encryptions required, it would stand to reason that they were not torn down at a very high rate. Teardowns might happen as a result of the OP rotating circuits due to a specific encryption chain having been in existence for some predetermined amount of time and thus having been deemed as more vulnerable to malicious attacks. Computer resources could also be a substantial variable in the amount of circuits our relay can simultaneously be a part of, but we have not gathered any data on system load during this case study and therefore cannot theorize on this further with any substantial evidence to back us up.

We have now seen that our somewhat meagre relay has received a huge number of handshakes and served at best as a part of almost half a thousand circuits. The next important and interesting measurement is no doubt the amount of data sent through those circuits. Unfortunately Tor does not keep track of traffic per circuit, as it would be a new piece of information a malicious entity might use to distinguish nodes in circuits. The amount of traffic sent and received is cumulative for the entirety of the node's uptime, meaning in our case that our traffic amount is accumulated from 8th of April forwards. By the end of April 25th's surveillance period our node had sent 621.33 gigabytes of data and received 619.59 gigabytes. The increase in data can be seen in figure 9, while the complete collected data can be found as attachment 3.

What is perhaps most interesting in this data is that it allows us to extrapolate somewhat crudely, what might the total amount of traffic be, calculated from the start of our case study, February. From the data we can see that between a single six-hour period, the amount of traffic the node has sent increases by roughly 3 gigabytes. That means that in a 24-hour period our node sends around 12 gigabytes of data through the Tor network. Taking into account that this accumulation started on the 9th of March, 48 days prior, and that on a rate of 12 GB per day it would have taken almost 52 days to get to the amount of sent data. This implies that the increase hasn't been entirely linear, but somewhere outside of our data there has been a burst of traffic. What is more, since our relay has been turned on the 11th of February, and therefore our uptime has been 74 days, excluding the day power was lost to the machine, We can very roughly estimate that our total bandwidth has been somewhere around the 1 Terabyte mark.

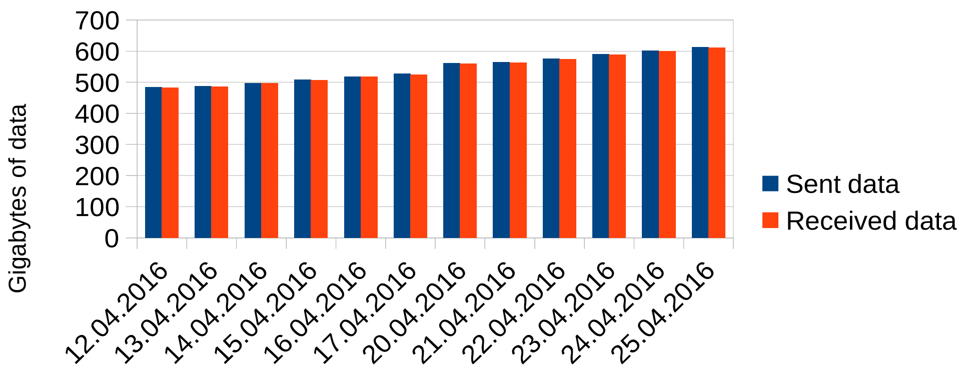

Figure 9. Accumulation of sent and received traffic over data collection period.

Finally, our data on the geographical spread of connections to our relay is set into perspective due to the insight into overall popularity of our node. All in all, it turns out that our relay processed connections from 58 countries geographically, from America to Japan. Total number of connections was 232 010, roughly equating to 46% of all processed handshakes in our relay. Daily distribution of connections can be found in attachment 4.

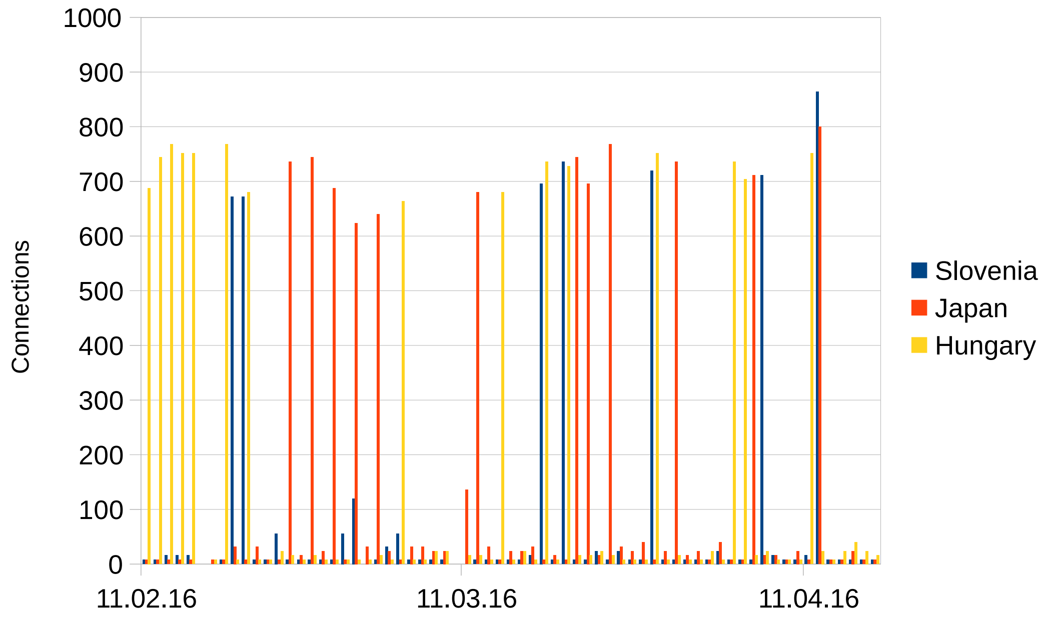

Figure 10. Connections from the top three countries by number of connections on a single day.

Figure 10 shows how the number of connections fluctuates for three countries that had the three highest amounts of connections on a single day. As is clearly evident, connections fluctuate highly on a daily basis, making it reasonably clear that OPs perform circuit teardowns far more frequently than what we assumed at first from the description of a circuit as a logical structure. More than that, there seems to be no pattern or reason behind the fact that the amount of connections fluctuates so wildly for a given country. What we can only verify from this is the fact that there does not seem to be much coherence behind the decision-making when an OP forms a new circuit, at least in the sense that it would reuse the same nodes often or frequently.

Turning our attention to the countries themselves, the most often connected country is Slovenia by 864 connections during the data collection period, followed by Japan at 800 connections and at third place both Hungary and Slovakia at 768 connections. Japan is a bit of surprise, as a somewhat liberal country you would not exactly expect there to be a need for anonymity services. Then again, Finland, a western liberal and democratic country also features on our connected countries list with 688 connections on the 15th of April. On the other hand, Slovenia, Slovakia and Hungary are all countries with historic roots and connections to Russia, and generally thought of as not as liberal and progressive as many other European countries. It is believable that in those countries citizens and for example journalists would have more reason to hide their internet usage patterns.

5.4 Summary

In this section we have introduced the case study in which we tried to ascertain the popularity of the Tor network. As such, we ran a Tor relay without exit node capabilities from February to April, but unfortunately due to various difficulties lost most of the gathered data, including the data amounts transferred, handshakes processed and circuits our relay had been a part of, from February to early April. However, the number of connections our relay has received remained intact and complete all the way from February to April and the end of our data collection period.

Our data showed that even with just a little bit of computing power and memory, and 1 megabyte of burst bandwidth, our relay constantly saw traffic, relaying in total over one terabyte from 12th to 25th of April as sent and received data. On average our relay had 3461 connections from up to 57 distinct countries, with some connections being from countries the logging system was unable to identify, on a daily basis. While the number of different countries is not as high as one would expect from a traditional peer-to-peer network, to us it seems evident that Tor as a network is popular and ready to utilize even meagre new resources that become available. In this sense, we deem our case study a success.

6 Conclusion

This thesis' core has dealt with the problem traffic analysis presents for anonymity, one of the cornerstones of the current internet paradigm, and anonymous overlay networks, one kind of answer for the same problem. The act of traffic analysis, even though originally meant for monitoring the network health and status for problems like congestion or failing hardware. However, due to largely commercial interests, the term now refers more to the act of identifying specific applications from data traffic, and using this knowledge to identify and track individual internet users. This presents a problem for people that have any reason to hide their identities. There is of course an ethical debate to be had about whether or not legally sound reasons for hiding one's identity trump the illegal or morally questionable ones, or whether commercial interests should be enough to start undermining this anonymity. The fact remains that traffic analysis can be and is largely used in todays internet, and that either as the intended or a side product, it is affecting the anonymity of internet users.

In order for the reader to understand how traffic analysis can be performed, it is first necessary to get acquainted with the mechanics of traditional IP network routing. Therefore, the first section of our thesis is aimed to show, how a network device will forward a packet from the sender to the recipient, starting from the routing table and how the device will refer to it and chooses the next target. Building on that, we then introduce other effecting factors, like Autonomous Systems and briefly touch the subject of prioritizing specific subnets or ASes over some others because of monetary reasons, namely the owner of the subnet using traffic analysis techniques to identify and bill certain traffic. After reading the section, we hope that the reader will have a solid understanding of how traffic flows through the network to its intended recipients and how this normal model for traffic might constitute a problem for anonymity.

After gaining sufficient knowledge of the normal packet forwarding procedure, the techniques traditionally used in traffic analysis are discussed. In nowadays internet, with all the emerging new protocols and applications that use non-standardized ports as their communication ports, it has become the norm for traffic analysis to be centred on analyzing and identifying certain traffic patterns to be belonging to certain type of traffic. However, this same technique can be used to single out specific users connecting to a site under surveillance. After introducing the different mechanisms of traffic analysis, we briefly discuss the implications on privacy, most notably the fact that after big reveals of breaches on anonymity, such as Edward Snowden's revelation, a lot of people start censoring their own behaviour on the internet, even though they might not be doing anything inherently illegal. In that sense we think we have clearly demonstrated the fact that fear of surveillance already affects all people, whether they have reason to fear or not. As such, the implication that traffic analysis has – in addition to an impact on anonymity – a negative effect on people's perception of freedom and behaviour.

One of the answers for the problem of traffic analysis is the development of anonymous overlay networks, application-level networks built on top of the IP network, implementing their own logic for routing and forwarding messages. We identified source anonymity as a sender's expectation that his or her act of messaging will remain anonymous in that the partners of the communication will not be revealed to outside entities and discussed how three different examples of anonymous overlay networks try to achieve this. We discussed GNUnet, Freenet and the Tor network, and found out that they each have a distinct aim and definition of anonymity in general. GNUnet and Freenet are mainly file-sharing platforms, but while GNUnet tries to secure anonymity by strong encryption strategies, while the developers of Freenet believe that by sufficiently obfuscating who is sharing which file, each user is effectively anonymous, because they can all claim that they are not the initiator of any specific request. Finally, the Tor network is more of a communications platform, which utilizes onion routing to achieve source anonymity, by ensuring that no node in charge of forwarding the message, apart from the very last one to forward the message to its intended recipient, knows the participants in any way.

We also took a look at what kind of traffic analysis attacks could be and have been utilized against every one of these anonymous overlay networks. Being distributed networks all, if a global attacker is assumed, then anonymity will surely break in every one of them. After all, if every other node you connect to in a network is malicious and wants to find out your identity, there is nothing a benevolent entity can do to hide himself. However, even excluding the global attacker option, it turns out that the anonymity in all of our example networks can be compromised to some degree with sufficient effort, resources and time. As such, none of our examples turned out to be the perfect solution, but in lieu of one, using an anonymous online network definitely provides a given user with more anonymity than using the “public” IP network.

Finally, as a case study, we have taken a look at the Tor network's utilization by running a Tor relay and gathering some data on how many connections it serves and how much data flows through it, as well as where does that traffic originate from. Remember, that a tor relay knows from where it receives a message and to where it is supposed to send it next. We found out, that Tor is in fact heavily used based on the amount of traffic seen in just under a month of data, and extrapolating that to the three month period of writing this thesis. Contrary to our expectation that connections would originate from all over the world as would be the case in a normal peer-to-peer network. However, it seems like connections clustered pretty heavily to some nations. Reasons for this could be innumerable. Most notably, China is not present in our list of origin countries at all, probably due to China's Great Firewall's attempt at blocking Tor as an application – again, through means of traffic analysis and identifying Tor's specific traffic patterns.

Taking a look at our two research questions for this thesis, we wanted to find out firstly whether GNUnet, Freenet or Tor provides effective means of protection against traffic analysis methods. For our first research question, the results seem to be inconclusive. Although they certainly introduce a new layer of security to protect user anonymity, we have also introduced ways in which traffic analysis methods can be used to breach that new security with some resources, malicious intent, and some finite amount of time.

Our second research question asked "Can Tor's perceived popularity be witnessed from the viewpoint of a single relay operator as high traffic volume and a large number of connections?" To verify or disqualify our research question, we ran a tor relay and logged various measurements on a very limited hardware with very limited bandwidth. Even though our limitations, the relay ended up receiving and sending in total approximately 1,2 terabytes of data within our collection period, with around half a million handshakes in the same period and couple of thousand of active connections daily from various locations around the globe, we feel that our second research question can be verified.

As a closing note, we would once again like to remind the reader, that traffic analysis, user surveillance and tracking on the Internet is a large problem domain, and not solvable just by one solution, such as the anonymous networks we have just discussed. In the future, work needs to surely continue on a strong anonymity network, but most likely it will be enhanced by something like a proxy solution incorporated into the network. Additionally, like always with the subject of network security, the best of mechanical solutions will not hold, if the user decides to give it up, i.e. by publicly identifying himself or allowing himself to be tracked. Such would be the case if the user would install tracking software on his computer, most probably due to carelessness or systematic baiting on the behalf of an attacker.

7 Acknowledgements

The author would like to extend a special acknowledgement and thank you for professor Jussi Kangasharju, for his mentoring and insight during the writing of this thesis. Without his advice this thesis would not have been written.

References

ADS03	Acquisti, A., Dingledine, R. and Syverson, P., On the economics of anonymity. Financial Cryptography, Springer-Verlag, LNCS 2742, 2003.
BGT02	Bennett, K., Grothoff, C. and Tzvetan, H., GNUNET – A truly anonymous networking infrastructure. Proceedings of the Privacy Enhancing Technologies Workshop (PET), 2002.
BMS01	Back, A., Möller, U. and Stiglic, A., Traffic analysis attacks and trade-offs in anonymity providing systems. Information Hiding (IH'01), Springer-Verlag, LNCS 2137, pp. 254-257, 2001.
BPS01	Berthold, O., Pfitzmann, A. and Standtke, R., The disadvantages of free MIX routes and how to overcome them. Designing Privacy Enhancing Technologies – International Workshop on Design Issues in Anonymity and Unobservability, 2001.
BPT14	Biryukov, A., Pustogarov, I., Thill, F. et al., Content and popularity analysis of Tor hidden services. 2014 IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 188 – 193, 2014.
CDG02	Castro, M., Druschel, P. and Ganesh, A., Secure routing for structured peer-to-peer overlay networks. Proceedings of the 5th symposium on Operating systems design and implementation OSDI 02, volume 36, pp. 299-314, 2002.
Cha81	Chaum, D., Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, volume 24, issue 2, 1981
ClJ03	Clausen, T. and Jacquet, P., Optimized Link State Routing Protocol (OLSR), RFC 3626, https://tools.ietf.org/html/rfc3626 [24.3.2016].
CMH02	Clarke, I., Miller, S. and Hong, T., Protecting free expression online with Freenet. IEEE internet computing, volume 6, issue 1, pp. 40-49, 2002.
CSW00	Clarke, I., Sandberg, O. and Wiley, B., Freenet: A distributed anonymous information storage retrieval and storage system. Designing privacy enchancing technologies lecture notes, volume 2009, pp. 46-66, 2000.
Com13	Comer, D. E. Internetworking with TCP/IP, Vol. 1: Principles, Protocols,

and Architecture, 6th ed., Pearson, 2013.

Dee89 Deering, S., Host Extensions for IP Multicasting, RFC 1112, 1989. https://tools.ietf.org/html/rfc1112 [15.3.2016].

DiA99 Dierks, T. and Allen, C., The TLS protocol, RFC 2246, 1999. https://tools.ietf.org/html/rfc2246 [13.4.2016].

DMS04 Dingledine, R., Mathewson, N. and Syverson, P., Tor: The second-generation onion router. SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium, 2004.

EMA06 Erman, J., Mahanti, A. and Arlitt, M., Internet Traffic Identification Using Machine Learning. IEEE Communications Society, 2006.

FJZ11 Fan, Y., Jiang, Y. and Zhu, H. Network Coding Based Privacy Preservation against Traffic Analysis in Multi-Hop Wireless Networks. IEEE Transactions on Wireless communications, volume 10, issue 3, pp. 2-28, 2011.

FuL06 Fuller, V. and Li, T., Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan, RFC standard 4632, 2006. https://tools.ietf.org/html/rfc4632 [15.3.2016].

Gre14 Greenwald, G. Glenn Greenwald: Why privacy matters, TEDtalk transcript, October 2014, http://www.ted.com/talks/glenn_greenwald_why_privacy_matters/transcript?language=en [27.3.2016].

GrM13 Greenwald, G. and MacAskill, E. NSA Prism program taps in to user data of Apple, Google and others. http://www.alleanzaperinternet.it/wp-content/uploads/2013/06/guardian.pdf [27.3.2016].

GRS96 Goldschlag, D., Reed, M. and Syverson, P., Hiding routing information. Workshop on information hiding, 1996.

GRS99 Goldschlag, D., Reed, M. and Syverson, P., Onion Routing for Anonymous and Private Internet Connections. Communications of the ACM, volume 42, issue 2, pp. 39-41, 1999.

Har02 Hardie, T., Distributing Authoritative Name Servers via Shared Unicast Addresses, RFC 3258, 2002. https://tools.ietf.org/html/rfc3258 [16.3.2016].

How13 How to handle millions of new Tor clients, Torproject blog, https://blog.torproject.org/blog/how-to-handle-millions-new-tor-clients [27.4.2016].

JGJ00 Jannotti, J., Gifford, D. and Johnson, K. Overcast: Reliable Multicasting with an Overlay Network. Proceedings of the 4th conference on Symposium on Operating System Design & Implementation, Vol. 4, 2000.

KCF08 Kim, H., Claffy, K., Fomenkov, M. et al. Internet Traffic Classification demystified: myths, caveats and the best practices. ACM CoNEXT, Volume 50, issue 4, pp. 1-12, 2008.

Kug03 Kügler, D., An analysis of GNUnet and the implications for anonymous, censorship-resistant networks. Privacy Enhancing Technologies lecture notes, volume 2760, pp. 161-176, 2003.

KuR08 Kurose, J. and Ross, K Computer networking, A top-down approach. Addison-Wesley, 2008.

KWH05 Kim, M., Won, Y. and Hong, J. Application-Level Traffic Monitoring and an Analysis on IP networks. ETRI Journal Vol. 27, Issue 1, February 2005.

LCP05 Lua, E., Crowcroft, J., Pias, M., et al. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys and Tutorials, Vol. 7, Issue 2, pp. 72-93, 2005.

LLY09 Ling, Z., Luo, J. and Yu, W. et al., A new cell counter based attack against Tor. Proceedings of the 16th ACM conference on Computer and communications security, pp. 578 – 589, 2009.

Mac15 MacAskill, E. The NSA's bulk metadata collection authority just expired. What now? http://www.theguardian.com/us-news/2015/nov/28/nsa-bulk-metadata-collection-expires-usa-freedom-act. [28.11.2015]

Mal98 Malkin, G. RIP Version 2, RFC 2453. https://tools.ietf.org/html/rfc2453 [23.3.2016].

MaT15 Marthews, A. and Tucker, C. Government Surveillance and Internet Search Behavior, 2015.

MDM05 Murdoch, S. and Danezis, G., Low-cost traffic analysis of Tor. 2005 IEEE Symposium on Security and Privacy (S&P'05), pp. 183 – 195, 2005.

Mog84 Mogul, J. Boardcasting internet datagrams, RFC 919, 1984. https://tools.ietf.org/html/rfc919 [15.3.2016].

Moy97	Moy, J. OSPF Version 2, RFC 2178. https://tools.ietf.org/html/rfc2178 [16.3.2016].
MoZ05	Moore, A. and Zuev, D., Internet Traffic Classification Using Bayesian Analysis Techniques. SIGMETRICS'05, pp. 50-60, 2005.
MYH05	Myung-Sup, K., Young, J. and Hong, J., Application-Level Traffic Monitoring and an Analysis on IP Networks. ETRI Journal, Volume 27, Issue 1, pp. 22-42, 2005.
Nor16	Norte, J. Advanced Tor browsing fingerprinting, http://jcarlosnorte.com/security/2016/03/06/advanced-tor-browser-fingerprinting.html [26.3.2016].
Oni16	The OnionView, https://onionview.com/?relaytype=0 [22.4.2016].
PfW86	Pfitzmann, A. and Waidner, M. Networks without user observability. Institut für Informatik IV, Germany, 1986.
Pre15	Preibusch, S. Privacy behaviors after Snowden. Communications of the ACM, volume 58, issue 5, pp. 48-55, 2015.
RLH06	Rekhter, Y., Li, T. and Hares, S., A Border Gateway Protocol 4 (BGP-4), RFC 4271, https://tools.ietf.org/html/rfc4271 [24.3.2016]
Rüh09	Rührup, S. Theory and Practice of Geographic Routing. University of Freiburg, Germany, 2009. http://archive.cone.informatik.uni-freiburg.de/people/ruehrup/georouting-chapter-draft.pdf [16.3.2016].
SGR97	Syverson, D., Goldschlag, M. and Reed, M., Anonymous connection and onion routing. IEEE Symposium on Security and privacy, volume 16, issue 4, 1997.
SMK01	Stoica, I., Morris, R., Karger, D. et al., Chord: A scalable peer-to-peer lookup service for internet applications. Proceedings of the 2001 conference on Applications, technologies, architectures and protocols for computer communications (SIGCOMM'01), pp. 149-160, 2001.
TDB03	Tian, G., Duan, Z. and Baumeister, T., A Traceback Attack on Freenet. IEEE Proceedings of INFOCOM'03, pp. 1797 – 1805, 2003.
THS05	Tatara, K., Hori, Y. and Sakurai, K., Query forwarding algorithm supporting initiator anonymity in GNUnet. Proceedings of the 2005 11th international conference on parallel and distributed systems (ICPADS'05), volume 2, pp.

235-239, 2005.

Wil01 Williamson, C. Internet Traffic Measurement. IEEE Internet Computing, 2001.

Wil11 Williams, M. Playstation Network Hack Timeline, 2011. http://www.p-cworld.com/article/226802/playstation_network_hack_timeline.html. [1.5.2011]

Attachment 1

date	TAP completed	TAP total	NTor completed	NTor total
12.04.2016 07:00	2127	2127	7825	7825
12.04.2016 13:00	2829	2829	8271	8271
12.04.2016 19:00	3123	3123	8519	8519
13.04.2016 01:00	3335	3335	10246	10246
13.04.2016 07:00	2049	2049	8084	8084
13.04.2016 13:00	2487	2487	7913	7913
13.04.2016 19:00	2780	2780	8606	8606
14.04.2016 01:00	2855	2855	8853	8854
14.04.2016 07:00	1841	1841	7276	7276
14.04.2016 13:00	2283	2283	7733	7733
14.04.2016 19:00	3066	3066	9369	9369
15.04.2016 01:00	2876	2876	9396	9396
15.04.2016 07:00	2065	2065	7352	7352
15.04.2016 13:00	2715	2715	7768	7768
15.04.2016 19:00	3384	3384	8657	8657
16.04.2016 01:00	3266	3266	8764	8764
16.04.2016 07:00	1967	1967	7436	7436
16.04.2016 13:00	2422	2422	6975	6975
16.04.2016 19:00	2366	2366	6997	6997
17.04.2016 01:00	1551	1551	4674	4674
17.04.2016 07:00	1127	1127	4650	4650
17.04.2016 13:00	1727	1727	5674	5674
17.04.2016 19:00	2219	2219	6226	6226
20.04.2016 07:00	2018	2018	7239	7239
20.04.2016 13:00	2691	2692	7362	7363
20.04.2016 19:00	2868	2868	7762	7762
21.04.2016 01:00	2851	2851	8482	8482
21.04.2016 07:00	1961	1961	6974	6974
21.04.2016 13:00	2447	2447	7633	7633
21.04.2016 19:00	2874	2874	8210	8210
22.04.2016 01:00	3600	3600	10315	10316
22.04.2016 07:00	2507	2507	8926	8926
22.04.2016 13:00	3051	3051	9894	9895
22.04.2016 19:00	3419	3420	11597	11597
23.04.2016 01:00	2891	2891	10216	10217
23.04.2016 07:00	1860	1860	7625	7625
23.04.2016 13:00	2315	2315	8961	8961
23.04.2016 19:00	3031	3031	9823	9823
24.04.2016 01:00	2760	2760	10391	10391
24.04.2016 07:00	1731	1731	7506	7506
24.04.2016 13:00	1778	1778	7363	7363
24.04.2016 19:00	3060	3060	12172	12172
25.04.2016 01:00	3077	3077	11832	11832
25.04.2016 07:00	2080	2080	8966	8966
25.04.2016 13:00	2654	2654	10048	10048
25.04.2016 19:00	2999	2999	9521	9521
Total	116983	116985	386082	386087

Attachment 2

date	circuits
12.04.2016 07:00	233
12.04.2016 13:00	326
12.04.2016 19:00	323
13.04.2016 01:00	298
13.04.2016 07:00	249
13.04.2016 13:00	239
13.04.2016 19:00	264
14.04.2016 01:00	243
14.04.2016 07:00	231
14.04.2016 13:00	277
14.04.2016 19:00	351
15.04.2016 01:00	279
15.04.2016 07:00	226
15.04.2016 13:00	256
15.04.2016 19:00	303
16.04.2016 01:00	268
16.04.2016 07:00	237
16.04.2016 13:00	249
16.04.2016 19:00	186
17.04.2016 01:00	153
17.04.2016 07:00	194
17.04.2016 13:00	198
17.04.2016 19:00	196
20.04.2016 07:00	217
20.04.2016 13:00	260
20.04.2016 19:00	305
21.04.2016 01:00	276
21.04.2016 07:00	258
21.04.2016 13:00	296
21.04.2016 19:00	351
22.04.2016 01:00	328
22.04.2016 07:00	302
22.04.2016 13:00	327
22.04.2016 19:00	352
23.04.2016 01:00	288
23.04.2016 07:00	254
23.04.2016 13:00	306
23.04.2016 19:00	357
24.04.2016 01:00	319
24.04.2016 07:00	224
24.04.2016 13:00	270
24.04.2016 19:00	400
25.04.2016 01:00	330
25.04.2016 07:00	326
25.04.2016 13:00	334
25.04.2016 19:00	366

Attachment 3

date	sent	received
12.04.2016 07:00	478.68	477.36
12.04.2016 13:00	481.29	479.97
12.04.2016 19:00	484.76	483.43
13.04.2016 01:00	486.93	485.59
13.04.2016 07:00	489.57	488.22
13.04.2016 13:00	492.25	490.89
13.04.2016 19:00	495.4	494.04
14.04.2016 01:00	497.9	496.54
14.04.2016 07:00	501.22	499.85
14.04.2016 13:00	502.64	501.27
14.04.2016 19:00	504.87	503.49
15.04.2016 01:00	507.53	506.14
15.04.2016 07:00	509.52	508.14
15.04.2016 13:00	511.79	510.39
15.04.2016 19:00	514.38	512.98
16.04.2016 01:00	518.35	516.94
16.04.2016 07:00	521.49	520.06
16.04.2016 13:00	522.88	521.46
16.04.2016 19:00	525.4	523.97
17.04.2016 01:00	526.88	525.44
17.04.2016 07:00	528.35	526.91
17.04.2016 13:00	530.99	529.55
17.04.2016 19:00	533.04	531.59
20.04.2016 07:00	554.66	553.11
20.04.2016 13:00	557.77	556.21
20.04.2016 19:00	560.71	559.15
21.04.2016 01:00	564.64	563.06
21.04.2016 07:00	566.89	565.3
21.04.2016 13:00	570.15	568.55
21.04.2016 19:00	572.84	571.24
22.04.2016 01:00	576.12	574.51
22.04.2016 07:00	579.39	577.77
22.04.2016 13:00	584.57	582.93
22.04.2016 19:00	587.91	586.26
23.04.2016 01:00	590.13	588.48
23.04.2016 07:00	592.6	590.95
23.04.2016 13:00	595.09	593.43
23.04.2016 19:00	598.07	596.4
24.04.2016 01:00	600.96	599.28
24.04.2016 07:00	603.76	602.07
24.04.2016 13:00	605.22	603.53
24.04.2016 19:00	608.49	606.79
25.04.2016 01:00	613.02	611.3
25.04.2016 07:00	615.6	613.87
25.04.2016 13:00	618.56	616.82
25.04.2016 19:00	621.33	619.59

Attachment 4

date	al	at	au	be	bg	br	by	ca	ch
11.02.16	8		8	24	16	16	8	104	80
12.02.16	120	32	8	32	16	16	8	8	88
13.02.16	8	8	56	32	8	16	8	128	8
14.02.16	128	120	48	32	8	8	8	8	88
15.02.16	128	120	48	32	8	8	8	8	88
16.02.16	8		8					8	8
17.02.16	88		128	16	8	16	8	8	64
18.02.16	128	32	48	32	16	24	8	8	88
19.02.16	8	8	8	8	48	736	8	120	96
20.02.16	8	8	40	32	16	16	8	128	88
21.02.16	8	8	8	8	40	696	8	120	96
22.02.16	120		8	32	8	8	8	32	88
23.02.16	120		160	8	24	8	8	32	88
24.02.16	128	8	8	40	496	8	8	8	96
25.02.16	128	128	8	8	24	24	32	32	96
26.02.16	8	136	56	48	496	24	8	128	96
27.02.16	128		152	496	24	8	8	8	96
28.02.16	120	40	8	40	472	24	32	8	88
29.02.16	112		8	16	8	32	8	32	80
01.03.16	8		8	32	472	24	32	112	80
02.03.16	104		8	8	16	8	552	32	72
03.03.16	104		144	40	456	24	24	32	80
04.03.16	112		152	464	32	32	608	32	80
05.03.16	112		8	40	472	16	8	32	88
06.03.16	8	32	48	8	48	640	8	120	8
07.03.16	112		8	464	32	8	16	32	88
08.03.16	112	112	8	8	24	8	32	8	72
09.03.16	112	112	8	8	24	8	32	8	72
10.03.16	8		24					24	8
11.03.16	8		8	8	8	32		8	8
12.03.16	8		8	32	472	24	32	112	88
13.03.16	120	128	8	8	40	744	8	8	80
14.03.16	8	112	8	32	8	16	8	120	88
15.03.16	120		8	456	40	8	8	8	88
16.03.16	8	120	8	8	24	8	32	128	8
17.03.16	120	128	8	8	40	712	8	8	88
18.03.16	128	32	8	32	16	16	8	40	88
19.03.16	136	144	8	16	24	24	40	40	88
20.03.16	8	8	56	40	16	24	8	8	88
21.03.16	128	8	8	40	488	24	8	48	96
22.03.16	112	8	8	40	480	24	8	40	88
23.03.16	128		8	16	24	8	32	8	88
24.03.16	8	32	8	40	496	24	8	120	96
25.03.16	8	8	56	8	48	760	8	120	96
26.03.16	136	144	192	488	40	8	8	40	96
27.03.16	8	8	8	8	40	760	8	128	8
28.03.16	8	32	8	40	16	16	8	128	104
29.03.16	128		184	24	8	8	8	40	96
30.03.16	8	144	8	40	480	16	8	120	88
31.03.16	8	24	8	472	40	8	8	120	8
01.04.16	128	136	184	496	40	8	8	40	104
02.04.16	8		176	480	40	8	32	120	88
03.04.16	8	8	56	8	40	736	8	120	104
04.04.16	128	8	8	32	16	16	8	8	88
05.04.16	8		56	32	16	16	8	120	96
06.04.16	128		8	40	480	8	8	8	88
07.04.16	8	56	8	16	24	8	8	112	88
08.04.16	112	56	8	8	32	24	8	40	80
09.04.16	8		176	24	8	8	8	112	8
10.04.16	8		200	496	40	8	8	128	80
11.04.16	8	8	56	40	24	16	8	136	88
12.04.16	136	152	8	48	512	24	8	8	104
13.04.16	120		8	40	704	32	16	48	96
14.04.16	8	32	8	496	32	8	32	120	8
15.04.16	112	8	16	40	32	8	24	8	88
16.04.16	120		184	488	32	8	8	32	96
17.04.16	8	32	8	8	24	8	8	120	96

date	cz	de	dk	do	ee	eg	es	fi	fr
11.02.16	8	32	488	8	8	8	8	8	40
12.02.16	48	40	496	8	8	8	8	8	40
13.02.16	8	40	488	16	32	8	8	8	48
14.02.16	8	32	496	8	8	8	16	8	32
15.02.16	8	32	496	8	8	8	16	8	32
16.02.16	8			8	8	8	8		
17.02.16	8	16	488	8	8	24	200	8	24
18.02.16	8	40	496	8	24	8	8	8	40
19.02.16	8	24	8	8	8	16	8	24	32
20.02.16	8	32	488	8	16	8	16	8	40
21.02.16	8	24	8	16	8	16	8	24	32
22.02.16	168	24	664	8	8	32	8	24	32
23.02.16	8	680	32	8	16	8	280	32	472
24.02.16	48	8	32	8	16	40	176	8	8
25.02.16	8	720	32	8	16	8	176	8	488
26.02.16	8	8	32	8	8	48	16	8	8
27.02.16	8	16	8	8	8	32	8	32	24
28.02.16	40	8	24	8	16	48	16	8	8
29.02.16	144	24	632	8	8	32	8	8	24
01.03.16	40	8	24	16	16	32	8	8	8
02.03.16	144	8	16	8	32	464	8	8	624
03.03.16	8	8	24	8	8	40	8	8	8
04.03.16	8	16	8	8	16	32	248	16	24
05.03.16	8	8	24	8	16	40	144	8	8
06.03.16	8	24	8	16	8	16	8	24	24
07.03.16	8	16	8	8	16	32	144	8	24
08.03.16	8	672	32	8	16	8	160	8	464
09.03.16	8	672	32	8	16	8	160	8	464
10.03.16	8			8			8		
11.03.16	8	8	8	8	8	8	24		8
12.03.16	8	8	24	8	16	40	160	8	8
13.03.16	48	24	8	8	16	8	8	24	32
14.03.16	48	32	472	8	24	8	16	8	40
15.03.16	48	16	8	8	16	32	160	32	24
16.03.16	56	704	32	16	8	8	16	8	464
17.03.16	48	24	16	8	24	8	16	32	24
18.03.16	8	32	480	8	24	8	16	8	48
19.03.16	48	744	32	8	16	8	16	8	480
20.03.16	40	40	480	8	24	8	16	8	40
21.03.16	8	8	24	8	16	48	184	8	16
22.03.16	8	8	24	8	8	48	8	8	16
23.03.16	8	744	40	8	16	8	176	8	472
24.03.16	8	8	24	8	8	40	8	8	16
25.03.16	8	24	16	8	24	8	8	24	32
26.03.16	8	24	8	8	16	32	304	32	24
27.03.16	56	24	16	16	24	8	8	24	32
28.03.16	56	40	480	8	24	8	16	8	48
29.03.16	8	8	712	8	16	40	280	8	32
30.03.16	48	8	24	8	24	48	16	8	8
31.03.16	8	16	8	16	16	32	176	32	24
01.04.16	8	16	8	8	16	32	280	32	24
02.04.16	8	16	8	8	16	32	280	8	24
03.04.16	8	24	16	8	24	8	16	24	40
04.04.16	56	40	488	8	24	8	16	8	40
05.04.16	8	32	488	8	24	8	16	8	48
06.04.16	64	8	24	8	8	48	16	8	8
07.04.16	56	696	40	8	16	8	168	8	480
08.04.16	8	680	32	8	24	8	168	8	480
09.04.16	16	24	656	16	16	40	312	8	40
10.04.16	8	16	8	8	16	32	344	8	32
11.04.16	8	40	504	8	8	8	16	24	48
12.04.16	56	8	24	8	8	40	16	8	8
13.04.16	200	8	16	8	48	16	8	8	8
14.04.16	48	16	8	16	16	32	176	8	24
15.04.16	48	472	40	8	16	8	16	688	8
16.04.16	16	16	8	8	16	32	304	32	32
17.04.16	56	680	40	8	8	8	16	24	488

date	gb	gr	hr	hu	id	ie	il	im	in
11.02.16	48	16	24	688	8	160	8	8	8
12.02.16	176	16	32	744	16	8	8	8	8
13.02.16	8	96	24	768	8	8	32	8	184
14.02.16	8	16	24	752	8	8	8	8	176
15.02.16	8	16	24	752	8	8	8	8	176
16.02.16	8	8			8	16	16		8
17.02.16	8	8	8	8	112	8	8		8
18.02.16	8	16	32	768	8	8	40	8	8
19.02.16	48	16	480	8	8	8	32	8	8
20.02.16	8	16	24	680	8	8	40	8	168
21.02.16	48	8	472	8	8	8	8	24	8
22.02.16	8	16	16	8	280	160	8	8	16
23.02.16	8	16	8	24	160	16	8	120	8
24.02.16	160	16	40	16	16	8	8	8	8
25.02.16	152	16	8	8	16	8	8	8	8
26.02.16	8	16	40	16	8	8	8	8	160
27.02.16	8	16	704	8	256	168	8	8	8
28.02.16	8	16	24	8	8	8	8	8	152
29.02.16	8	16	16	8	240	144	8	120	8
01.03.16	136	8	32	8	16	8	8		8
02.03.16	8	8	8	8	240	144	8		8
03.03.16	8	16	24	16	16	8	8	8	8
04.03.16	8	16	664	8	152	8	8		8
05.03.16	136	16	32	664	8	8	8	112	8
06.03.16	8	80	456	8	8	8	8	8	8
07.03.16	160	16	656	8	16	8	8	120	8
08.03.16	160	16	8	24	16	8	8	8	8
09.03.16	160	16	8	24	16	8	8	8	8
10.03.16	8	8			8	8			8
11.03.16	8	16	8	16	8	8	8		40
12.03.16	152	16	40	16	16	8	8	56	8
13.03.16	8	16	488	8	8	8	48	8	160
14.03.16	8	16	24	680	16	8	32	8	152
15.03.16	8	16	704	8	16	8	8	56	168
16.03.16	8	96	8	24	8	8	8	8	160
17.03.16	8	16	480	8	8	8	40	8	176
18.03.16	184	16	24	736	8	8	40	8	8
19.03.16	8	16	8	8	8	8	16	8	192
20.03.16	8	16	32	728	8	8	40	8	8
21.03.16	200	16	40	16	16	8	8	8	8
22.03.16	192	16	40	16	16	8	16	8	8
23.03.16	192	16	8	24	16	8	8	136	8
24.03.16	192	16	40	16	16	8	16	8	8
25.03.16	8	16	496	8	8	8	40	16	8
26.03.16	8	16	760	8	192	16	8	8	8
27.03.16	8	96	496	8	8	8	40	8	200
28.03.16	8	16	24	752	8	8	16	8	192
29.03.16	8	16	24	8	176	16	8	32	8
30.03.16	8	16	40	16	8	8	40	8	184
31.03.16	192	88	704	8	16	8	8	8	8
01.04.16	8	16	736	8	184	16	8	8	8
02.04.16	8	16	688	24	168	16	8	32	8
03.04.16	8	16	480	8	8	8	32	8	184
04.04.16	8	16	24	736	8	8	40	8	192
05.04.16	8	16	24	704	8	8	40	8	184
06.04.16	8	16	40	16	8	8	16	56	192
07.04.16	192	16	8	24	16	8	8	8	8
08.04.16	168	16	8	8	16	8	8	8	8
09.04.16	8	88	16	8	168	16	8	56	8
10.04.16	8	16	744	8	176	16	8	136	8
11.04.16	16	16	24	752	16	8	16	8	200
12.04.16	8	16	40	24	16	8	16	8	216
13.04.16	8	16	24	8	344	168	8	8	8
14.04.16	192	88	720	24	16	16	8	8	8
15.04.16	184	16	8	40	8	8	32	8	8
16.04.16	8	16	680	24	168	16	8	56	8
17.04.16	192	16	8	16	8	8	16	8	8

date	is	it	jp	kr	kz	lt	lu	lv	md
11.02.16	32	8	8	8	24	24	16	8	8
12.02.16	32	32	8	8	24	16	16	24	8
13.02.16	32	8	8	744	24	8	16	8	8
14.02.16	8	32	8	728	24	24	16	32	32
15.02.16	8	32	8	728	24	24	16	32	32
16.02.16		8				48	8	8	
17.02.16	440	24	8	24	8	32	8	16	8
18.02.16	24	32	8	8	24	8	24	8	8
19.02.16	8	8	32	8	8	24	8	8	8
20.02.16	8	8	8	8	24	8	16	8	32
21.02.16	8	8	32	8	16	24	8	16	8
22.02.16	8	48	8	8	8	48	8	16	16
23.02.16	16	8	8	8	8	8	8	16	16
24.02.16	8	32	736	696	24	16	8	8	32
25.02.16	8	8	16	8	8	16	8	8	8
26.02.16	32	8	744	16	8	16	16	32	8
27.02.16	16	32	24	8	8	8	40	16	24
28.02.16	8	32	688	8	8	16	8	8	8
29.02.16	8	40	8	8	8	32	8	16	584
01.03.16	8	8	624	56	8	16	8	16	8
02.03.16	8	40	32		16	32	32	8	8
03.03.16	16	48	640	8	8	16	16	24	16
04.03.16	8	48	24	112	8	8	40	16	8
05.03.16	32	8	8	8	24	16	8	8	8
06.03.16	8	32	32	568	16	24	8	8	8
07.03.16	24	8	32	8	32	16	40	8	8
08.03.16	8	24	24	8	8	16	8	8	8
09.03.16	8	24	24	8	8	16	8	8	8
10.03.16		8							
11.03.16		32	136		8	8	8	16	
12.03.16	8	32	680	8	8	16	8	8	8
13.03.16	8	32	32	672	16	8	16	8	8
14.03.16	32	32	8	8	24	8	16	8	8
15.03.16	8	32	24	8	8	16	40	8	8
16.03.16	8	32	24	8	8	16	8	24	8
17.03.16	8	40	32	688	24	8	16	8	8
18.03.16	8	8	8	8	24	8	16	8	32
19.03.16	8	8	16	8	8	16	8	8	8
20.03.16	8	128	8	8	24	8	16	8	32
21.03.16	32	8	744	752	8	16	8	8	8
22.03.16	32	8	696	728	8	16	16	16	8
23.03.16	8	40	16	8	8	16	8	8	16
24.03.16	32	40	768	8	8	16	24	24	8
25.03.16	8	8	32	16	24	8	16	8	8
26.03.16	8	48	24	8	8	8	40	16	8
27.03.16	8	40	40	8	24	8	16	8	8
28.03.16	8	8	8	8	24	16	16	8	32
29.03.16	8	56	24	56	32	48	8	16	16
30.03.16	32	8	736	736	8	16	16	8	8
31.03.16	8	32	16	8	8	16	40	8	8
01.04.16	8	8	24	8	8	8	40	16	16
02.04.16	8	32	8	56	8	8	48	16	8
03.04.16	8	8	40	8	24	8	16	8	8
04.04.16	8	32	8	16	24	8	16	8	24
05.04.16	24	8	8	728	24	8	16	8	8
06.04.16	32	40	712	8	24	16	16	24	8
07.04.16	32	40	16	8	8	16	8	8	8
08.04.16	8	8	16	8	8	16	8	8	32
09.04.16	16	40	8	8	32	40	8	16	8
10.04.16	16	40	24	8	32	8	40	24	8
11.04.16	8	8	8	8	8	16	16	24	8
12.04.16	32	40	800	8	8	16	16	24	8
13.04.16	8	56	8	8	8	8	32	16	16
14.04.16	8	40	8	712	8	16	48	8	8
15.04.16	24	40	24	16	8	8	16	8	8
16.04.16	8	8	8	8	8	8	40	16	16
17.04.16	8	40	8	8	8	16	16	24	16

date	mk	mx	my	nl	nz	no	pa	pl	pt
11.02.16		24	8	16	8	272	32	40	160
12.02.16		8	8	288	8	32	8	40	8
13.02.16	136	8	8	176	8	8	8	40	304
14.02.16	32	32	8	288	8	32	8	40	8
15.02.16	32	32	8	288	8	32	8	40	8
16.02.16			8	8	8	8	16	8	8
17.02.16			8	32	16	8	16	8	8
18.02.16		64	184	184	16	8	8	48	304
19.02.16	64	8	160	168	8	8	8	8	280
20.02.16	24	8	8	168	8	8	8	40	288
21.02.16	8	616	168	8	8	288	40	8	168
22.02.16		32	8	8	16	8	24	8	8
23.02.16			8	32	8	8	8	24	8
24.02.16	128	8	8	8	24	8	8	8	40
25.02.16	32	64	8	8	24	8	8	32	48
26.02.16	32	64	8	280	8	40	8	8	8
27.02.16		40	16	8	16	8	24	40	8
28.02.16		8	8	264	24	32	8	8	8
29.02.16			8	8	16	8	24	8	8
01.03.16			8	256	8	24	8	8	8
02.03.16			16	8	16	8	16	24	8
03.03.16		32	8	240	8	24	8	8	8
04.03.16			8	32	8	8	8	32	8
05.03.16			8	8	16	8	8	8	32
06.03.16		8	136	16	8	264	40	8	152
07.03.16			8	8	24	8	8	40	32
08.03.16	24	64	8	8	24	8	8	32	32
09.03.16	24	64	8	8	24	8	8	32	32
10.03.16			8						
11.03.16			8	8	8	8	8	8	40
12.03.16		112	8	8	24	8	8	8	56
13.03.16	24	8	8	168	8	8	8	8	288
14.03.16	32	56	8	168	8	8	8	40	264
15.03.16		128	8	8	16	8	8	40	40
16.03.16	32	56	8	272	8	32	8	32	8
17.03.16	32	56	8	168	8	8	8	8	272
18.03.16		8	8	184	8	8	8	40	288
19.03.16	32	48	8	288	24	40	8	32	8
20.03.16	136	8	192	192	8	8	16	48	288
21.03.16	152	8	8	8	24	16	8	8	32
22.03.16	128	8	8	288	8	32	8	8	8
23.03.16			8	8	16	16	8	32	40
24.03.16		8	8	296	8	40	8	8	8
25.03.16	8	8	208	176	8	8	16	8	288
26.03.16	40	48	8	40	8	8	8	48	16
27.03.16	8	16	8	184	8	8	16	8	296
28.03.16	8	8	8	288	8	40	8	48	8
29.03.16			8	40	8	8	8	8	16
30.03.16	40	8	8	176	8	8	16	8	280
31.03.16		48	8	8	24	16	8	40	40
01.04.16	40	56	16	40	8	8	8	40	16
02.04.16			8	40	8	8	8	40	16
03.04.16	56	8	8	176	8	8	16	8	280
04.04.16	144	8	8	8	8	8	16	40	288
05.04.16		144	8	8	8	8	16	40	280
06.04.16		152	8	304	8	40	16	8	8
07.04.16	8	8	8	8	24	16	8	32	40
08.04.16	8	8	8	8	24	16	8	32	40
09.04.16		128	8	32	8	8	8	16	16
10.04.16			8	40	8	8	8	48	16
11.04.16	8	8	8	360	8	40	8	40	8
12.04.16	40	8	8	360	8	40	8	8	8
13.04.16		40	16	16	16	8	24	480	8
14.04.16		56	8	8	24	16	8	40	40
15.04.16	8	8	8	8	8	8	16	8	296
16.04.16		40	16	40	8	8	8	32	16
17.04.16		8	8	304	8	32	8	32	8

date	ro	ru	rs	se	sg	si	sk	tr	tw
11.02.16	16	8	16		8	8	8	8	56
12.02.16	8	168	16	128	8	8	8	8	8
13.02.16	16	8	8	64	8	16	8	8	8
14.02.16	8	168	8	8	8	16	56	8	8
15.02.16	8	168	8	8	8	16	56	8	8
16.02.16	8	8						8	
17.02.16	24	8	40		8		88	8	
18.02.16	16	16	24	128	16	8	752	8	8
19.02.16	16	16	8	8	24	672	16	8	8
20.02.16	16	8	8	8	8	672	64	8	8
21.02.16	16	16	8	8	8	8	8	8	16
22.02.16	24	40	608		8	8	8	8	56
23.02.16	40	8	8		648	56	8	16	8
24.02.16	8	296	16	64	8	8	8	8	8
25.02.16	48	272	24	8	16	8	688	8	8
26.02.16	8	192	16	8	16	8	768	8	8
27.02.16	8	32	8		672	8	8	8	56
28.02.16	8	160	16	128	16	8	8	8	8
29.02.16	32	32	8		8	56	8	8	8
01.03.16	8	152	592		16	120	8	8	24
02.03.16	8	24	56		8		24	16	
03.03.16	8	144	584		8	8	8	8	56
04.03.16	8	8	8		8	32	8	24	
05.03.16	8	248	16		16	56	8	8	8
06.03.16	16	8	8	104	32	8	16	8	8
07.03.16	8	264	8		632	8	8	8	56
08.03.16	40	272	16	8	8	8	616	8	8
09.03.16	40	272	16	8	8	8	616	8	8
10.03.16		8							
11.03.16	136	56						8	
12.03.16	8	272	8	24	16	8	8	8	8
13.03.16	16	24	8	56	32	8	16	8	8
14.03.16	16	8	16	8	8	8	632	8	8
15.03.16	8	272	8	32	16	8	8	8	8
16.03.16	40	160	8	8	16	8	8	8	8
17.03.16	16	8	8	8	32	16	24	8	8
18.03.16	16	8	8	136	8	696	24	8	8
19.03.16	40	184	16	8	16	8	728	8	8
20.03.16	16	16	8	56	8	736	24	8	8
21.03.16	8	296	16	56	8	8	8	8	8
22.03.16	8	176	8	24	8	8	8	8	8
23.03.16	40	288	720		56	24	8	8	8
24.03.16	8	168	16	152	8	8	760	8	8
25.03.16	16	16	8	8	32	24	8	8	736
26.03.16	8	8	8	8	16	8	8	16	8
27.03.16	16	16	8	8	32	8	8	8	16
28.03.16	8	184	8	8	8	720	56	8	8
29.03.16	32	8	8		680	8	8	24	128
30.03.16	8	8	8	56	8	8	16	8	8
31.03.16	8	288	8	128	16	8	8	8	8
01.04.16	8	8	24	8	8	8	704	24	8
02.04.16	8	8	16		8	8	8	24	136
03.04.16	16	8	8	8	32	24	8	8	744
04.04.16	16	168	8	56	8	8	8	8	704
05.04.16	16	184	16	32	8	8	8	8	8
06.04.16	8	184	16	32	8	8	712	8	8
07.04.16	48	312	16	8	8	712	16	8	16
08.04.16	40	296	8	8	8	16	16	8	640
09.04.16	32	8	8	32	16	8	8	24	8
10.04.16	8	8	720		16	8	8	24	64
11.04.16	8	184	8	8	32	16	16	8	760
12.04.16	8	208	16	32	8	864	64	8	8
13.04.16	40	32	728		8	8	8	8	56
14.04.16	8	288	8	144	8	8	8	8	8
15.04.16	16	168	32	8	8	8	8	8	16
16.04.16	8	8	16		8	8	8	16	8
17.04.16	48	168	8	128	16	8	8	8	8

date	ua	us	vn	??	Total
11.02.16	96	616	32	8	752
12.02.16	56	696	8	8	768
13.02.16	8	24	32	8	72
14.02.16	8	16	8	8	40
15.02.16	8	16	8	8	40
16.02.16			8		8
17.02.16		8	32	400	440
18.02.16	8	8	8	8	32
19.02.16	8	16	32	40	96
20.02.16	8	8	24	8	48
21.02.16	8	24	32	40	104
22.02.16	128	8	8	456	600
23.02.16	32	8	40	40	120
24.02.16	8	24	8	24	64
25.02.16	8	8	56	40	112
26.02.16	8	8	32	32	80
27.02.16	128	8	48	8	192
28.02.16	56	664	8	32	760
29.02.16	24	8	8	464	504
01.03.16		8	32	32	72
02.03.16		104	8	32	144
03.03.16	120	8	8	24	160
04.03.16		56	8	8	72
05.03.16	32	584	48	32	696
06.03.16	56	16	8	40	120
07.03.16	32	8	48	8	96
08.03.16	8	8	48	40	104
09.03.16	8	8	48	40	104
10.03.16			8		8
11.03.16			8	152	160
12.03.16	8	616	48	32	704
13.03.16	8	16	8	40	72
14.03.16	8	8	8	8	32
15.03.16	8	664	8	16	696
16.03.16	8	648	8	32	696
17.03.16	8	16	8	40	72
18.03.16	56	8	48	8	120
19.03.16	8	8	8	32	56
20.03.16	8	16	8	8	40
21.03.16	8	24	48	32	112
22.03.16	8	56	56	24	144
23.03.16	32	8	48	40	128
24.03.16	56	8	56	40	160
25.03.16	8	8	40	48	104
26.03.16	8	712	8	16	744
27.03.16	752	8	8	48	816
28.03.16	8	16	32	8	64
29.03.16		8	8	488	504
30.03.16	8	8	32	32	80
31.03.16	8	696	48	16	768
01.04.16	8	8	48	16	80
02.04.16		672	48	8	728
03.04.16	8	8	40	48	104
04.04.16	8	16	8	8	40
05.04.16	56	16	40	8	120
06.04.16	8	24	8	32	72
07.04.16	8	8	8	40	64
08.04.16	8	8	56	32	104
09.04.16	8	704	56	480	1248
10.04.16	32	8	56	16	112
11.04.16	8	8	32	8	56
12.04.16	8	8	8	40	64
13.04.16	128	8	8	24	168
14.04.16	8	24	8	8	48
15.04.16	672	8	8	8	696
16.04.16	144	680	48	8	880
17.04.16	56	656	8	32	752