Master's Thesis 2018
Faculty of Science and Technology

Evaluating the correlation between site speed and its effect on conversion rates for Norwegian Air Shuttle ASA

Ali Vindenes Fetouni

Preface

The assignment within the thesis has enable for deeper exploration of the field of web site performance and Application Performance Management. The subject by itself is vast as it connects, or at least has the possibility to connect, many different parts of a company. There is as much interest for a market division as it is for a IT operations team, to know how performance ties in with the business. After discussions around the subjects with Michael Breen and Philip Duns at Red Ocean, an idea crystallised and was presented to Norwegian. More people got involved further down the path of the thesis which made things more interesting and bearable.

The actual assignment was conducted with the help of several different people, mostly inside Norwegian's IT operations and IT development teams. It is with great ease I would like to thank several people for their contribution. First of all, Lars Ove Claesson, for enabling the assignment on behalf of Norwegian. Secondly a thank you to Gunhild Berge at the marketing division for all the material regarding funnels and user journeys. Finally, a very big thank you to Kai Boris Bienek who on several occasions has used is spare time to aid with setup of software and configuration of servers. I would not have done it without you.

In Red Ocean Technology, I would like to thank Philip Duns, Michael Breen and Tore Davidsen for enabling this assignment, and having enough trust and belief in my capacity and knowledge. Furthermore, I would like to thank Sanpreet Garcha. Torsten Granli, Tarjei Utnes and Arild Palencia for being technical sparring and discussion partners. A special thanks to Gary Kaiser at Dynatrace for providing me with reading material.

In NMBU, I would like to thank Tor Kristian Stevik for the guidance I have received and for opening the right doors when needed. I literally would not have able to finish the assignment without you. A big thank you to Bjarne Gjerde at Nofima, for taking time to listen to me ramble, discussing statistics and pointing in the right direction.

Finally, a big thank you to my beloved family, Linn, Julian and Leander, who have had more than enough patience with me. I owe you my future.

Abstract

The matter of how speed correlates with conversion rates is a field that is built up by many commercials testimonies. Many of these testimonies are based on statements that have little to no root sources. The research starts with investigating the relationship between speed and conversion. By eliminating backend metrics that have less effect on page load times, the focus is turned to frontend time. Having established this notion, end user data is collected by utilising Dynatrace Application Monitoring User Experience Monitoring. The choice of method excludes several dimensions which are elaborated in the discussion chapter. In the analysis, the several logistic models that were tested showed that several of the original metrics could be omitted due to the lack of effect. The data analysis shows that there is a strong relationship between page speed and conversion rates. The results of the method and the analysis is presented in such a way that is can be applied universally.

Table of contents

List of figures

List of equations

List of tables

Abbreviations and symbols

ACK	ACKnowledgement
AMD	Agentless Monitoring Device
API	Application Programming Interface
APM	Application Performance Management
CAS	Central Analysis Server
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSSOM	Cascading Style Sheets Object Model
DC RUM	Data Centre Real User Monitoring
DOM	Domain Object Model
DPM	Digital Performance Management
FIFO	First In First Out
Gbps	Gigabits per seconds
HCI	Human Computer Interaction
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	Input / Output
kB	Kilobyte
MSS	Maximum Segment Size
NIC	Network Interface Card
OSI	Open Systems Interconnection
RTT	Round Trip Time
RUM	Real User Monitoring
SYN	SYNchronize
TCP	Transmission Control Protocol
TTFB	Time to First Byte
UDP	User Datagram Protocol
URL	Uniform Resource Locator
XHR	XML HTTP Request
XML	Extensible Markup Language

Definitions

Backend time

The time it takes for a server (here web server) to get the first byte back to the client. Hereafter called Time To First Byte or TTFB. This time also includes DNS time, socket connections, SSL negation and redirects.

Document onLoad (onLoad)

Measures when the entire page is loaded including images, JavaScript and other included resources.

DOM Interactive (domInteractive)

marks the point when the browser has finished parsing all of the HTML and DOM construction is complete.

Frontend time

The time it takes from client has gotten the first byte until the web page is fully loaded. Is represented by the Navigation Timing API variable fullyLoaded.

Fully loaded (fullyLoaded)

The time from the start of the initial navigation until there was 2 seconds of no network activity after Document Complete. This will usually include any activity that is triggered by JavaScript after the main page loads.

Measures

In AppMon measures are metrics, or data points, that are collected stored either periodically or based on transactions. Measures are used for long-term charting, trend analysis, and as the basis for configuring incidents.

Network time

The time the network takes to deliver the request to the server and to deliver the resulting response back to the user. In other words, network time is the portion of the operation time that is spent on transferring data over the network.

Site

An IP network from which a user logs in to a monitored network.

Time to First Byte (TTFB)

Also referred to as backend time. This metric also includes DNS time, socket connections, SSL negation and redirects.

URL – Uniform Resource Locator
Informally also known as a web address e.g. http://www.norwegian.no. URL refers to the method of obtaining a resource. URL and URI will be used interchangeably.

URI – Uniform Resource Identifier
A string of characters to identify a resource (physical or abstract). URL and URI will be used interchangeably.

Visit
A visit is a collected set of user actions performed by a user within a certain time period.

Web request
The action of a browser fetching a resource on a web site, utilising the HTTP protocol.

1 Introduction

1.1 Background

Since the dawn of digitalisation, inventions and innovation have gained speed and transformed all in their path. The knowledge gained throughout this transformation has been more and more democratised and somewhere along said path became, on one side a commodity, and on the other side a necessity. One of the results of this democratisation of knowledge is a plethora of technologies and areas of application said technologies. The intricacy and myriad of technologies are mirrored in the way computer infrastructure is set up and the amount of integrations needed for all these technologies to work together. This is the complexity all companies are facing today. By itself this is a challenge, as many companies are governed by the means of structures formed indirectly by employee specialisations and corporate hierarchy. Often, this results in lack of transparency and "the right hand not knowing what the left hand is doing"-syndrome. Adding the customer base into this rather complex equation, translates into having to navigate "in the dark". The combination of what the customer wants and the services a company delivers is a, if not *the*, fundamental concept of commerce. Having to move away from the brick and mortar transactions that has governed the natural order of business of the market, presents challenges for companies. The speed of transactions and the reach beyond the local arena is appealing and at its core is transforming the way companies see themselves and how the need to operate in order to survive. This by itself is not something new, nor is it unique as technology always disrupts markets and thus forcing already established player to come to realisation of the new disguise. Even though reality is more complex than a couple of statements, it is at its core plain and simple – adapt or see yourself disrupted. Michael Corbat, CEO of Citi bank describes this shift in paradigm in this manner, *[I]n many ways, we see ourselves as a technology company with a banking license.*

With the democratisation of knowledge and technology it is easier to play amongst the big players. It is easier than ever to scale companies both horizontally and vertically without needing the manpower one needed 50 or more so years ago. This may be something that by itself has high value, it's hard to argue otherwise, but it at the same time adds to the complexity and additional layers of obscurity. The more infrastructure the harder it is to have an overview. The less of an overview, the less control one has of one's customers and their desires. This directly influences the bottom line. There are, not surprisingly, many variables to this complex equation. In a macroscopic view the complexity consists of infrastructure, comprised of servers, clients and network components; the software and configuration of it that goes on top of the servers and client machinery and finally there are the machines being utilised by the users and their behaviour. There are probably many factors that influence what resides within the three categories presented, but as this thesis argues none as important as speed.

1.2 Problem statement

The web industry, in 2006, advocated a 4 second rule for a web page to load (Jupiter research, 2006). Research backs this rule by concluding with more or less the same results. Google shows that users that experience latency exceeding 4-5 seconds are prone to choosing a faster search engine (Brutlag, Hutchinson & Stone, 2008). This stage is set relatively early in the 21st century, dare one to say by those that could benefit most from it.

Since then a lot have happened and mobile devices push the boundaries of what is perceived as fast. After all, the consumers have everything at their fingertips and the virtue of patience is conspicuous by its absence. Google encourages for webpages to be faster and has good reasons for doing so. In 2010, the company proclaimed that they would take site speed into account in their search rankings (Google, 2010). Thus, the search engine company pushes the boundary of the future to be, more or less forcing companies to focus on site speed. There are probably many underlying factors considered when promoting the inclusion of page speed into search ranking. Some are more obvious than others, but at the core of the lies a principle that tagged along humanity since day one, namely to do things better and faster. As we shall see later on, speed affects conversion rates. Evermore so, the perception of things being fast affects conversion rates. What is frustrating and remains unclear in many investigations are the underlying principles that affect the conversion rate of a random web site. The reasons for the facts remaining unclear, may be due to the very sensitive nature of the numbers. Not many companies would like to give up their secrets, Tesla being one exemption, but amid the fog of obscurity is the need to simultaneously show one's customers that progress is made. Much of the information, on which the thesis finds its nourishment, remains behind the veil and consequently acts as a spring board for curiosity (and energy to move forward with investigations on the subject).

The hypothesis before proceeding with the master thesis, is that there is a relation between the digital performance of a web site and its underlying components and the number of sales on the same web site (conversion rate). Not clear is if this relation only resides within the realm of performance and what other factors that can explain the amount of sales. Furthermore, defining a user journey as *one* whole cannot be considered correct. A user journey is comprised of many steps, as it is made up of the total impression and all the touch points that a company exposes its users to.

Thus, the main objective of this thesis is:
To research if web site speed has any effect on conversion rates for visitors of the web site.

The company under the loupe is Norwegian Air Shuttle ASA.

1.3 Scope, relevance and limitations

The boundaries of the work from this page and forward, embody an investigation limited to data collected from servers within the realm of Norwegian Air Shuttle ASA (Norwegian from this point forward). However, for the pure purpose of research acting as a basis that will lead up to the understanding of the collected data, other resources will nonetheless be utilised.

The data collection platform that is utilised comes from Dynatrace and is called Dynatrace Application Monitoring (AppMon). The platform per-se, has the possibility to collect data from every transaction made, for the later to be defined user journey. Even though that the possibility of capturing all transactions with many underlying details, are at one's fingertips, there is a possibility that the resulting overhead will be too cumbersome on the systems. Thus, a scenario where the amount of details, correlated with each and every transaction, might be tuned down.

Having the possibility of collecting a vast amount of data results in the possibility of gaining synergetic effects. This of course depends on the outlook one haves and what questions one asks with the data foundation at hand. As a request, having its roots in the plethora of data, Norwegian would like to track users who experience bad performance and if they return to the site once the performance issues have been alleviated. Furthermore, Norwegian would like to see the effect on performance correlated to conversion rates, from one software release to the other. These sub objectives fall outside the scope of the thesis, since there is a time and resource limitation to adhere to.

2 Theory

As a base for this thesis, a framework is introduced for which one can base performance analysis upon. The framework will, hopefully, enable the reader to gain understanding of a world that is affected by complexity and many complicated issues. Within the framework, part of the attention will be directed towards addressing understanding how communication protocols work. The protocol to be highlighted throughout the thesis is Transmission Control Protocol (TCP), as it plays the major role of network communication. Many complex problems require the breakdown of said problem into pieces that can be handled, this is not an exemption. To serve as a least common factor, will be the concept of a packet, and how this packet flows in the context of TCP/IP. One does, of course not need to be as specific and detailed on how packet communication is conducted between the different nodes of a network. Nonetheless the assumption that this granularity of detail is needed will become self-evident as the individual parts, step by step gives the reader the total picture. With the concept of a packet in the context of networking, one cannot omit the idea of packets travelling from one node to the other. To comprehend the flow of packets, the packet flow diagram is introduced and used throughout the technical part of the theory. The nodes mentioned the preceding text, is the basic elements that comprise a network i.e. clients, servers, routers, switches etc. For the sake of understanding, clients and servers will be utilised to illustrate core concepts, as these are the core components that are required in order for communication to work.

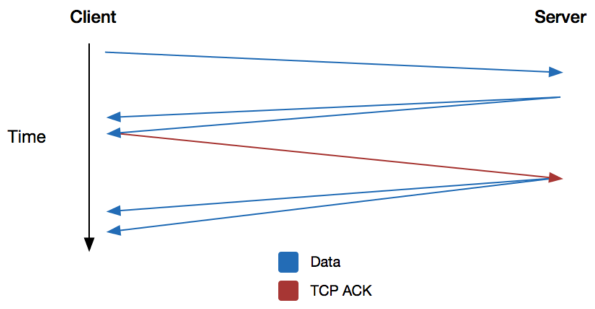


Figure 1 – Illustration of a packet flow diagram

In **Figure 1**, a basic packet flow diagram is used to illustrate transactions and message flow for which much of the work in this thesis is based upon. The convention of said flow diagram are:

- Time flows from the top down
- Each row represents one TCP packet
- Blue arrows represent data packets
- Red arrows represent TCP ACK packets
- The slope of arrows represents network delays

A transaction can be many things, but in the context of a user it is henceforth defined as the unit of work an application does on behalf of the user. It is rather a rule than an exception that one can use the term transaction in very different manners, and correctly so as different contexts, as will become evident when describing the data collection platform, require a different approach with regards to decrypting the logic (that make out the transaction). The concept of a transaction represents a performance metric that is essential to the business as well as for the user and the IT department. One of the reasons for this is the waiting time a user can experience when executing a transaction e.g. clicking a link. This can, on a high level can lower the productivity of users as they are forced to wait for the transaction to complete, and is directly connected to the lower-level IT-managed services and hardware. Each and every transaction is made up of requests and responses on the application level on the client and the corresponding server. At the application level a request message is forwarded to the TCP/IP stack, i.e. residing on lower levels of the OSI reference model, and gets segmented into packets, given an address and transmitted. On the receiving end the process is reversed (Tanenbaum, 1996: 41-48). Within this request/response message exchange between client and server lies, what can be considered as the basis for analysing the performance of transactions. The first part being the actual processing of these messages on the server and client side, and the second being the message transmission i.e. time on the network. Furthermore, one can envision a set of these request/responses as threads, as it most impractical do conduct conversations based on single packets. Since an application layer message most certainly will require more than one data packet, it is evident that the terminology needs to correspond to the nature of packet exchanging.

Another concept that must be introduced before delving into the TCP/IP stack and its components, it the one of the application chattiness. As users perform transactions the application will flowingly send a request with and receive a following response. Not all applications are written (coded) in the same way and thus the amount of request/responses performed by the application and underlying transaction will have an effect on the performance of said application. What becomes interesting is the number of requests and responses and how much information is sent by the application in each exchange. The basic though is that the chattiness of a transaction is directly proportional to the number of turns said transaction takes. That is, the more turns a transaction take, the chattier it is. Seen from the view of the payload – the number of bytes transferred with each application turn – it is much more efficient, and less chatty, if a transaction takes ten turns and transfers 1000 KB, than a ten-turn transaction that only transfers 1 KB. Not farfetched is the thought of sending several transactions in parallel, a topic that is examined in **section 2.3**. Handling parallel requests is something that a browser does in order to serve increase the efficiency and ultimately enhance the user experience. Chattiness, per se, is not something that is inherently negative. In a scenario where a chatty application resides on an internal network (LAN) where latency has little or nothing to say, described in **section 2.2.1**, the implication is rather non-existent. If one on the other hand, move out from the local network and need to take in consideration (and one does) the geographic distances between nodes on different continents i.e. a Wide Area Network (WAN), chattiness can become something of a nuisance.

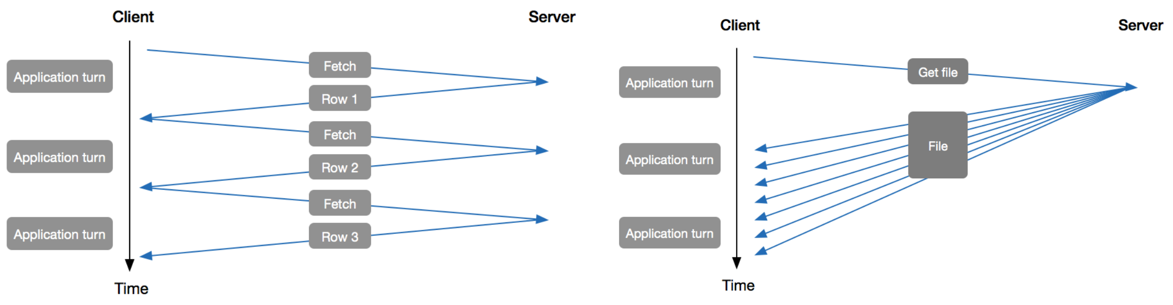


Figure 2 – Two examples of application chattiness

Figure 2, illustrates two simple communication examples. On the left, a client communicates with an SQL server, and on the right, a client communicates with an FTP server. The database application uses queries and the result from all of these queries is fetched one row at a time, resulting in many application turns and a chatty application. On the other hand, the second example provides the opposite of prior approach. The client wants to retrieve a file, and does so through one request, whereas the FTP server replies with sending the entire file. In the second approach, only one application turn is utilised and is therefore much more effective with regards to payload efficiency and application chattiness. For the means of analysing the performance of transactions, one must be able to distinguish the actual time it takes to make the request and receive the response, from the time the server and client needs to process them. **Figure 3**, expands the packet flow diagram preview earlier, with the addition of four additional categories to be considered when pursuing performance analysis. The server node processing metric starts being measured when the server has received the last request data packet from the client (1). The server processing delay is ended with the event of the first packet being sent from the server towards the client (2). Server node sending delay starts when the first packet is sent from the server and ends with the last packet in the response. This measurement is taken from the server's perspective and does not take in consideration the last package being received from the client. The same is true for the client processing time and sending delay.

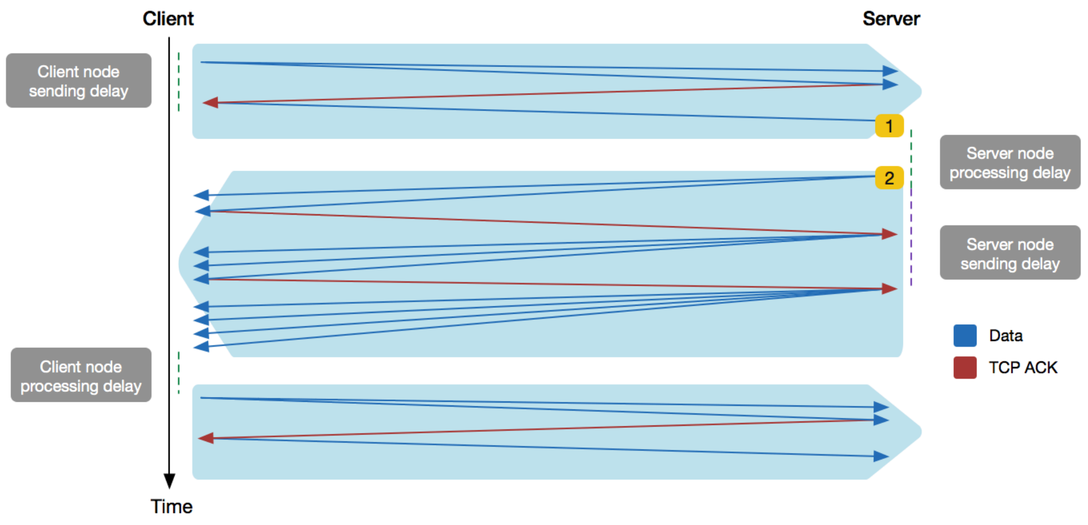

Client
Server
Client node sending delay
Server node processing delay
Server node sending delay
Client node processing delay
1
2
Data
TCP ACK
Time

Figure 3 - Packet flow diagram with network conditions

2.1 TCP/IP building blocks

Analogous to the series of steps conducted in **Figure 1**, is the way human beings communicate with each other. There are an inherent set of rules that one adheres, when communicating with another person. This set of rules will differ when moving from context to context e.g. speaking with a friend or speaking with a government official. Not abiding a set of rules will most certainly trigger some sort of reaction in either or both of the participants of a conversation. Computers, just like humans, need a set of rules to understand each other. This set of rules is often referred to as a protocol, and different protocols are invoked dependent on the different tasks one intends to perform. Utilising a reliable protocol that returns an acknowledgement (ACK) for each request received, when streaming a video would require much more data than required. In that particular situation, it would be more convenient to utilise a protocol that transmits, here streams, the video to a client without checking if the client receives the actual stream. When the stream is done, it automatically shuts down, not bothering with whether the client is there or not. This concept of a protocol is used throughout all networks, and is how remote entities are governed if they intend to communicate with each other (Kruse, 2013: 7-9). Without having any evidence to support the claim, it could be argued that the TCP/IP stack is the most basic and commonly used protocol stack on the Internet. From the point of view of performance analytics, there is an essential number of behaviours of the TCP/IP stack, that needs to be addressed in order to thoroughly understand the complexity of flow of transactions.

2.1.1 Three-way handshake

For an application to communicate with a remote host, a TCP connection must be established between client and server. Under this connection establishment client and server tells the receiving end what sequence number is to be used, TCP receive window size, Maximum Segment Size (MSS) and window scaling option amongst other things. With each packet sent back and forth a timestamp is added in order to facilitate computation.

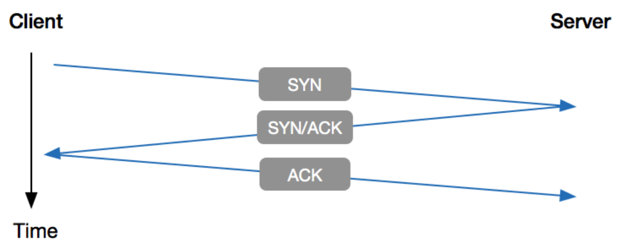


Figure 4 - Illustration of a three-way handshake

This three-way handshake is often referred to as SYN/SYN/ACK, which is the labels of the packet sequence taking place under the handshake as illustrated in **Figure 4.** The client sends a SYN packet in order to start the negotiation, followed by a SYN/ACK by the server, and finally by an ACK by the client. Once the ACK packet is sent from the client it can proceed with sending data to the server and the server needs to await the ACK package before it can proceed with dispatching any data to the client. The role of the three-way handshake applies to all connection establishments between client and server. Moreover, it plays a central role in performance as every connection will have a full round trip of latency before a server can transfer any data. The round-trip time (RTT), seen from a client's perspective is the delta between SYN and SYN/ACK. Whereas the round-trip time seen from as server's perspective is the delta between SYN/ACK ACK (ISI, 1981: 24-39).

2.2 Network Performance

The word performance at its core is "the action or process of performing a task or function" (Oxford Dictionaries). Nonetheless there is an intrinsic significance embedded which relates to performing the action in an effective way, thus performance also is about capabilities of this said action or process. In computer networks, such as the Internet, consisting of millions of nodes over which data packets need to traverse in order to reach their destination, there is an expectancy of these nodes not only performing, but doing so well. It is not an assumption that the structure, due to the democratisation of knowledge the Internet provides, has become more intricate over time and is increasing in complexity this very moment. When one speaks about performance in the context of web pages, the term performance is, amongst other things, correlated to the time it takes for a web page to be downloaded and displayed in a user’s web browser (Google 2010). The partitive relation of the expression performance and its child nodes, network performance and web page performance, are in many aspects tightly entwined but also have inter-nodal dependencies. One cannot have web page performance if one hasn’t got network performance. In order to address the hierarchy of expressions, one must start at the top to understand the various factors that have implications on network performance.

Many technology companies claim to have an answer to what the bottom line economical effect on slow web pages. Beating on the drum, the mantra that is chanted and repeated throughout the web is the one of performance or the lack thereof and the drastic effects it has on sales and conversion rates (Khan, F, 2015). Although speculative, Google is said to use about 200 metrics for page ranking, where Page loading speed via HTML, is one of the metrics utilized (several sources). While it’s note entirely sure which metrics Google is using the have stated that they have taken page speed into account in their search rankings (Google 2010). Furthermore, the company claims that users spend less time on slow sites than on faster ones (Google 2009). Others claim that site speed improvements are a source of cost reduction for operations.

But before continuing a few words about measurements. Network performance is measured in two fundamental ways: *bandwidth,* also called *throughput* and *latency,* also called *delay* (Peterson & Davie, 2012: 40). The terms latency and delay will henceforth be used interchangeably.

2.2.1 Latency

TheFreeDictionary.com defines the term as: “The time period between a request for a network to perform an action and the action being carried out” (The free dictionary, 2016). In a broader sense the gist of the term refers to the time delay between the cause and effect of a physical change in a system being observed. When breaking the definition down into smaller pieces one comes to the understanding that the term ‘latency’ is a word that spans over several disciplines. What is clear is that there is a stimulating end and a responsive end. The actual information that is stimulated and received needs to have a medium on which it is transported. Finally, due to the nature of stimulation and response, a time will have elapsed between the two endpoints, thus attributing to the relative understanding of said elapsed time and how it’s perceived.

Latency at its core is as simple as stated above, the reality is more complex though. For a deeper understanding, one consequently needs to segment latency to get a grasp of which points one can address. A network is basically two or more computers (nodes) communicating over some medium, be it air or cable, thus we have latency on the actual nodes, executing tasks, and between them when information is sent back and forth.

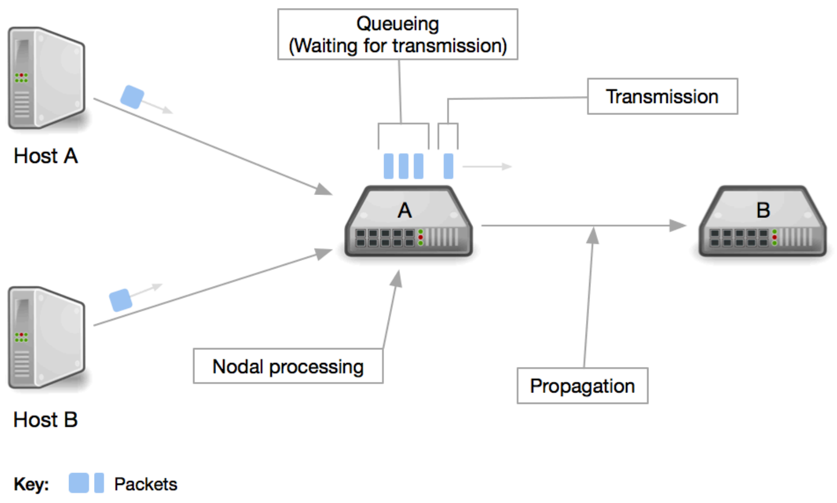


Figure 5 - Central nodal delay components

Figure 5, shows how data packets are sent over a network, and has the intention to demonstrate where different types of delay can occur. Measuring latency is a matter of measuring how long time it takes for a message from point A to point B. Measuring latency for a message from A to B to A again is called round-trip time (RTT) (NITA, ITS 1996).

At the lowest level, we have latency for the low-level infrastructure e.g. the CPU, memory I/O, disk I/O and network related I/O, i.e. nodal-delay(s). A higher level of latency follows, for the actual transmissions of data packets across the network and for each point they traverse, with processing delay, queue delay, transmission delay and propagation delay being most central (Peterson& Davie, 2012: 40-44). The accumulation of the two levels of latencies or delays, gives the total delay.

2.2.2 Processing delay

Packets sent from one of the computers at point A, **in Figure 5**, will be processed at point B, through an examination of the packets' headers to determine where they should be redirected to. This time and eventual other delays e.g. "the need to check for bit-level errors in the packet that occurred in transmitting the packet" (Kurose and Ross 2013: 36), constitutes the processing delay time.

2.2.3 Queueing delay

Since the packets can only be examined and forwarded one at the time, there is risk for queueing delay consequently, contention, congestion and packet loss if the queueing buffer is full (Peterson& Davie, 2012: 176-177). Packets to be transmitted on the link and haft to wait in a queue before being granted access, is said to be experiencing queueing delay (Kurose and Ross 2013: 37).

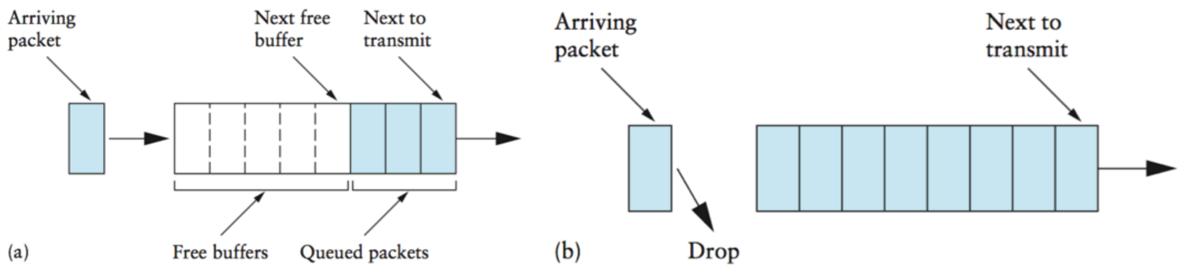


Figure 6 – Packet queue buffer and packet loss (FIFO)

2.2.4 Transmission latency

If one studies the logic of which packets arrive at the buffer in **Figure 6**, the packets side or length can be denoted L. The speed of which the packet is transferred over a link, here between point B and point C, as seen in **Figure 5**, is dependent of the speed of the link itself. The rate of the speed can be denoted R, and can for example have the dimension 10 Mbps. The denotation for transmission latency is therefore L/R, and represents the time it takes for all packets e.g. a message to be transmitted into the link (ibid.: 2013:37), that is from for the packages to pushed out of the router at point B. This is different, but easy to mistake from propagation delay described next.

2.2.5 Propagation delay

Propagation latency is the speed of the data traversing a link i.e. the time it takes for the first bit to travel from sender to receiver. For the bit to travel at higher speeds it is dependent on the physical medium on which it propagates. Speeds range between $2 \cdot 10^8 \frac{m}{s}$ to $3 \cdot 10^8 \frac{m}{s}$. The delay itself is calculated by dividing the distance d between two endpoints, with the speed, s, the propagation speed of the link, d/s (ibid.: 2013:37). Reducing the propagation delay is therefore a matter of increasing the speed of the link or shortening the distance between two endpoints.

2.2.6 Total delay

To allow the concept of delay to be graspable, one must not focus entirely on the parts of the whole but also on how these parts contribute to affecting the whole itself. If one adds the individual delays of each contributor a formula for calculating the total nodal-delay can be denoted, $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$. The individual parts are, as described above, process, queueing, transmission and propagation delays (ibid.: 2013:39).

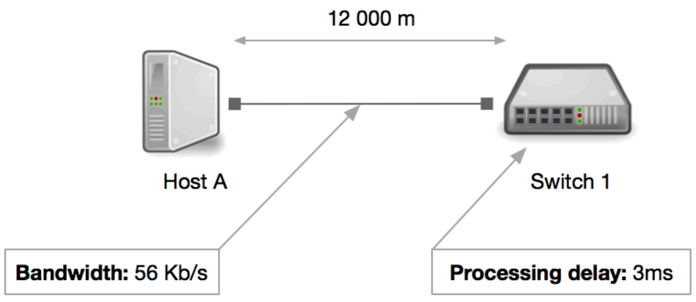


Figure 7 - Network delay contribution factors

If you, for example want to calculate the propagation time for 10 packets, each being 1Kb in size, from “Host A” to “Switch 1” as displayed in **Figure 7**, you could utilise:

Equation 1 - Nodal delay

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} = d_{proc} + \frac{Q}{R} + \frac{L}{R} + \frac{d}{s}$$
$$Q = Queue\ depth\ (bits)$$

Assuming the following setup for an example (UCalgary, 2014)

- 1000-byte packet to be sent to ISP which is 12 km away
- Dialup modem: 56 Kbps
- Processing: 0.003 sec (laptop) = 3ms
- Queuing: 0 sec

Equation 2 - Example calculation of nodal delay

$$d_{nodal} = d_{proc} + \frac{Q}{R} + \frac{L}{R} + \frac{d}{s} = 3ms + 0ms + \frac{1000\ bytes \cdot 8\ bytes}{56000\ bps} + \frac{12000\ m}{2 \cdot 10^8\ m/s}$$

$$d_{nodal} = 142{,}8ms + 0{,}06ms = 142{,}86\ ms$$

The example illustrates that the transmission time dominates quite profoundly, this is not always the case though. As the example utilises the transmission of data over a 56Kbs modem, a technology that is outdated many years ago, it is not at all surprising that the transmission contribution is overshadowing that of the propagation. For other areas, this is not the case though; sending the same amount of data over satellite link, where the distance is far greater, has a much more impact on the propagation delay.

2.2.7 Bandwidth delay product

When a client communicates with a server, the manner of which they communicate is a predefined set of rules called a protocol. Furthermore, the client and server must negotiate, with that basic set of rules as a starting-point, on which options to use for said communication e.g. maximum segment size of the packet sent. With this in mind and regards to setting up networks and optimising their performance, one has to take in consideration the amount of data a sender transmits before the first bit arrives at the other side. To address this concept, one can use the product of bandwidth multiplied with delay, see **Figure 8.** Here the latency or delay, is the length of the pipe and the bandwidth is the width. The volume of the pipe corresponds to the number of bits that can be in transit through the pipe at any given time (Peterson& Davie, 2012: 44-46). If for example, a communication link has a bandwidth of a 100 Mbps and the latency for that particular distance is 20 ms, the bandwidth-delay product (BDP) is $100 \cdot 10^6 \cdot 0{,}2 \cdot 10^{-3} \approx 200 \cdot 10^6\ bits$ or 195 kB ($200 \cdot 10^6/1024 = \ 195 \cdot 10^3$).

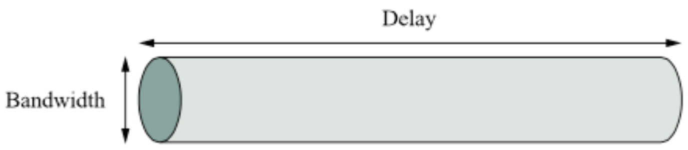


Figure 8 - Bandwidth x delay product

As for the use of TCP/IP as a communication protocol stack, two-way communication is the basis for all interactions between sender and receiver. When a sender expects a receiver to acknowledge the data packets sent (the sender does!), it takes another latency before the signal can reach the sender. This means that the sender can send up to two BDP-volumes of data before hearing anything from the receiver. For the example above the amount of data would accumulate to 390 KB and the RTT is 40ms. Additional examples are given in **Table 1**.

Table 1 - Sample delay x bandwidth products

Link type	Bandwidth (typical)	Distance (Typical)	Round-trip delay	Delay x latency
Dial-up	56 Kbps	10 km	87μs	5 bits
Wireless LAN	54 Mbps	50 m	0.33μs	18 bits
Satellite	45 Mbps	35 000 km	230ms	10 Mb
Cross-country fibre	10 Gbps	4 000 km	400ms	400 Mb

A limitation that one must bear in mind, that affects the BDP, is the speed of which light operates. We cannot surpass that limit, nor can we change the laws upon which the speed is based. Furthermore, one can have in the back of the head Nielsen's law that dictates that the rate of which the users' bandwidth increases is 50% per year (Nielsen, 1998). To illustrate the implications of the limit of latency in correlation to bandwidth, one can take a look at the table above. The latency for a cross-country fibre with the distance of 4 000 km between point A and B, can only be reduced if the distance itself is reduced. This implies that as networks bandwidth increases, the round-trip delay rather than the bandwidth, will have an ever-growing impact on the network. An example to illustrate this scenario would be to compare sending a 1 MB file over the distance of 4 000 km with a 1 Mbps connection and sending the same file over a 1 Gbps connection. If it assumed that the data travels over the 4 000 km at the speed of $2 \cdot 10^8 m/s$, the latency in one direction would be:

Equation 3 - Latency equation for 4 000 km distance

$$latency_{4kKm} = \frac{4 \ \cdot 10^6 \ m}{2 \cdot 10^8 \ m/s} = 20 \ ms$$

Therefore, the round-trip delay would be two times the latency i.e. 40ms. Another viable scenario is depicted in **Equation 4**, would be to transfer the same 1 MB file with over the distance of 35 000 km with a 45 Mbps connection resulting in a round trip delay of 234ms and a latency of:

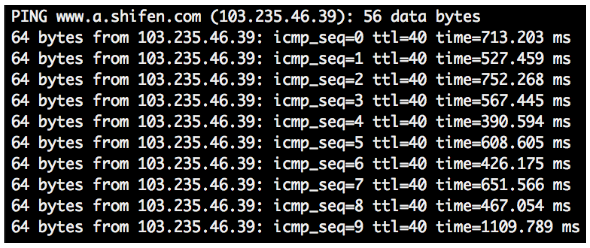
Equation 4 - Latency equation for 35 000 km distance

$$latency_{35kKm} = \frac{35 \ \cdot 10^6 \ m}{3 \cdot 10^8 \ m/s} = 117 \ ms$$

Table 2 - Bandwidth - latency relations

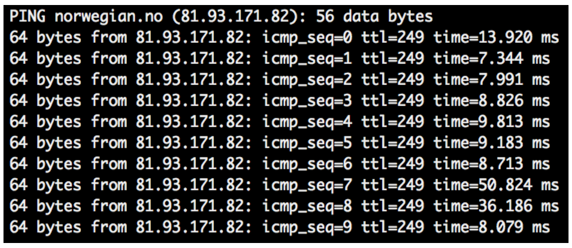
Bandwidth	Transmit time	RTT	Transf. time	Throughput (transf. size/transf. time)	BW x delay
1 Mbps	$\frac{8 \cdot 10^6 \ bits}{10^6 \ bits/s} = 8000 \ ms$	$\frac{8\,000 \ ms}{40 \ ms} = 200$	$8040 \ ms$	$\frac{8 \cdot 10^6 \ bits}{8{,}04 \ s} = 0{,}995 \ Mbps$	40 Kb
45 Mbps	$\frac{8 \cdot 10^6 \ bits}{45 \cdot 10^6 \ bits/s} = 178 \ ms$	$\frac{178 \ ms}{233 \ ms} = 1$	$411 \ ms$	$\frac{8 \cdot 10^6 \ bits}{0{,}411 \ s} = 19{,}45 \ Gbps$	10,5 Mb

The effect distance has on latency can be seen when utilising the *ping* utility. The two different *pings* are made from Norway as illustrated in **Figure 9** and **Figure 10**. The figure on the left targets www.baidu.com in China, thus resulting in longer response times than the *ping* in the right figure which targets www.noregian.no in Norway. In the **Table 2** above, it can be seen that the 1 Mbps connection is more effective when it comes to its throughput utilising 99,5% of its capacity, whilst the 45 Mbps connection only utilizes 43,2% of its total capacity. Following the logic, it would mean that the 45 Mbps connection is much more sensitive to loss of packets, that is if a packet gets lost in transit it will have to be resent. Thus, resulting in having the user wait a longer time relative to the user receiving packets over a shorter distance. For example, since the bandwidth delay product for the link is 10,5 Mb, the window size of the receive buffer negotiated for the TCP/IP connection, would optimally be 312,5 KB $(1050000/8 = 312500\ bytes)$, meaning that the entire 312,5 KB would need to be retransmitted. Thus, dramatically lowering the throughput. This by itself is an important insight, though not the main area for the coming work. Instead the focus with regards to latency and bandwidth within this thesis lies around how a webpage loads and the effects of having a badly structured web site.



```
PING www.a.shifen.com (103.235.46.39): 56 data bytes
64 bytes from 103.235.46.39: icmp_seq=0 ttl=40 time=713.203 ms
64 bytes from 103.235.46.39: icmp_seq=1 ttl=40 time=527.459 ms
64 bytes from 103.235.46.39: icmp_seq=2 ttl=40 time=752.268 ms
64 bytes from 103.235.46.39: icmp_seq=3 ttl=40 time=567.445 ms
64 bytes from 103.235.46.39: icmp_seq=4 ttl=40 time=390.594 ms
64 bytes from 103.235.46.39: icmp_seq=5 ttl=40 time=608.605 ms
64 bytes from 103.235.46.39: icmp_seq=6 ttl=40 time=426.175 ms
64 bytes from 103.235.46.39: icmp_seq=7 ttl=40 time=651.566 ms
64 bytes from 103.235.46.39: icmp_seq=8 ttl=40 time=467.054 ms
64 bytes from 103.235.46.39: icmp_seq=9 ttl=40 time=1109.789 ms
```

Figure 9 - Ping output for www.baidu.com



```
PING norwegian.no (81.93.171.82): 56 data bytes
64 bytes from 81.93.171.82: icmp_seq=0 ttl=249 time=13.920 ms
64 bytes from 81.93.171.82: icmp_seq=1 ttl=249 time=7.344 ms
64 bytes from 81.93.171.82: icmp_seq=2 ttl=249 time=7.991 ms
64 bytes from 81.93.171.82: icmp_seq=3 ttl=249 time=8.826 ms
64 bytes from 81.93.171.82: icmp_seq=4 ttl=249 time=9.813 ms
64 bytes from 81.93.171.82: icmp_seq=5 ttl=249 time=9.183 ms
64 bytes from 81.93.171.82: icmp_seq=6 ttl=249 time=8.713 ms
64 bytes from 81.93.171.82: icmp_seq=7 ttl=249 time=50.824 ms
64 bytes from 81.93.171.82: icmp_seq=8 ttl=249 time=36.186 ms
64 bytes from 81.93.171.82: icmp_seq=9 ttl=249 time=8.079 ms
```

Figure 10 - Ping output for www.norwegian.no

2.2.8 Throughput

To transmit bits and bytes over a communication link, the link in question has physical properties e.g. copper, fibre and so on, furthermore one speaks about the bandwidth of said link. The term refers to the number of bits one is able to transmit on the link e.g. 10Mbps, 1Gbps and so on. Ideally the number of bits on the link corresponds to the capacity of the link itself, this is not always the case, thus it becomes essential to measure the performance of the link – the throughput. If for example, one only populates 100Mbps over a 1Gbps link one would only utilise 10% of the existing capacity, meaning that at that given point in time 100 million bits fill a “pipe” that has the room for 1000 million bits.

In a simplified network with two end points, a server and a client and two communication links connected with a router, as illustrated in **Figure 11**, one can study the throughput and the parts that constitutes it.

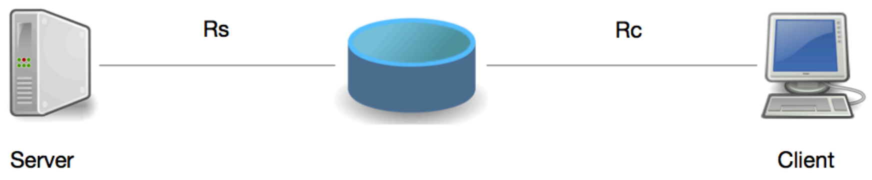


Figure 11 - Simple network setup

The rate of which the server sends a data packets to the router at the first end of the link can be denoted as R_s and the rate the router sends the same data packets to the client is denoted as R_c. The speed of which the server can send packets, is limited to the properties of the actual communication link, that is R_s bps. The corresponding thing applies for the client and its preceding link. Therefore, if $R_s < R_c$, the bits will travers the router without any friction and arrive at the client with the speed of R_s. On the other hand, if $R_c < R_s$ the router will have to queue the bits in its buffer, waiting for the client to allow more bits to pass through, giving the possible throughput of R_c. If for example a file has the size 44 million bits, the transmission rate of $R_s = 2\ Mbps$, and the corresponding rate for $R_c = 1\ Mbps$, then it would take 44 seconds to send a file from server to client, not accounting for any packets drops or other types of delays (Kurose and Ross 2013: 44-45).

2.3 Web content performance

According to a network research group at Virginia Tech, *"roughly half of the time is spent from the moment the browser sends the acknowledgement (ACK) completing the TCP connection establishment until the first packet containing page content arrives"* (Habib & Abrams, 2000)[1]. The research of Habib and Abrams refers to the wait time, that is the time before the first byte (TTFB) after the connection between client and server has been set up, for the 290 top sites at that moment. It must be stated that the research was performed in such a way that the tests conducted, had an origin in the United States and some of them had a destination overseas. This resulted in the skewing of average time before first byte if the sites were not cached, but nonetheless the results were quite significant due to the fact that the time before fist byte comprised 40-60% of the total connection setup time. The interpretation of the findings in the research suggests that the servers hosting the web pages are the bottlenecks and should thus be the primary target when optimising the end-user experience, as seen from a connection point of view. Although this might have been the circumstance in the beginning of the 21st century, this is far from the reality of the modern-day web page.

[1] The download time in the study was divided into: DNS query, connection setup time, time to first byte (TTFB) and downloading time.

The webpage has evolved hand in hand with the evolution of web technology and the democratisation of the knowledge connected to it. The transformation of web pages and the contents therein, has changed from being static to having a more dynamic and complex structure. More precise they started out as hypertext documents and have reached the state of web applications. To strengthen these statements, one can analyse the contents and size of webpages. The mean response size for webpages grew between 2000 to 2007, from 13 kB to respectively 68 kB. Furthermore, the very nature of the contents between these years, shifted from being image and text centric to becoming almost entirely made up out of video and binary files. Notable is that the size of images on average increased by 30% (Sadre and Haverkort, 2008).

Table 3 - Fraction of traffic volume and average response size by type for the year 2000 trace (left), year 2007 trace (right)

Type	Volume	Size (Kb)
image/jpeg	21.5%	10
image/gif	15.5%	4
text/html	14.6%	9

Type	Volume	Size (Kb)
application/octets	34.6%	1776
image/jpeg	6.6%	13
application/x-otrkey	6.6%	240610

It is not difficult to discern that the ever-increasing size of broadband is resulting in larger and larger web sites, something that becomes clear when analysing the average total transfer size and the number of requests of the top 300 000 plus web sites. **Figure 12,** shows that the total transfer size has risen from 702 kB in 2010, to about 2 300 kB in the end 2016. At the same time the number of HTML requests have almost doubled. To understand the implications this has on the end-user experience one has to understand how a modern web application is built up and what one is able to affect within its domains.

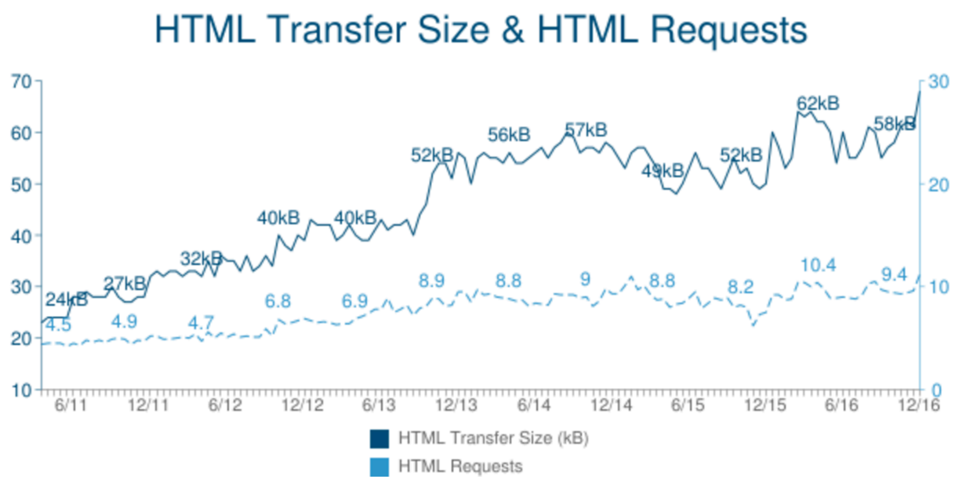


Figure 12 - Total transfer size & total requests from Nov 10 to May 16 (HTTP Archive, 2017)

2.3.1 Anatomy of a modern web application

As stated earlier, early webpages were hypertext documents. These documents were built up by static text, links and images. Morden pages, however are built up out of mark-up, the basic structure of a webpage; stylesheets, the layout of the page; and scripts which allow for interactivity and response of user input. In the **chapter 2.3.3**, more detailed information on the interaction between these three elements is reviewed. Furthermore, the applications and the pages they reside on are evolving.

Table 4 - Anatomy of average web application Nov -10 (left) and May -16 (right)

Type	Requests	Size (kB)
HTML	5.6	34
Images	48	416
JavaScript	11	113
CSS	3.5	25
Other	4.9	114
SUM	73	702

Hosts involved	10

Type	Requests	Size (kB)
HTML	11.7	64
Images	57	1425
JavaScript	21	387
CSS	6.8	70
Other	12.5	366
SUM	109	2312

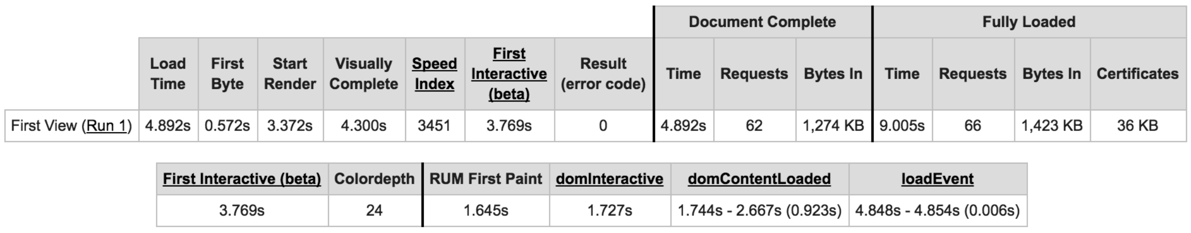
Hosts involved	20

Table 4, gives a picture of the transformation of web applications and how they are dependent on more hosts in order to deliver contents to the end-user (HTTP Archive, 2017a). Given this increased complexity it becomes a concern, although probably not a clear one, that in order to address what on the surface looks like a webpage one needs to break down the whole into distinct components and address these individually. Doing this requires a deeper understating of how these parts are built up, after all the process of loading the average web application consists of communicating with 20 different hosts through 109 requests and downloading about 2300 kB in a matter of a couple of milliseconds.

2.3.2 Resource waterfall

According to Steve Souders, less than 10-20% of the end user response time is spent on getting a HTML document from a server to the client, this time is here defined as backend time and is mostly represented by the time elapsed, described in **chapter 2.2.1** (Souders, 2007:3-5). This directly implies that the rest of the time, 80-90% is spent on making HTTP requests, downloading the remaining components and rendering them on the display. This time is defined as frontend time. The act of uncovering what this 80-90% are comprised of will not only provide deeper insight, but allows for intended parties to focus their efforts on resolving frontend application issues. To help aid the understanding of how a web page is loaded and where on the frontend and backend the time is actually spent, there are several tools at one's disposal. One of the most powerful is the resource waterfall as shown in **Figure 15.** There are many different representations of the resource waterfall, in fact all the major web browsers enable for resource timing, that is measuring how long it takes for a resource to get loaded. For this assignment WebPagetest is chosen as it smoothens the inconsistency gap that would have existed between the different browsers, had one chosen to compare them (WebPagestest, 2017). The website WebPagetest lets users explore how other websites behave when loading them. In order to be able to explore the behaviour (and structure) of web sites one needs to fill in the URL of the website one wants to visit, from where one wants to simulate this visit and which type of browser one wants to utilise for the simulation. On the backend performing this action of surfing to the named URL is, what is called a synthetic agent, a script designed to do certain steps. Once the answer comes back, the behaviour of the named URL is broken down into requests for each resource that is included for said URL. As discussed earlier, these resources are images, fonts, scripts etc. Each resource takes a certain amount of time to load and the total time dimension is broken down into many different metrics.

The waterfall chart in **Figure 15** shows all the HTTP requests (represented as rows) for a page and the stages for these requests. This particular visit was made connection browser dialup method as displayed in **appendix section 10.1** Examining the first row, one can see that the initial request is comprised of *DNS lookup*, *Initial connection*, *TTFB* and *Content download*. Usually also included in this initial request is *SSL time*, this can be seen on second row, as the user is redirected from http://www.norwegian.no to https://www.norwegian.no. The request before we reach the *TTFB* on the second row (light blue color), takes 572ms. Out of these 572ms, 466ms (124ms +342ms) is spent on waiting for the network, that is about 93% of the time spent on latency. This goes against the proposition made by Souders, but does not reveal the truth in its entirety. Analysing a bit further we can **in Figure 13** and **Figure 15**, see that that the web page makes 66 requests from 10 different hosts, adding up to 1 423 KB. Comparing this result to **Table 4 ,** the webpage reaches about half the values of the average web application in the HTTP Archive. Furthermore, one can see in the resource waterfall chart that requests are being made as the content of the web pages is being downloaded. These requests are made incrementally, allowing the browser to discover the resources required resources at an early stage and forward them in parallel if needed. The consequence of this is that different structure for the page mark-up results in different outcomes (Grigorik, 2013:171-175).

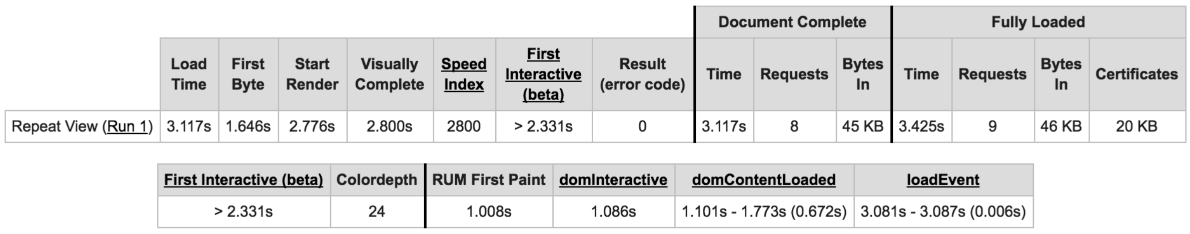


								Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In	Certificates
First View (Run 1)	4.892s	0.572s	3.372s	4.300s	3451	3.769s	0	4.892s	62	1,274 KB	9.005s	66	1,423 KB	36 KB

First Interactive (beta)	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
3.769s	24	1.645s	1.727s	1.744s - 2.667s (0.923s)	4.848s - 4.854s (0.006s)

Figure 13 – "First hit" summary of resource waterfall for www.norwegian.no

As noted, the waterfall chart indicates how a user would experience a web page the first time they visit the actual page. But how about when a user returns to the web site? Will the experience be the same? Better? The answer lies in the setup of the machine that is serving the web content. A setup where caching is allowed would allow for the user's browser to cache, or store, certain resources on the visited website. This means that certain resources will not be downloaded on a recurring visit, thus resulting in a hopefully better user experience. A repeat view, or a "cached hit" is displayed in **Figure 14**, and shows that decrease in *Load Time* from 4 892ms to 3 117ms, a 36% decrease. In fact, enabling for cached websites is one of the remedies for slow websites, discussed in the book High Performance Websites by Steve Souders (Souders, 2007:22-27).



								Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Visually Complete	Speed Index	First Interactive (beta)	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In	Certificates
Repeat View (Run 1)	3.117s	1.646s	2.776s	2.800s	2800	> 2.331s	0	3.117s	8	45 KB	3.425s	9	46 KB	20 KB

First Interactive (beta)	Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
> 2.331s	24	1.008s	1.086s	1.101s - 1.773s (0.672s)	3.081s - 3.087s (0.006s)

Figure 14 - "Cached hit" summary of resource waterfall for www.norwegian.no

In **Figure 15**, there are many metrics one can utilise, but not all are highlighted in this thesis. The green vertical line represents the "Start render", and is the time when the user can start interacting with the page, whereas the blue line represents "Document complete". The firing of the event "Document complete" means that all the static content of a page is loaded, this is when the user perceives the page to be fully loaded, but this is not always the case. Web pages and applications continue to load content on the backend, usually through JavaScript or some other means of communicating with the server. This is called asynchronous communication.

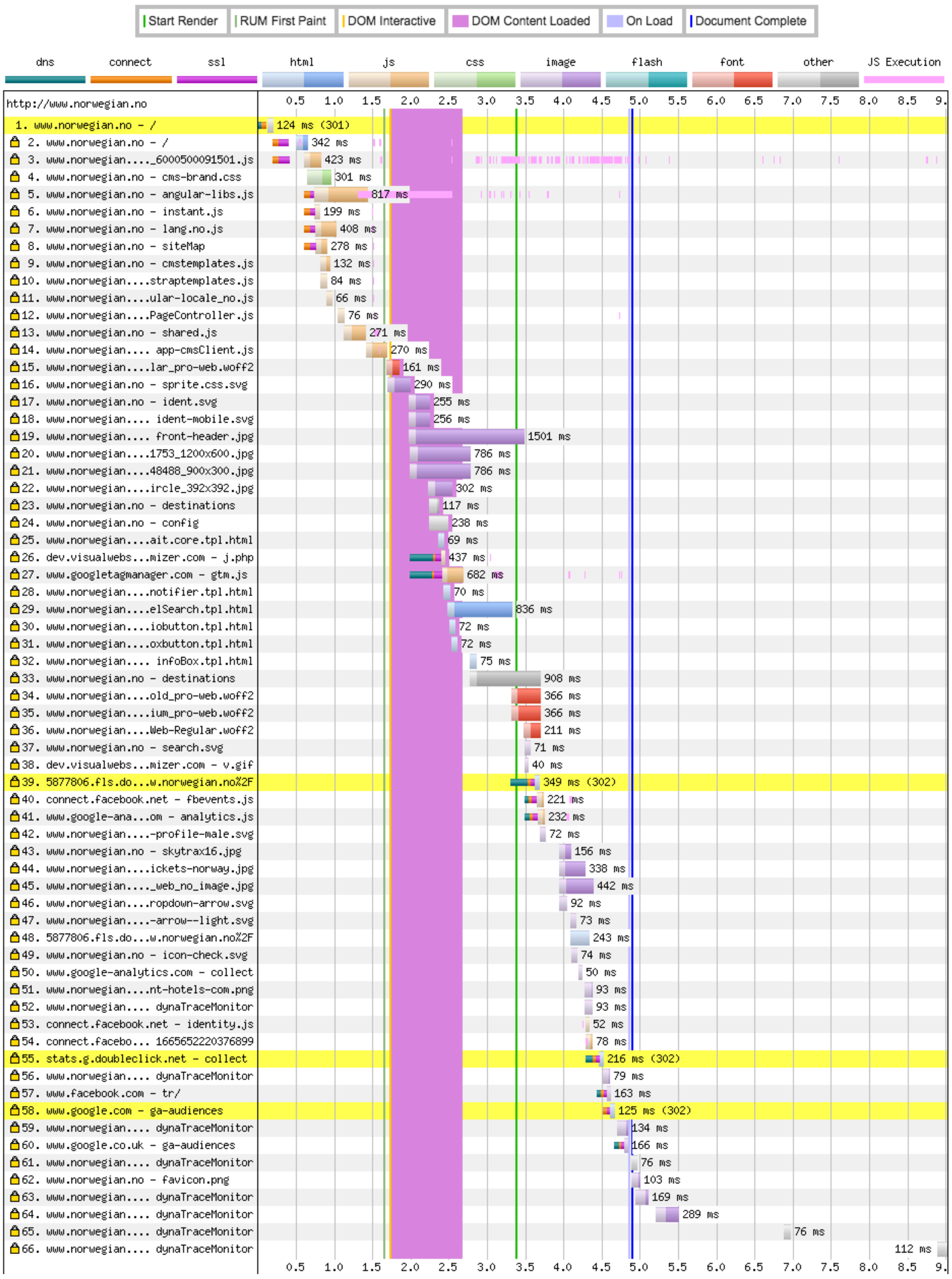


Figure 15 - Norwegian.no Resource waterfall

2.3.3 The DOM and critical rendering path

In order to display webpages or web applications, there is a need for the browser to know what to present and how to render it. This is done by a series of events based on a predefined set of rules, recommended by the World Wide Web Consortium (W3C). The use of the word recommended, is because of the autonomy of each browser vendor, meaning they do not have adhere the suggested. This is of course a dilemma in the browsing world, a dilemma that poses many problems for web developers as not all vendors follow the recommendation but chose a different path. Since a webpage in its essence is comprised by HTML, CSS, images and JavaScript, the path itself refers to the steps browsers take in order to render the outcome, utilising these four building blocks. What is the reason for this, and more important why is this important? Even if webpages are only comprised of a small set of building blocks, developers exploit the possibility to use many instances of these building blocks. How the mark-up is structured has implications on how fast the page loads and consequently how the end-user perceives the page speed. As has been shown in the article "User Preference and Search Engine Latency, slower webpages result in higher risk for one's users to leave for other webpages e.g. competition (Brutlag, Hutchinson & Stone, 2008).

Figure 16, shows the elements that are included in the process a browser needs to take in order to display a web page. The figure illustrates the parsing of HTML and CSS into to two objects models – the Domain Object Model and the Cascading Style Sheet Object Model.

The Domain Object Model (DOM), *…is a programming API for HTML and XML documents* and …defines *the logical structure of documents and the way a document is accessed and manipulated* (W3.org, 2000).

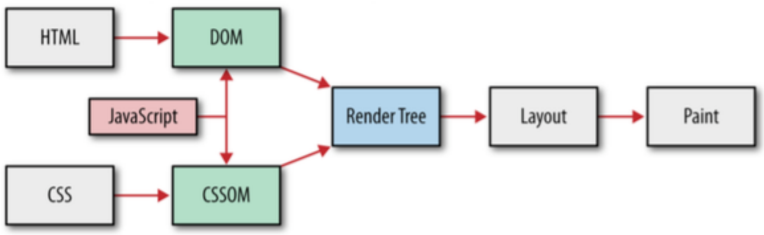
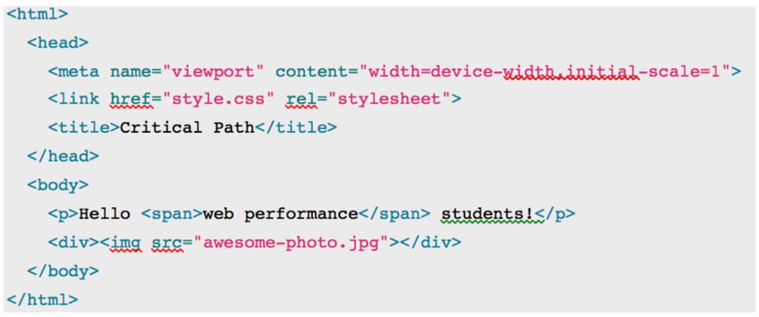
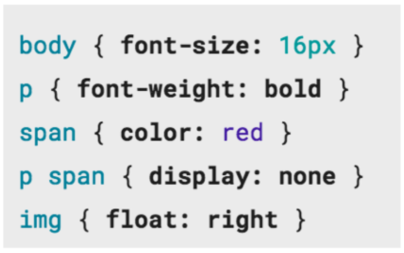


Figure 16 - Browser processing pipeline

An example of a simple HTML page is given in **Figure 17** and its purpose is to aid in the understating of how the browser processing timeline is being traversed. The code is the starting point from which the browser constructs the object model. The same is true for Cascading Style Sheets the browser encounters throughout the HTML page. The example provided includes only one external CSS file, namely style.css.



```html
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link href="style.css" rel="stylesheet">
    <title>Critical Path</title>
  </head>
  <body>
    <p>Hello <span>web performance</span> students!</p>
    <div><img src="awesome-photo.jpg"></div>
  </body>
</html>
```



```css
body { font-size: 16px }
p { font-weight: bold }
span { color: red }
p span { display: none }
img { float: right }
```

Figure 17 - HTML code example (left), CSS code example (right)

For the HTML mark-up to results in a DOM, the browser reads raw data, the bytes that represent all the different combination of zeroes and ones, and converts it into characters. The characters are turned into tokens, which is the browser way of knowing where a tag starts and a tag ends, thus also finding out the parent-child relations between these tokens. The tokens are all consumed and converted into nodes, which are the building blocks that make up the Document Object Model. Once all tokens are converted into nodes, one arrives at the DOM structure a web browser uses to process a webpage.

Figure 18, illustrates a structure for a HTML example. It has been specified before, but must be noted again that the browser builds the DOM incrementally, meaning that the HTML is parsed as soon as tokens arrives. The process can be thought of as a queue where the FIFO principle is applied. During the analysis of the mark-up the browser encounters a CSS resource, in this case an external one, and needs to send a request to fetch that particular resource. The same would have been true if one had a page with several different CSS resources. Since the DOM captures the different properties and relations between the elements in the document mark-up, there is a need for the browser on how to render said elements. Enter CSSOM.
The CSSOM acts in the same way the DOM does in that it converts raw data into characters and so on, see **Figure 18 (right).** The CSSOM does however not parse the incoming tokens incrementally. The reason for this is that, as implied by the name, that the style sheets are cascading. Inspecting the CSS code in **Figure 17**, the code will be read in such a manner that the browser takes the first line then converts it into a token, then continues to line two converting that and so on. For the example provided, the browser reaches line two p { font-weight: bold }, which interprets into all <p>-tags being bold, noteworthy is that the <p>-tags inherit the font-size from the parent <body>-tag, which says that the size of the font should be 16 pixels. Moving on to line four p span { display: none }, meaning that the <span>-tag within the <p>-tag should not be displayed, as represented in **Figure 18 (right).** Had he CSSOM parsed the information in the same manner as the DOM does, it would mean that each line would be rendered upon receiving tokens, resulting in longer load times for webpages. Thus, the browser blocks page rendering until it receives and processes all of the CSS, this is called render blocking.

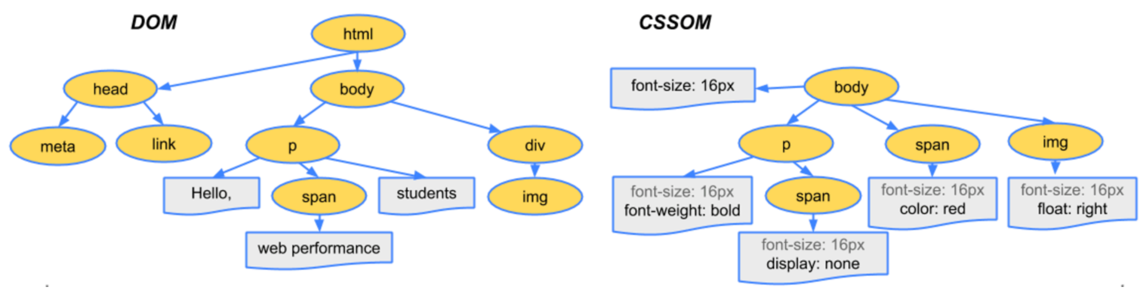


Figure 18 – Document Object Model (left) and CSS Object Model (right) for the example webpage

When the browser has traversed both object models it reaches a step called the render tree. For a browser to reach its destination, that is the render tree, it in a broad sense traverses all the elements in the DOM tree, including only what will be visible on the webpage. This means that all <meta> and <link> tags are ignored, here the line p span { display: none } results in the tag not being displayed. For each visible node in the DOM, a corresponding node is matched with the CSSOM and the rules are individually applied. Once that is done as in **Figure 19**, the nodes are released for the next step – the layout.

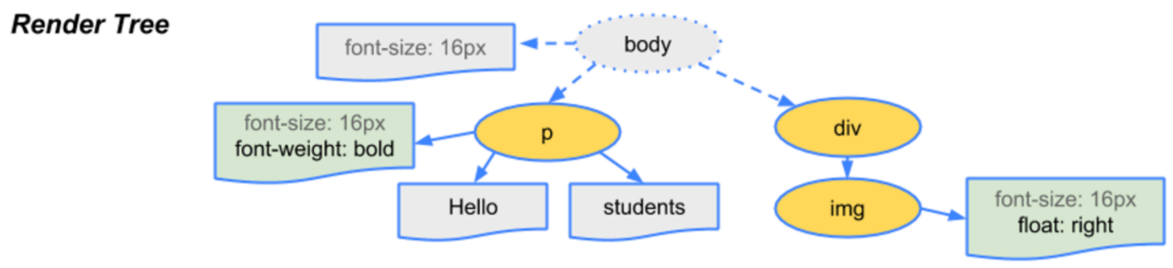


Figure 19 - Render tree from the DOM and CSSOM

The layout step in the browser processing pipeline has the responsibility for calculating the geometry and placing the different elements of the webpage in the correct positions. The node affected by this step, in the example provided, is the image file which is meant to float on the right-hand side of the web browser. The browser can of course be more precise that that, placing desired elements in specific positions e.g. at 20% from the left-hand side or 42 pixels from another image and so on. After the browser has traversed the DOM and CSSOM, produced a render tree of all the visible elements, calculated the geometry and placed each element where in its designated position the web page can be painted (Google Developers, 2017).

The subject not addressed yet is JavaScript and how it affects the rendering of a web page. Per default JavaScript is render blocking, exactly like CSS. To make things a bit more complicated, the positioning of the actual script in the HMTL mark-up can affect the performance of webpages. A script is executed where it is encountered in the mark-up by the HTML parser, resulting in a standstill for the DOM construction. The browser waits until the JavaScript engine is done with the script before it picks up and continues constructing the DOM. If the script contains any elements that manipulate the CSSOM, the browser needs to wait until the CSSOM is downloaded and constructed before it continues executing the script, furthermore the DOM construction is blocked during these two steps. Placing a JavaScript at the bottom of the HTML markup, e.g. <script src=”myScritp.js”></script>, gives the browser time to process both the DOM and the CSSOM, but it still blocks rendering until the script is fully executed, because it does not know what the script will do. Adding the async keyword to the script tag e.g. <script src=”myScritp.js” async></script>, results in the browser not taking any consideration to the actual script execution, resulting in a quicker rendering of the web page. The loading and structure of the elements discussed in this section, affect how long time it takes to load a web page and thus also the response time of said page.

2.3.4 Performance Timings

The critical rendering path aid developers and analysts to understands what is going on behind the scene and implicitly gives the them the possibility to design webpages and applications in an optimised way. One thing that has been disregarded on purpose in the section above, is the measuring metrics needed to establish a foundation based on facts. This is where timings play a central role. Timings is the backbone of, and an important quality benchmark for web applications. There are several aspects to timings because of the very nature of what is to be timed. What have been stated earlier is the need to understand one’s customers from the perspective of the customer, implicating the need to be as close to the customer as possible to get a picture of what is being experienced. The W3C group has introduced what is known as PerformanceTiming interface, a JavaScript API. The purpose of this interface is to act as mechanism that provides client-side latency measurements within applications, and is referred to as *Navigation Timing* and is illustrated in **Figure 20**. The big benefit of navigation timing is that is presents data related to the previously covered topics such as DNS and TCP connect times. Like navigation timing, there are other timings that are important that one as a developer or analyst needs to take in consideration when developing web applications. *Resource Timing* and *User* Timing are the ones with highest importance with regards to web performance. The combination of these three timing APIs is what is needed to conduct real-user performance measurements (W3C, 2012). If, for example, one wants to measure the time required for a DNS lookup for a particular web application, the time delta between *domainLookupEnd* and *domainLookupStart* gives the correct value. The same applies if one wants to know how long a request takes i.e. the difference between *responseEnd* and *requestStart*.

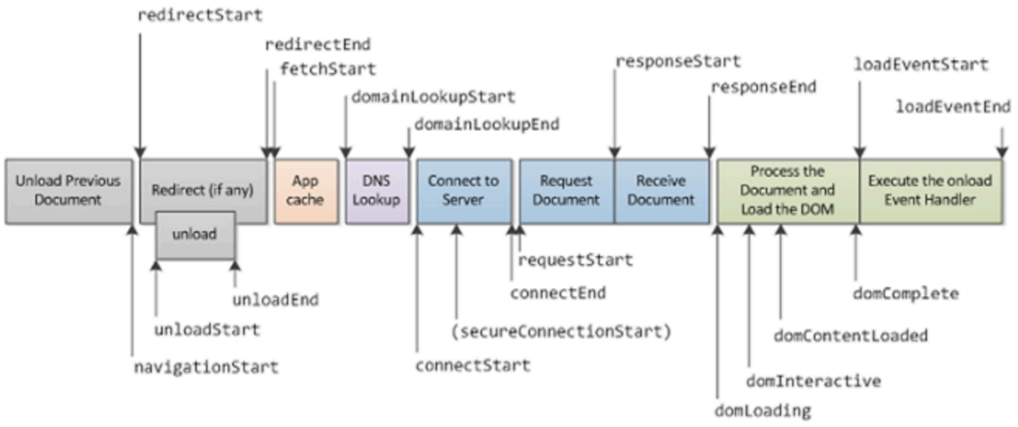


Figure 20 - Navigation timing and its attributes

The attributes above provide a foundation from which a myriad of measurements can be made. The measurements consequently serve as vessels for baselines, charting and analytics. In combination with, we will see this in later sub-chapters, infrastructure health metrics one can get a much clearer picture of how the environment is doing in the context, or rather with the eyes of the user. For a more detailed explanation of the attributes see in the **appendix section 10.1.**

2.4 User perception

Within the chapters of this thesis, performance has been and will continue to be a central topic. Undoubtedly it is safe to state that performance in most areas drives, amongst others, user satisfaction. Entire fields of study are dedicated to exploring this subject and, not surprisingly many if not all fields are predisposed in their hunt for better performance; it lies in the nature of mankind.

From the point of view of an engineer, one can argue the lack of importance of the user perception of time with regards to interacting with computer software. After all a second is a second. However true the statement is, it's safe to assume that, although time can be measured objectively, the experience of it is purely subjective. Thus, a dimension is augmented, the need to design or engineer time, or the perception thereof carries with it not only the feeling of the speed of which actions are taking place, but also implicit productivity and costs attached to it.

The Technology Acceptance Model, **Figure 21**, (Davis et al., 1989:982-1000) and other like it, tries to explain how users perceive new technologies and use said technologies based on their perception. More specifically the model tries to relate the perceived usefulness and perceived ease of use, to acceptance and sustained use of technologies.

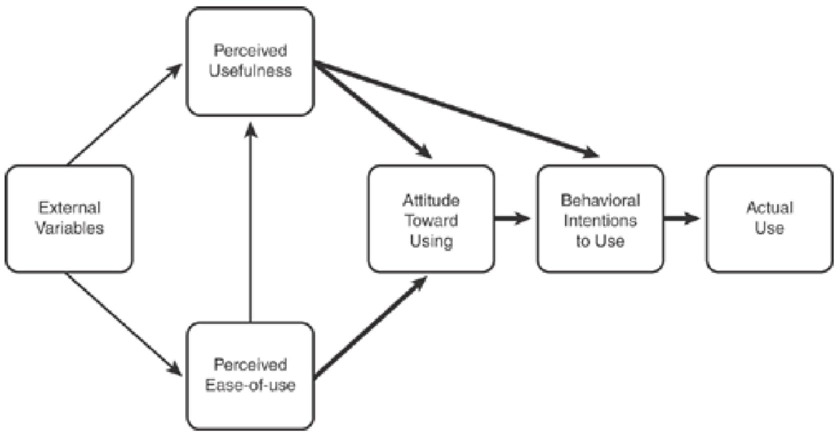


Figure 21 - Technology Acceptance Model (TAM)

Although there are many factors such as usability, aesthetics and technology, that drive user satisfaction, what glues these components together is the perception of them.

2.4.1 Perception and tolerance

Three main areas are of importance when approaching users and what they perceive when utilising software.

1. An operational management approach
2. A perception management approach
3. A tolerance management approach

Whereas the operational approach relies on configuring and tweaking the infrastructure, its components and processes in order to increase performance, perception management deals with diverting people's attention in order to reduce the perceived time at hand. In the context of operational management would relate to making a website faster by the means of scaling, upgrading hardware, utilising better technology and such in order to reduce actual response times. While objective timing concerns facts, and does not take into account if time elapsed is sensed or not, perception management concerns how individuals experience that objective timing. The last item on the list, tolerance management, deals with the attempt to disguise delays occurring; the subject falls outside the scope of this thesis. (Seow, 2008:26-29)

2.4.1.1 Perception

The two approaches have different focuses, where the former is dependent on the establishment of actual duration measurements to serve as a basis of comparison against user-reported time. Although not being the centrepiece in this dissertation, perception still has it natural place in the discussion of performance. The role perception ultimately plays, affects what is viewed as good performance. As an example, for this, one can consider the topic of the effect mobile phones have had on what is believed to be a fast web page. Naturally, in the early days of smart phones, connections speeds were not as fast as they are today. As a consequence, for the these "low" speeds, web sites and applications needed to be adapted to suite the available bandwidth. Needless to say, the limitations of the technology during the initial years, have had implications on both the shaping and perception of web sites.

Time itself is a subject of much speculation and has been debated by many minds in just as many fields. Although not being inclined to further the discussion nor augment to it, there needs to be some lines drawn in order to grasp the subject at hand. **Figure 22**, tries to illustrate what has been initiated in the text above, namely the difference between what is actual and what is perceived. Viewing at an object, results in the light bouncing of that object reaching the cornea. The light or the image, then gets focused by the lens onto the retina which contains photoreceptors. The image gets converted by these photoreceptors into electrical signals, which in turn get relayed through the optical nerve to different parts of the brain. What happens at that point and forward is dependent on an array of uncountable variables, e.g. age, gender, culture, experiences etc.

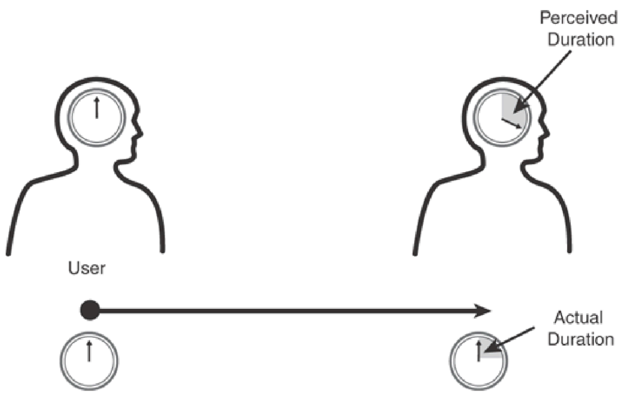


Figure 22 - Illustration of perceived and actual duration

Take for instance the cross symbol; for a mathematician, it serves as an operator that enables for arithmetic calculations. For a pirate, the cross could symbolise the position of long sought for treasure. For a student a failed task, for the writer a letter on a page, and so on. What serves as an interpreting vessel, is our brains and what has shaped the brain to the point of interpretation.

With the notion of interpretation comes the notion of retrieving the elements that server as the foundation for retrieval, namely memory. Not impeccable, memory as everything else is under constant change and subject for distortion. When experiencing an event, the parts of the brain that correlate with memory are commonly viewed as a three-stage process – encoding, storage and retrieval. At and during these three stages, memory is susceptible to distortion due to the very nature of being human. As stated earlier we are prone to bias and that shapes how these three stages are affected (Sternberg, 2012:230-246). As an example, for the susceptibility of distortion, students were shown a film on an automobile accident and later asked to estimate the speeds of the cars involved. Estimation of the speed of the cars differed, dependent on how the research question was formulated. When test groups were asked about how fast the cars were going when the *smashed* each other rather than *hit* each other, the average estimated speed rose 10.5 km/h, from 54.7 km/h to 65.2 km/h (Loftus and Palmer, 1974).

2.4.1.2 Tolerance

What has not been floating on the surface, but implicitly is ever present when discussing the topic of perception is waiting. A person in a hurry to her first day on a new job would probably have a very high stress level if she was stuck in traffic for 15 minutes. A person waiting at the airport would probably tolerate an additional 10-minute wait to the three hours he has already been waiting. As a natural consequence tolerance for any given duration is subjected to the context of the given situation and the inherent beliefs and experiences of the person experiencing that duration. The experience of any given situation is, as has been hinted, subject for the phrasing of sentences, thus prone of being elastic in the interpretation of duration. What furthermore is interesting is that in the machinery that is our brain, there seems to exist a mechanism that yields systematic asymmetry, known as time-order error. This occurs when the brain is subjected to successive stimuli that are physically equal (Hällström, 1985). The same goes for phenomena with stimulus pairs of different magnitude, stating that people tend to overestimate short durations and underestimate long durations. This is known as Vierordt's law (Kanai et al., 2006).

Since not most browsing users deliberately clock the conducted operations and processes, something else must account for feeling of the speed of a task conducted. As a matter of fact, if nothing served as a benchmark for the experience of the task, it would be rather difficult to assess something at all. With one's own experiences regarding the subject of something being fast or slow, it is rather safe to assume that everyone has some sort of reference of comparison. These experiences stored in memory and acting as the locus future actions, are the very things enable users to decide how to characterise a perceived duration. For example, if you log in to a system and the task usually takes 12 seconds, an increase of 18 seconds would be registered as an event that is out of the ordinary. The log-in task now taking 30 seconds, will now affect the user’s tolerance thus directly serving a tolerance threshold to judge future tasks with. Subsequently, tasks finished in a duration beneath the threshold will be perceived as fast.

Just as perception can be affected by different factors such as age, culture and gender, tolerance can be affected just the same. Roughly speaking one can divide the factors influencing tolerance into time-related and non-time-related factors (Seow, 2008:38-41). Time-related factors, the ones that have an attached temporal dimension, are assembled in such a way that they reside in a dimension bound by an explicit timing metric. One factor, usage frequency, where the procedure of logging in to a system can serve as an example, another is experience. Factors that are not directly set by the user’s experience explicitly are standards and benchmarks. As has been mentioned before, the introduction of smart phones has shifted consciousness of browsing towards an experience that is faster than the one perceived on a desktop browser. In consideration to all the variables that can influence the perception and tolerance, popular players i.e. web sites, play a major role on what is the “de facto” standard of what is to be perceived as fast. An example of this is a blogpost from Google stating that they’re *“[…] including a new signal in our search ranking algorithms: site speed”* (Google, 2010). Taking into account that Google is a major contributor of developing the web and its contents, what the company stands for and how they drive the evolution of the web, becomes something that companies in the web business need to identify. Weaved in the logic is consequently to adapt one’s products and services to these benchmarks and standards, in the case of non-compliance, eventually take corrective measures to be in the race. Just as with a comparison between stimuli, e.g. experiences, it is not unsafe to assume that users undertake comparison between references such as the same service provided by different companies. Examples of this could be performing a web search on Google versus a web search on Bing, alternatively a comparison of the experience of buying an air plane ticket on Norwegian’s web site versus buying a ticket on SAS’s web site. Finally, there is a need of mentioning the user interface and the indication said interface provides. Just as with the other examples mentioned above, one can tie in user tolerance and the success/failure to stay inside the boundaries of it.

The nature of non-time-related factors play out in such a way that they influence the tolerance of users, but don’t contain any timing metric per se. Take for example the number of attempts a user partakes if a web page displays a 404-page loading error. The reasons for this error could be several, the page could in fact be missing or the user could have typed in the wrong URL The willingness to stay within the limits of one’s tolerance is of course affected by several variables; the first being the reliability and stability of a product, and the second being the perceived certainty why something is not working (Seow, 2008:38-41). As some products naturally perceived as being highly reliable, it would be rather odd to see features in them that are not deemed as essential by the user. A refresh button in a web browser is expected and accepted, but would seem quite peculiar on a television set. In the case of perceived certainty about why something is not working, users could perhaps be more tolerant when configuring a newly bought smart phone. On the other hand, trouble shooting why a printer is not working for the nth time, drains both the patience and the tolerance.

Other factors influencing user tolerance are the time of day and/or season, emotive states, bias, trends and culture. One does not have to look but more at oneself to realise that one’s patience is affected by the time of day. Having to commit to a time limit, e.g. a 30-minute lunch break, would probably evoke more impenitence than two hours prior to the same lunch. Equally one would not tolerate a system doing a batch jobs during working hours since that would interfere with daily task. Equally important is the need to design systems to respond to the emotive state a user can be in when performing a task. An example of this could be that an information retrieval system utilised in a police car, need to be as simple as just typing in the license plate number during a car chase, or perhaps even better just spell out the digits on the plate. Another factor is the bias one brings to the table; mentioned above is the disgruntled user that is in need of learning a new system. Needless to say, is that biases can function in the other direction, that is the Apple Watch had the expectation of being as revolutionary as the iPhone was, the so-called *Halo effect*. Finally, one could sum up the biases of several individuals in order to speak about fads, trends and culture, depending on how many is included in the breakdown.

2.4.2 Response times

In its simplest form, response time can be defined as time difference between completion of a process or interaction and the initialisation of it, as illustrated in **Figure 23**.

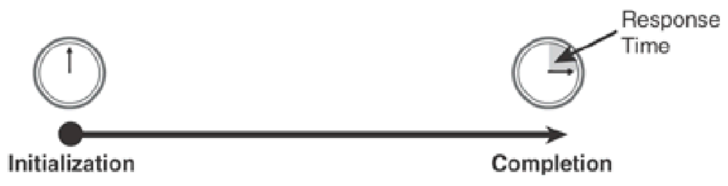


Figure 23 - Illustration of response time

Following that logic, user response time becomes the time delta of the user wanting to start a process and the moment the action to be observed is made. As there are several unobservable processes taking place, such as recognition of what is to be done and the decision to make. Determining the exact start time of a user initiated process becomes somewhat random in nature. Nonetheless this initialisation can be self-triggered, such as turning on a device; or come as a triggered response, such as choosing amongst alternatives given for a question (Seow, 2008:49-52). Needless to state, is that the latter is simpler to interact with and more interesting in the context of optimising for performance. In the other direction, the systems response time becomes “*[...] the number of seconds it takes from the moment a user initiates* *an activity (usually by pressing an ENTER or RETURN key) until the computer begins to present results"* (Shneiderman, 1984).

2.4.2.1 User response times

When presented to a simple stimulus, humans take around 200ms before giving back a simple response. Not surprisingly when humans are presented to more stimuli, the reaction time increases accordingly (Hick, 1952). This statement builds the foundation of what is known as Hick-Hyman's law. Set in the context of HCI, or more specific a web application the law implies that the more choices a user is presented with, the longer choice reaction time said has before making a choice. The law holds true when users are presented to unordered stimuli, that require very much of the user's attention in order to make a response. If, however a user is presented with an interface that enables for a non-linear search, such as an alphabetically ordered list, the relation between the length of the list and the reaction time will not hold in accordance to Hick-Hyman's law. The user is now able to scan the list for clues and reach the desired destination faster (Fitts & Seeger, 1953). Commonly the Hick-Hyman law is under the subject of inspection with regards to the nature of speed versus accuracy. Simply put, the idea is that if one is subjected to the notion of being forced to speed up a task, the result will be several more errors than if one is subjected to the notion of being as precise as possible. The speed-accuracy trade-off is illustrated in **Figure 24**. This augmentation to the Hick-Hyman law, or a response thereof, is captured in the principle known as Fitts' law. The law statuses that there is a linear relationship between the difficulty of a task and movement time (Seow, 2005).

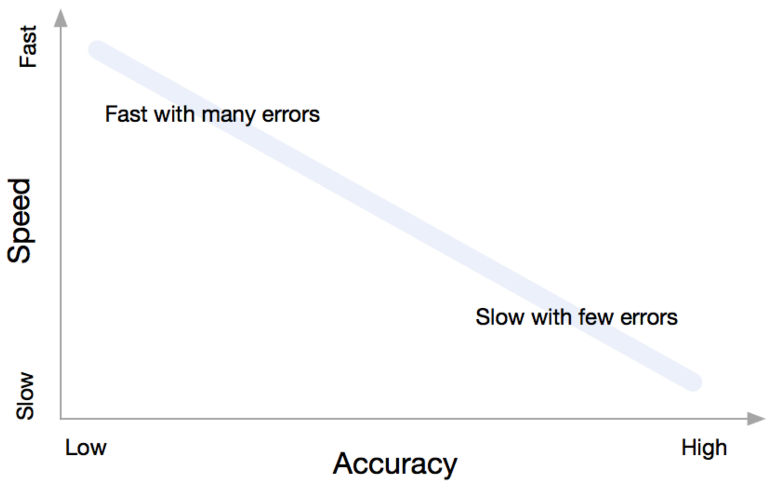


Figure 24 – Speed/Accuracy trade off

Forcing a user to conduct tasks and/or make decisions where the user interface is designed is such a way that the user has to use more brain power than needed is evidently bad practice. This directly implies that users will make faster and more precise choices when presented with an interface that is to the point, and devoid from ambiguous information. Appropriately these principles don't differ a lot from the principles that need to be invoked when designing interfaces that need to be presented on mobile devices, namely lightweight, simple design and easily graspable contents.

2.4.2.2 System response time

What has been elaborated in previous subchapters is system response, though only in the light of network performance and latency. Subjected to the same scrutiny are the delays that are existent in processes with even shorter durations. An every-day user clicking a button is, more or less, unaware of the what is going on behind the scenes. Although this is true, the same user probably has some expectation of what is going to happen; as has been shown earlier. The actual clicks, as far as the user is concerned, is the start of a process that subsequently yields a result on the screen e.g. a search in a database. Depending on how detailed one wants to be though, one has to take in considerations what is to be included in the system response. One could for instance choose to include the keyboard typing time, where the time the character pushed on a keyboard, plus the rendering time of said character on a screen, is included. There are several industry standards of system response time that has been developed over time, often by governmental organs and for the sake to impose control on system behaviour. One subset of guidelines from one of these standard is EDS/MITRE which can be seen in **Table 5**, which was developed for the U.S. Air Force, by the MITRE corporation.

Table 5 - Subset of guidelines in ESD/MITRE

Section	Topic/Action	Timing (sec)	Guidance
1.0/4	Fast response	0.2	Maximum time for delays in displayed feedback for normal operation
1.1/5	Fast acknowledgement of entry	0.2	Maximum time to acknowledge the entry of a designated position
1.1/7	Responsive cursor	0.5	Maximum time for moving the cursor from one position to another
2.7.1/6	Fast response to display request	0.5 to 1.0	System response to a simple request for data display
3.0/19	Control availability	0.2	Maximum time for control delays or lockouts
3.0/28	Appropriate computer response time	0.5 to 1.0 2.0	System response to a control entry. System response to simple entries
4.3/11	Appropriate response time for error messages	2.0 to 4.0	Display error message

The level of detail, of course, depends on the task at hand and the information necessary to extract varies with the task. Although an interesting subject do explore, for the purpose of this thesis, there is no need to attain the suggested level of detail. Finally, it is important to Souders that even with the most powerful hardware and the latest version of different software, a system will be subdued to compromise if the system is poorly designed.

2.4.3 Responsiveness

When one speaks of responsiveness of a technical solution it is to describe the characteristics of that solution, but as interaction is necessary it becomes difficult to define it without the involvement of a user. As for many areas in the computing context, the interaction between human and computer has its roots in the interaction between people. Thus, the responsiveness of a certain undergone task is as relative to the task as it is subjective to the person partaking in the interaction. In the same manner, you would expect a response within a certain time frame when greeting someone, you have an expectation of how fast the system should answer when pressing a button. Not surprisingly this would vary depending on the interaction one has with the system, as one would not expect the same responsiveness (or delay) when clicking on a web page link as one would when hitting a key on the keyboard and seeing a character rendered on the display. If the greeting does not get an answer within the expected time frame; perhaps the greeted did not hear the greeter; one naturally tries to rationalise the situation. The explanation one is providing oneself is of course a manifestation of confirmation bias, where one interprets the situation in a manner that fits ones pre-existing beliefs and experiences. In the context of computing, Occam's razor – the simplest explanation is the best one – must take its large share of the blame game. Not to seldom a newly installed system gets the blame for being bad, when the reason often resides on the lack of time users have had with the new system.

2.4.3.1 User-centric expectancy

As stated above the discussion about responsiveness relies on involving a system as well as the user utilising that system. Hardware and software enables for systems to become steadily more responsive but that is not the case for the abilities of a human being, at least not in the short run. Therefore, there is a need to define user-centric as opposed to technology centric metrics. A model that accommodates for a user-centric response time guidance, purposes to divide time constants into three segments, as shown in **Table 6.**

Table 6 - Human time constants

Time constant	Value
Perceptual processing	0.1 s
Immediate response	1.0 s
Unit task	10 s

The three different levels each represent the time limits for human perceptual abilities. An important side note is that these limits are advice and not written in stone.

- 0.1 seconds – Staying within the perceptual processing limit makes the user feel that the system is reacting instantaneously. This means that the user does not, or should not, expect to see any other feedback than the actual response itself.

- 1.0 seconds – The immediate response limit has the intention of keeping the user’s flow of thought uninterrupted, even though the user will notice the delay. The limit is purposed as a backchannel of keeping the user informed that the system is still engaged in the “conversation.

- 10 seconds – The unit task limit is the boundary under which the user’s attention can be kept focused without the user doing something else while a task is completed. Subsequently users should be informed when the system is to finish the operation it is performing.

These limits have their roots in human to human interaction and should cohere with technology-centric metrics as much as possible. Usually one would like for systems to respond as quickly as possible, but this is not always desirable. For example, scrolling speed could be set to being too fast, making it difficult for the user to stop at the anticipated moment (Card et al., 1991).

The model of the response time guide lines above, serve in the same way across all different software and web applications. As stated, there needs to be a notion of partly separating from the technology-based metric view, in order to let a user-centric metric view share the scene. The red thread, or the least common denominator would be the user interface, that serves as tool for which response times are adapted to.

3 Data collection platform

The data collection platform has the function, as implied by its name, of gathering data. What is not implied is, if the data is to reflect what is actually happening at the user's end, the collecting function needs to be as near to the user as possible. The rationale behind this requirement lies within the core purpose of collecting user data and data tied to it in the first place; one wants to know what one's users experience in order to optimise that experience. The data collection needs to be set up in such a way that it interferes as little as possible with the transactions the user is performing, i.e. add too much overhead, experienced as waiting time. In the context of Application Performance Management (APM), there are several ways of collecting data. APM is a discipline within IT that is dedicated to deal with performance optimisation and error detection. The two main methods for collecting data within APM are passive monitoring, also called Real User Monitoring (RUM), and agent based monitoring. There is a plethora of ways to use agent based monitoring, but for the main purpose of this thesis the efforts will be focused on User Experience Monitoring (UEM). More precisely the software that will be used is Dynatrace Application Performance UEM. All terms contained within this chapter and its sub chapters in one way or another is associated with the software in question. The purpose of using UEM instead of passive monitoring resides in the ability to follow user actions in the browser and how they affect the value chain further down, in the data centre. Using this approach allows for in depth tracking of user behaviour and performance monitoring based on said behaviour. On a high level the technology works as illustrated in Figure 25. A JavaScript Agent is injected into the HTML code by a web server or Java (server-side) agent, on the desired web site. The JavaScript agent then executes in, on said site, resulting in a continuous stream of data being sent back to the site where the server-side agent resides. The server-side agent, in turn, intercepts these requests made by the user, and sends them to a data collector (PurePath Collector in the figure). Further, the data is sent to the Dynatrace server where these requests are correlated to data farther in the value chain. The figure illustrates the possibilities of utilising the JavaScript agent for mobile applications, something that is outside scope for the thesis.

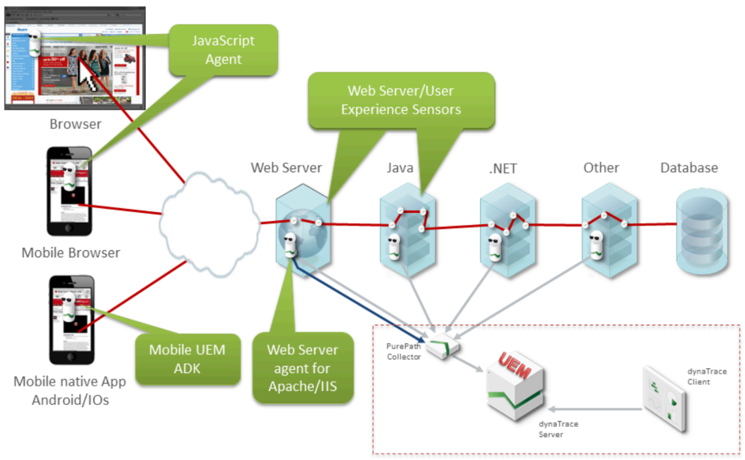


Figure 25 - UEM architecture overview

3.1.1 Application monitoring architecture

The underlying architecture enabling for the data collection can be comprised of components shown in **Figure 26.**

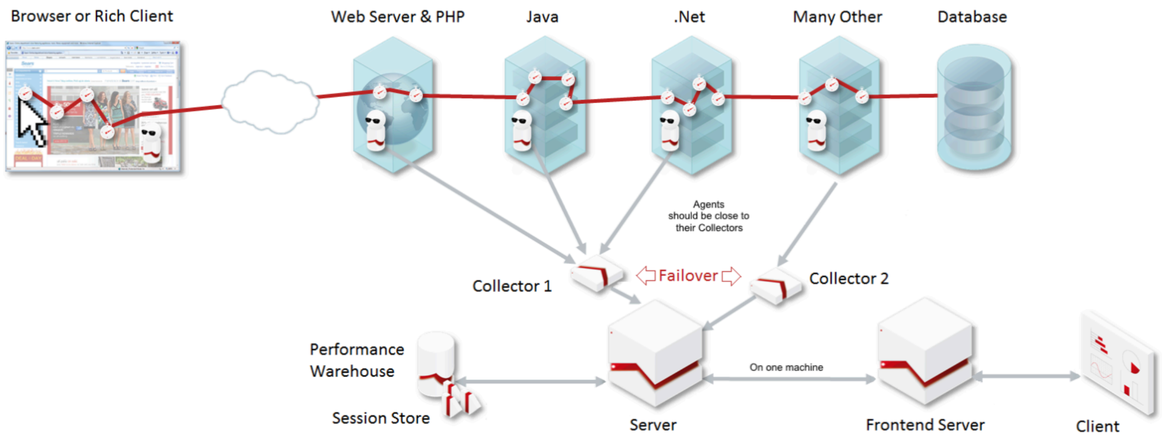


Figure 26 -Components in the AppMon architecture

Server (backend)

The server or more precisely backend server, is the central or core component of the Application Monitoring (AppMon) environment. Its main function is to correlate PurePath data that it receives from the different agents via the collector. The PurePath data is initially stored on the local file system.

Frontend server

The frontend server handles data analysis tasks as requested by the client. The frontend server and the backend server reside on the same machine but in two different processes.

Collector
The main task of the collector is, as implied by its name, to gather and bundle data that it receives from agents. In order to put as little burden on the agent machines and the backend server as possible, the collector does post-processing of the data before sending off the payload.

Performance warehouse
The performance warehouse is a database where long time series and measure data that the agent delivers. The time series and measure data is extracted from the PurePaths that are stored on the local file system (Dynatatrace, 2017a).

3.1.2 The PureModel

In order to be able to track what users are doing, in the contents of the application monitoring, one needs to have a technology with the ability to correlate or more precise "stitch" together incoming data. In the world of Dynatrace APM that technology is called PureModel and is comprised of two underlying proprietary technologies called PurePath and PureStack. As just recently mentioned, the idea is to have a means of stitching together all incoming data for each and every user session, and to give detailed information on what the user is doing in relation to server and network behaviour.

One can imagine a value chain in a horizontal line, starting and ending with a user, see **Figure 26.** User actions start with clicks (requests) that travers across the internet, through a data centre and back to the user (response). For all the touchpoints contained within that request that a user initiates, there is data to be gathered. Furthermore, information about what happens between and on these different touchpoints can be added to the calculations.

For the PurePath and PureStack technologies, data is sent asynchronously by agents to one or several collectors. Each and every agent here acts as a touchpoint and tags incoming and outgoing data, so that the server can keep track of which packets that belong to which user session; this is referred to as PurePaths. For the PureStack technology, the agents residing on servers within a monitored application environment, have the responsibility of gathering infrastructure health data for each transaction. This health data is correlated with the transaction data. Thus, one can say something about how users are affected by performance issues as well as where these issues reside in the value chain. As stated above, agents send the data a collector, which in turn buffers this data and passes it on to the server. The backend server analyses and constructs the PurePaths (PurePath and PureStack) and the frontend server makes them available for the client to access.

To make sense of technologies described in the last sub chapter, one must first of all be comfortable with what happens when a user visits a site that is monitored by an agent. Several terms will be introduced below in order to for the ground works to be comprehensible. The main purpose is to tie together what has been discussed in the sub chapters above. Furthermore, the intention is to present a foundation from which the presented data can be understood in a deeper manner than would have been the case, if one only had focused on the gathered data and the implications sprung from it (Dynatrace, 2017b).

3.1.3 Visits and user actions

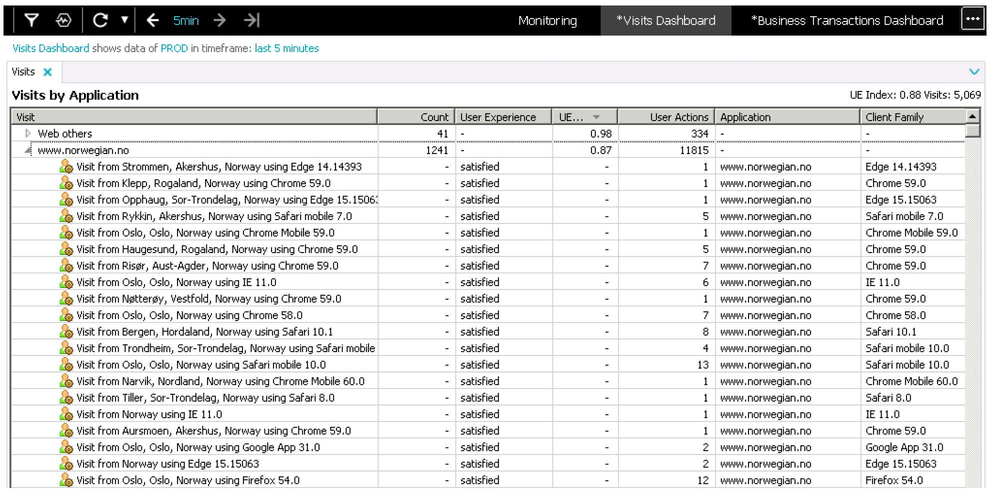
A user surfing to a web site that is monitored is said to vising this site. Our user, with the actions performed by him/her, triggers actions that the browser carries out as requests. These request in turn start a series of events further down in the value chain and hopefully comes back with an expected response. In order to track this visit, the page is tracked by a JavaScript agent. What constitutes a visit is the collected number of actions the user performs within a certain time period. A visit represents the user's click path and is the foundation for further analysis, and is initiated with the first user action performed. The nature of these actions varies depending on the request made, but has the least common denominator that the action take the user from one state to another. For example, a page load is initiated by a user request that causes the web browser to download multiple resources. The initialisation and transitioning of these requests are measured by timings. We have seen examples of these types of actions and timings in **chapter 2.3.4.**

In the above example a user could have triggered the action through an initial page load, that is entering the URL in the web browser's address bar. In doing so this initial load start time, triggered by the user action, is set to the W3C *navigationStart* time, if the technology is supported by the user's browser. If not supported, the start time is set to the time that the JavaScript agent was loaded in the browser. The latter method is less accurate but provides an alternative for older browsers that do not support the Navigation Timing API. The end time for the initial load is set when the last *onLoad* handler is has been completed. If a user clicks a link on a page the *navigationStart* is initiated with that actions and ends in the same manner as stated above. These timings are essential for the understanding of the data that will be collected and described in the **chapter 4.2.1** (method chapter).

Loading a page starts the classical request-response paradigm explained in **chapter 2.1.1**, but also results in the download and rendering of resources, that is a complete rendering a page or site. There are occasions though, when this in not true. A user click could trigger an action that only results in navigating within the actual rendered page. This is known as server side-side rendering and does not trigger any timing within the Navigation Timing API. An example of this is anchor links, which causes the user to jump from one section to another within the rendered page. The third case, and equally important as an initial page load and new page navigation, is the Web 2.0 action and/or the XHR call. These actions/calls are such in nature that they do not cause for the entire page to load. A user can, for example, click on a link that triggers an XML HTTP Request that alters the DOM-tree or the page, as explained in **chapter 2.3.3**. Each JavaScript execution with at least one XHR, constitutes a user action (as perceived by the software). Start time for these events are triggered when at the start of the user action and ends after the last *XMLHttpRequest.onreadystatechange*. This property is XHR inherent.

3.1.3.1 Visits

A visit in the context of Dynatrace Application Monitoring is a group of user actions performed by a user under a specific time frame (Dynatrace, 2017c). Visits can be vied in the *Visit Dashboard* as seen in **Figure 27**. The metrics collected by the visit gives, amongst others, information on how the overall experience is perceived by the user, as well of where they originate from which browser they are using and when the visit started.



Visits Dashboard shows data of PROD in timeframe: last 5 minutes

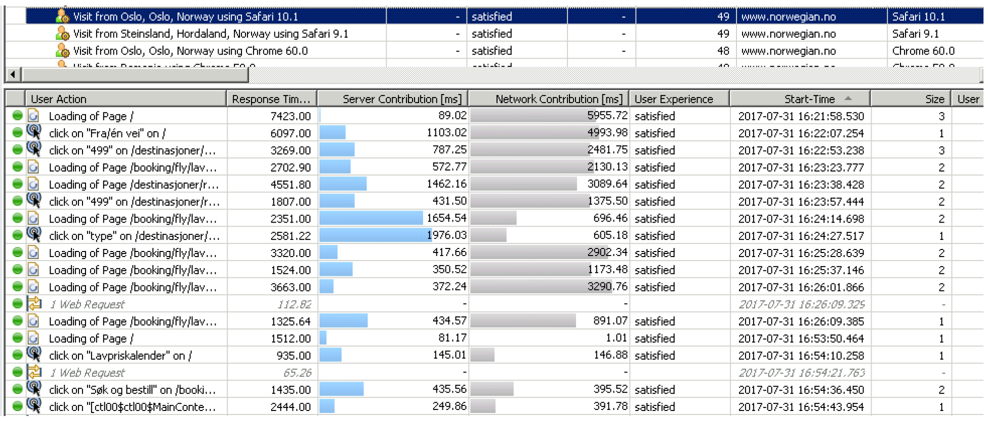
Visits by Application

UE Index: 0.88 Visits: 5,069

Visit	Count	User Experience	UE...	User Actions	Application	Client Family
Web others	41	-	0.98	334	-	-
www.norwegian.no	1241	-	0.87	11815	-	-
Visit from Strommen, Akershus, Norway using Edge 14.14393	-	satisfied	-	1	www.norwegian.no	Edge 14.14393
Visit from Klepp, Rogaland, Norway using Chrome 59.0	-	satisfied	-	1	www.norwegian.no	Chrome 59.0
Visit from Opphaug, Sor-Trondelag, Norway using Edge 15.1506	-	satisfied	-	1	www.norwegian.no	Edge 15.15063
Visit from Rykkin, Akershus, Norway using Safari mobile 7.0	-	satisfied	-	5	www.norwegian.no	Safari mobile 7.0
Visit from Oslo, Oslo, Norway using Chrome Mobile 59.0	-	satisfied	-	1	www.norwegian.no	Chrome Mobile 59.0
Visit from Haugesund, Rogaland, Norway using Chrome 59.0	-	satisfied	-	5	www.norwegian.no	Chrome 59.0
Visit from Risør, Aust-Agder, Norway using Chrome 59.0	-	satisfied	-	7	www.norwegian.no	Chrome 59.0
Visit from Oslo, Oslo, Norway using IE 11.0	-	satisfied	-	6	www.norwegian.no	IE 11.0
Visit from Nøtterøy, Vestfold, Norway using Chrome 59.0	-	satisfied	-	1	www.norwegian.no	Chrome 59.0
Visit from Oslo, Oslo, Norway using Chrome 58.0	-	satisfied	-	7	www.norwegian.no	Chrome 58.0
Visit from Bergen, Hordaland, Norway using Safari 10.1	-	satisfied	-	8	www.norwegian.no	Safari 10.1
Visit from Trondheim, Sor-Trondelag, Norway using Safari mobile	-	satisfied	-	4	www.norwegian.no	Safari mobile 10.0
Visit from Oslo, Oslo, Norway using Safari mobile 10.0	-	satisfied	-	13	www.norwegian.no	Safari mobile 10.0
Visit from Narvik, Nordland, Norway using Chrome Mobile 60.0	-	satisfied	-	1	www.norwegian.no	Chrome Mobile 60.0
Visit from Tiller, Sor-Trondelag, Norway using Safari 8.0	-	satisfied	-	1	www.norwegian.no	Safari 8.0
Visit from Norway using IE 11.0	-	satisfied	-	1	www.norwegian.no	IE 11.0
Visit from Aursmoen, Akershus, Norway using Chrome 59.0	-	satisfied	-	1	www.norwegian.no	Chrome 59.0
Visit from Oslo, Oslo, Norway using Google App 31.0	-	satisfied	-	2	www.norwegian.no	Google App 31.0
Visit from Norway using Edge 15.15063	-	satisfied	-	2	www.norwegian.no	Edge 15.15063
Visit from Oslo, Oslo, Norway using Firefox 54.0	-	satisfied	-	12	www.norwegian.no	Firefox 54.0

Figure 27 - UEM visits an example

A visit also contains all user actions performed and the click trail of the said users. The latter meaning, that one has the ability to get insight on where the user started his/her visit, which actions the user dis and when the user departed from the website. This can be seen in **Figure 28**, below.



Visit from Oslo, Oslo, Norway using Safari 10.1	-	satisfied	-	49	www.norwegian.no	Safari 10.1
Visit from Steinsland, Hordaland, Norway using Safari 9.1	-	satisfied	-	49	www.norwegian.no	Safari 9.1
Visit from Oslo, Oslo, Norway using Chrome 60.0	-	satisfied	-	48	www.norwegian.no	Chrome 60.0

User Action	Response Tim...	Server Contribution [ms]	Network Contribution [ms]	User Experience	Start-Time	Size	User
Loading of Page /	7423.00	89.02	5955.72	satisfied	2017-07-31 16:21:58.530	3	
click on "Fra/én vei" on /	6097.00	1103.02	4993.98	satisfied	2017-07-31 16:22:07.254	1	
click on "499" on /destinasjoner/...	3269.00	787.25	2481.75	satisfied	2017-07-31 16:22:53.238	3	
Loading of Page /booking/fly/lav...	2702.90	572.77	2130.13	satisfied	2017-07-31 16:23:23.777	2	
Loading of Page /destinasjoner/r...	4551.80	1462.16	3089.64	satisfied	2017-07-31 16:23:38.428	2	
click on "499" on /destinasjoner/r...	1807.00	431.50	1375.50	satisfied	2017-07-31 16:23:57.444	2	
Loading of Page /booking/fly/lav...	2351.00	1654.54	696.46	satisfied	2017-07-31 16:24:14.698	2	
click on "type" on /destinasjoner/...	2581.22	1976.03	605.18	satisfied	2017-07-31 16:24:27.517	1	
Loading of Page /booking/fly/lav...	3320.00	417.66	2902.34	satisfied	2017-07-31 16:25:28.639	2	
Loading of Page /booking/fly/lav...	1524.00	350.52	1173.48	satisfied	2017-07-31 16:25:37.146	2	
Loading of Page /booking/fly/lav...	3663.00	372.24	3290.76	satisfied	2017-07-31 16:26:01.866	2	
1 Web Request	*112.82*	-	-		*2017-07-31 16:26:09.325*	-	
Loading of Page /booking/fly/lav...	1325.64	434.57	891.07	satisfied	2017-07-31 16:26:09.385	1	
Loading of Page /	1512.00	81.17	1.01	satisfied	2017-07-31 16:53:50.464	1	
click on "Lavpriskalender" on /	935.00	145.01	146.88	satisfied	2017-07-31 16:54:10.258	1	
1 Web Request	*65.26*	-	-		*2017-07-31 16:54:21.763*	-	
click on "Søk og bestill" on /booki...	1435.00	435.56	395.52	satisfied	2017-07-31 16:54:36.450	2	
click on "[ctl00$ctl00$MainConte...	2444.00	249.86	391.78	satisfied	2017-07-31 16:54:43.954	1	

Figure 28 - User actions for specific UEM visit

3.1.3.2 Appdex and User Experience Index

Appdex is an abbreviation for Application Performance Index and is a standard way of depicting the performance of user actions. The method gives an easy way of assessing how good an application or web page is performing. The modus operandi of the method is to categorize all user actions by their response time (Appdex, 2007:8). The equation to calculate the Appdex number ranging from 0-1, is presented below in **Equation 5**. The subscript, T, is the desired threshold value.

Equation 5 - Appdex calculation equation

$$Appdex_T = \frac{Satisfied\ count \bullet \frac{Tolerating\ count}{2}}{Total\ samples}$$

In Dynatrace Application Monitoring the default threshold for a satisfied user action is two (2) seconds, and a frustrated user action is four times the satisfied value i.e. 4x2 second as depicted in **Figure 29**. Dynatrace AppMon, however adds another dimension into the mix, namely that of the user experience, meaning that user actions are also weighted with regards to the outcome of user actions and errors.

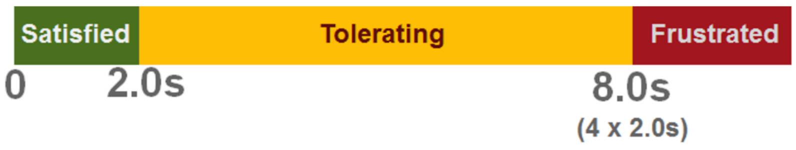


Figure 29 - Appdex grading scale

Thus one can categorise all user experience according to time and outcome of actions. A list of when an experience is categorised as satisfying, tolerating or frustrated is as follows (Dynatrace, 2017d).

Users have a frustrating experience if

- Their last action failed ("The Web site does not work - I'm leaving!")
- Their last action was frustrated ("The Web site is too slow - I'm leaving!")
- More than 50% of all actions were frustrating

Users have a satisfying experience if

- No action failed
- More than 50% of all actions were satisfying

Users have a tolerating experience if

- Their last action was not frustrating or failed
- Less than 50% of all actions were satisfying
- More than 50% of all actions were at least tolerating

3.1.3.3 User action timings

In the contents of User Experience Management (UEM), several timings are available to be used as a foundation for charting, calculations and more. An example of the loading of a page, usually index.html, is shown in **Figure 30**, below. Earlier in this chapter, PurePaths were discussed, here the concept of User Action PureParths is introduced. The only difference between PurePaths and User Action PurePaths is the origin of the actual data stream. For User Action PurePaths the data stream starts with the PurePaths that initialise in a user's browser.

Load actions

To be able to discuss some of the timings that will be described below one must have in the back of the mind the concepts of bandwidth and latency, as described in **chapter 2.2.1** and **chapter 2.2.7**. Bandwidth is calculated by the JavaScript agent as download size/download time and the latency is calculated by the change in time between the user browser and web or application server (here JavaScript agent round-trip time).

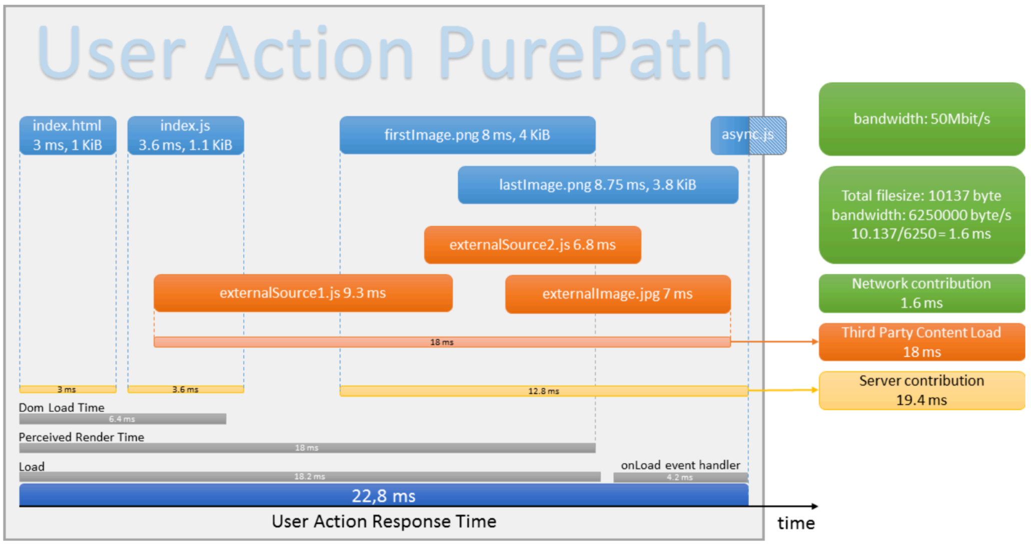


Figure 30 – Load action timings example

In the figure above, server side resources, that is resources that reside on the server where the request was aimed at, are coloured in light blue. External resources, that is resources that reside somewhere else on the server side are coloured orange. Resources that are loaded on one line are called synchronous or sequential resources e.g. here *index.html* and *index.js*. The other resources here are called non-sequential or asynchronous resources e.g. here *firstImage.png* and *lastImage.png.* The timings, coloured grey, are measurements as calculated by AppMon but have their origins in W3C timings see **chapter 2.3.4**.

The load action as calculated by AppMon starts when the page initialises (page load or user click) and ends at the completion of the *onLoad event handler* here called *User Action Response Time.* In between end and start, one can see the timings *DOM Load Time* and *Perceived Render Time*. The *DOM Load Time* event represents the time delta between an initialisation of the page and the *domContentLoaded* event. The latter event represents the time it takes for the initial visual part of the page to load. This is a truth with some modification as parts of the non-displayed contents can affect the total *Perceived Render Time.*

One can furthermore, see that the *User Action Response Time* does not mean that the loading of the actual page has ended. This is, in the graph, represented as the *asnyc.js* being wider than the actual *User Action Response Time.* The reason for this is that the script is triggered asynchronously and does not affect the initial user action. However, it contributes partially to the server contribution time.

Source actions

Source actions are triggered as a result of a user wanting to redirect to another page and/or load a resource within that page without leaving the current page. This can be done by triggering an XHR callback function, as described earlier. The JavaScript agent tracks these user clicks and correlates the corresponding request to the performed user actions. **Figure 31**, is a representation of such a source action and the timing occurs in the same manner as the previous example (Dynatrace, 2017e).

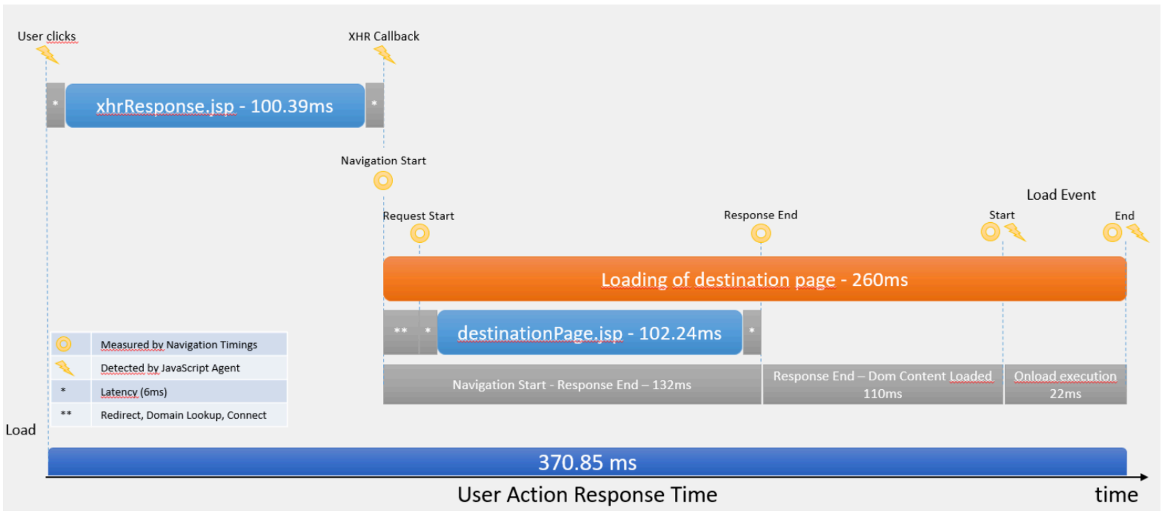
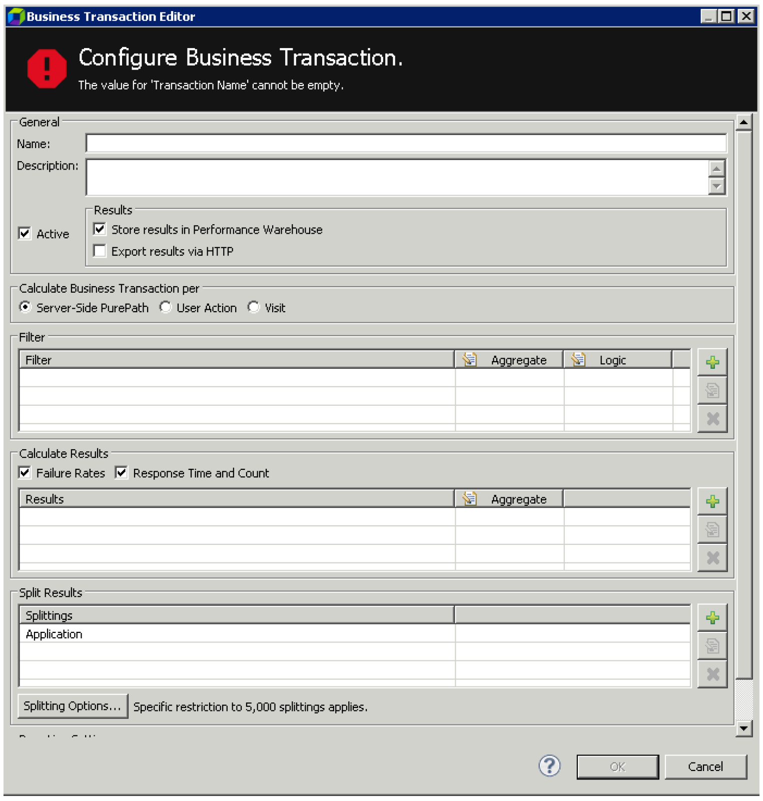

User clicks
XHR Callback
xhrResponse.jsp - 100.39ms
Navigation Start
Load Event
Request Start
Response End
Start
End
Loading of destination page - 260ms
destinationPage.jsp - 102.24ms
Measured by Navigation Timings
Detected by JavaScript Agent
Latency (6ms)
Redirect, Domain Lookup, Connect
Navigation Start - Response End – 132ms
Response End – Dom Content Loaded 110ms
Onload execution 22ms
Load
370.85 ms
User Action Response Time
time

Figure 31 - Source action timings example

3.1.4 Business transactions

According to Dyantrace a business transaction (BT) is the following *"A is a set of automated interactions between IT systems that execute a business task or business process. In AppMon, a business transaction is a categorization of PurePaths defined by filter or grouping parameters. When a PurePath matches those parameters, one or more evaluation metrics are calculated to create the result set of the business transaction."* (Dynatrace, 2017c). What this means is that one can keep track of specific transactions that are essential for a business. If one for example want to track how many users that have logged in or how many times users have bought a particular item, a business transaction would be the right choice.

On creating a business transaction, one can include a set of configuration steps in order to pinpoint the desired outcome. The steps are as following: Filter, Results and Splitting, as seen in **Figure 32.** Besides the name and description of the business transaction at hand, one is required to choose which view one wants. The choices are Server-Side PurePaths, User Actions or Visits. This initial selection tells the business transaction which incoming raw data to evaluate, e.g. does one want to include information from all nodes and the information between the nodes (Server-Side PurePath), does one only want to look on the Visits or perhaps only on the User Actions produced under said Visits?


Business Transaction Editor

Configure Business Transaction.

The value for 'Transaction Name' cannot be empty.

General

Name:

Description:

[✓] Active

Results

[✓] Store results in Performance Warehouse

[ ] Export results via HTTP

Calculate Business Transaction per

[✓ Server-Side PurePath] User Action Visit

Filter

Filter	Aggregate	Logic

Calculate Results

[✓] Failure Rates [✓] Response Time and Count

Results	Aggregate

Split Results

Splittings	
Application	

Splitting Options... Specific restriction to 5,000 splittings applies.

OK Cancel

Figure 32 - Business transaction editor

Filter

Filters act as a restricting mechanism, where one can look for specific outcomes of the raw data. If one for example would like to count the number of occurrences of one specific web request one would create a measure for that web request and include it in the *Filter* section in the business transaction. Furthermore, multiple filters can be aggregated creating a logical concatenation, in the same manner as logic is utilised for integrated circuits. For example, if the first FALSE condition is followed by an AND concatenation, then the complete expression evaluates to FALSE.

The measure acting as a filter for the business transaction, threshold values needs to be set to be able to trigger on the desired outcome. For example, one wants to track all visits that have a longer visit time than 30 minutes.

Result

Adding measures to the Result pane, will produce values only if there is a hit in the Filter pane.

Split

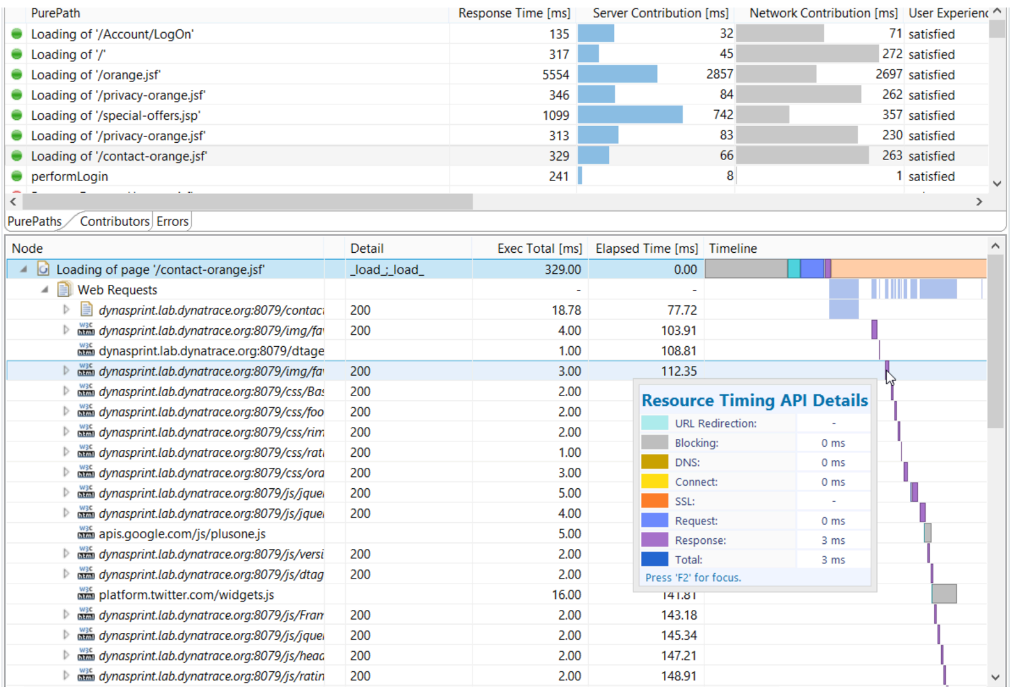
The measures within the Split pane, segments the data by the splitting criterion. One can for example split by country of origin or URI.

It helps to think about these four choices described in this section as an SQL query, as described below. The SQL query being on the left side and the business transaction equivalent on the right-hand side.

SELECT *	Result
FROM table	Server-Side PurePath / Visit / User Action
WHERE a=b	Filter
GROUP BY c	Split

3.1.5 W3C timings

Sub-chapter 2.3.4, introduced the concept of performance timings based on recommendation made by the World Wide Web Consortium (W3C). The same attributes introduced in that sub-chapter are utilised in the calculations of measurements included in Dynatrace AppMon. **Figure 33**, gives an example of how a specific *User Action PurePath* is broken down into several requests. These request, in term, can be broken down further, here we see how different measures (DNS, Connect, SSL etc.) are utilised; these measures having their base in the navigation timings of the W3C (Dynatrace, 2017f).



PurePath	Response Time [ms]	Server Contribution [ms]	Network Contribution [ms]	User Experienc
Loading of '/Account/LogOn'	135	32	71	satisfied
Loading of '/'	317	45	272	satisfied
Loading of '/orange.jsf'	5554	2857	2697	satisfied
Loading of '/privacy-orange.jsf'	346	84	262	satisfied
Loading of '/special-offers.jsp'	1099	742	357	satisfied
Loading of '/privacy-orange.jsf'	313	83	230	satisfied
Loading of '/contact-orange.jsf'	329	66	263	satisfied
performLogin	241	8	1	satisfied

PurePaths Contributors Errors

Node	Detail	Exec Total [ms]	Elapsed Time [ms]	Timeline
Loading of page '/contact-orange.jsf'	_load_;_load_	329.00	0.00	
Web Requests		-	-	
dynasprint.lab.dynatrace.org:8079/contac	200	18.78	77.72	
dynasprint.lab.dynatrace.org:8079/img/fa	200	4.00	103.91	
dynasprint.lab.dynatrace.org:8079/dtage		1.00	108.81	
dynasprint.lab.dynatrace.org:8079/img/fa	200	3.00	112.35	
dynasprint.lab.dynatrace.org:8079/css/Ba	200	2.00		
dynasprint.lab.dynatrace.org:8079/css/foo	200	2.00		
dynasprint.lab.dynatrace.org:8079/css/rin	200	2.00		
dynasprint.lab.dynatrace.org:8079/css/rat	200	1.00		
dynasprint.lab.dynatrace.org:8079/css/ora	200	3.00		
dynasprint.lab.dynatrace.org:8079/js/jque	200	5.00		
dynasprint.lab.dynatrace.org:8079/js/jque	200	4.00		
apis.google.com/js/plusone.js		5.00		
dynasprint.lab.dynatrace.org:8079/js/versi	200	2.00		
dynasprint.lab.dynatrace.org:8079/js/dtag	200	2.00		
platform.twitter.com/widgets.js		16.00	141.81	
dynasprint.lab.dynatrace.org:8079/js/Fran	200	2.00	143.18	
dynasprint.lab.dynatrace.org:8079/js/jque	200	2.00	145.34	
dynasprint.lab.dynatrace.org:8079/js/head	200	2.00	147.21	
dynasprint.lab.dynatrace.org:8079/js/ratin	200	2.00	148.91	

Resource Timing API Details

▨	URL Redirection:	-
▨	Blocking:	0 ms
▨	DNS:	0 ms
▨	Connect:	0 ms
▨	SSL:	-
▨	Request:	0 ms
▨	Response:	3 ms
▨	Total:	3 ms

Press 'F2' for focus.

Figure 33 - W3C timing utilization in AppMon

A table of the measures utilised by AppMon follows below in **Table 7**. The rightmost column shows the usage of the *Navigation Timing*, form the W3C. These metrics or measures as they are labelled here, are the same ones as presented in **Figure 20**. This is vital, as it shows that the same metrics are utilised across different software and platforms.

Table 7 - Measures utilised by AppMon

Measure	Description	Definition in terms of W3C specification (* = window.performance)
DNS	Time spent or resolving domain names.	*.domainLookupEnd - *.domainLookupStart
Connect	Time spent establishing a socket connection from the browser to the web server.	*.connectEnd - *.connectStart
SSL	Time spent establishing a secure socket connection from the browser to the web server.	*.connectEnd - *.secureConnectionStart
URL Redirection	Time spent following HTTP redirects.	*.redirectEnd - *.redirectStart
Request	Time spent waiting for the first byte of the document response.	*.responseStart - *.requestStart
Response	Time spent downloading the document response.	*.responseEnd - *.responseStart
Total	Time between the response being delivered and the onLoad event.	*.loadEventEnd - *.domLoading

4 Methodology

The research objective of the master thesis is to investigate the correlation between site speed and conversion rates, for a given user journey. Right from the start this *simple* research question offers a plethora of choices. How should the user journey be defined? Which aspects of the user journey affects the conversion the most. Are the data samples representative enough to make any conclusion? Furthermore, the theory explored in the previous chapters implores for extended research to be performed, as said theory points in the direction of a weighted user experience. This attribute of weighting has been discussed in **chapter 2.3**, and implies that a larger percentage of good user experience lies within the enhancement of the frontend, rather than the backend domain. To be accurate, Steve Souders points in the direction of this weighting of frontend and backend to have a relationship of 80/20, in favour of frontend (Souders 2007:3-5). Since theory points in the direction of a pretty drastic skewness between the two discussed domains, it would be unmistakably erroneous not to investigate this subject further. One can, and should, question why one should take any consideration into this matter at all? The answer lies in the realm of the multitude of available variables, more exactly the multitude of variables that can affect the correlation between site speed and conversion rates. Involving too many variables introduces uncertainties in such a way that the predicted model would be able to explain less then otherwise possible. Narrowing down the number of variables, abides in the research of the main objective; thus, an investigation of the relationship between frontend and backend time is in order.

The choice of research methodology is a quantitative approach since the focus is to find a correlational relationship between dependent and independent variables. It must be said that one can approach the end user experience through the user's perception, as has been elaborated in **chapter 2.4**. This weighting of the perception of a user, and the phenomena of making a choice, most certainly has its effects on the overall conversion rate. Due to the nature of the qualitative approach, and the contact that it requires with (here) the geographically disperse end users the method at hand will not be elaborated within the scope of the thesis. Although qualitative methods would add an extra dimension this could be tainted with bias and inconsistency. Finally, the data would not have been solid enough to act as a foundation needed for the statistical analysis needed, therefore, a qualitative approach has been disregarded.

This chapter describes the methods and procedures used, including specific setup for the software utilised. Furthermore, the chapter discusses the choices made with regards to data sampling and analysis.

4.1 Research design and context

As has been stated earlier, the research was conducted using quantitative methods and split up into two parts. The first part of the thesis undertakes an investigation with the purpose of pinning down where most of the user time is spent when loading a web page/site. The second part goes forth and studies the correlational relationship between web site speed and the conversion rate on that site, namely Norwegian's web site. Since the research has two different aims, however intertwined they are, it has been a requirement to utilise two different data sources. The first part of the thesis has required a broad approach, where many web sites say something about the characteristics of where user time is spent. The second part commemorates data gathering at its core and was done through a platform setup with the aim of capturing the raw user data.

To establish this first relationship between frontend and backend variables, the HTTP Archive is a more than a sound source of data. This database has been fed with web site data since 2010 and utilises the Alexa top 1 000 000 site list as its foundation (HTTP Archive, 2017e). Furthermore, the data collected by the HTTP Archive is made available on Google BigQuery (Google BigQurey, 2017). This makes it possible to perform queries with the intended aim. The many metrics gathered by the HTTP Archive enables for just as many choices, meaning that one needs to be precise with regards to which queries one resorts to. Based on the fact that there are several metrics compartmentalised within the dimension response time, as has been discussed in **subchapters 2.3.4** and **3.1.5**, there are several aspects of the analysis at hand, and thus also many branching possibilities. To limit this branching of possibilities, the metrics chosen in the investigation of frontend and backend distribution, are such in nature that they adhere to the perspective of the software at hand. What is meant by that is that the software utilised has limitations with regards to which metrics one is able to collect. The most preferable metric to utilise from the dataset would be *domInteractive*, which is when the browser has finished parsing all the HTML and the DOM construction is complete. This metric is a measure of when the user can start utilising the page and not when the page is completely loaded, thus a better choice than the alternative discussed soon. The limitation of the software has however forced the research in a slightly different direction, namely in the direction of choosing metrics that occur later on the time scale than *domInteractive.* The metrics chosen to look more closely upon are: *TTFB* (time to fist byte) and *onLoad*. These variables are explained in further detail in the next sub chapter.

The second part of the analysis focuses on raw user data analysis collected from Norwegian's web site. The collected contains information from the entire web site www.norwegian.no. Once again there are a plethora of metrics to choose from the collected data sets. Earlier in **chapter 3**, different data collection approaches have been unfolded. The approach behind this data collection is agent driven, meaning that an agent has been gathering the data (as compared by agentless data collection). Several different time dimensions have been discussed throughout the thesis. Due to the restrictions discussed earlier in this chapter, the dimension chosen to work with is this part of the analysis is what is referred to as *Loading Page Response Time* (hereafter *Response Time)*. This is the same as *onLoad* in the first part of the analysis. To collect data from specific steps of user visits, a user journey has been defined and configured, as described in **chapter 4.3.1**. The variable selection in this second part of thesis have been chosen with regards to the assumed correlation between said variables and the outcome, namely conversion. These factors are believed to have positive impact on the conversion

4.1.1 Logistic regression

Logistic regression, logit, is a case of the generalised linear model (GLM), where the outcome is dichotomous, that is either being a failure or a success, a 0 or a 1. For a logistic regression analysis, the model parameter estimates $(\alpha, \beta_1, \beta_2, \ldots, \alpha, \beta_p)$ should be obtained and it should be determined how well the model fits the data. Logistic regression does not assume any linear relationship between predictor and response variables. Furthermore, the response variables do not need to be normally distributed nor is there any assumption for homogeneity of variance. This means that the variance does not need to have to be the same within categories. Lastly there is no assumption for normally distributed error term (Field, A. 2009:273). One advantage of the logit model is that the explanatory variables types can be continuous, dichotomous, discrete or mixed. The response variable is either 0 or 1 and thus the probability for the outcomes is 1-π or π, respectively. Since one measures the outcome in such a way, the function of the response variable is not linear. Logistic regression uses a logarithmic transformation to the odds $\frac{\pi}{1-\pi}$ to transform the range of the response to a real number, as described in **Equation 6** below.

Equation 6 - Logistic regression function

$$Log_e\left[\frac{P(Y=1|X_1,\ldots,X_p}{1-P(Y=1|X_1,\ldots,X_p}\right] = Log_p = \left[\frac{\pi}{1-\pi}\right] =$$

$$= \alpha + \beta_1 X_1 + \cdots + \beta_p X_p = \alpha + \sum_{j-1}^{p} \beta_j X_j$$

where

$$\alpha = The\ constant\ of\ the\ equation$$
$$\beta = The\ coefficient\ of\ the\ predictor\ variables$$

To fit a logistic model, one must determine which predictor variables that are significant enough to include in the actual model. The goal is inherently to predict outcomes by finding a model that is the most parsimonious. Thus, one needs to utilise the predictor variables that are most fit when it comes to the prediction of the outcome. There are several ways of maintaining the model with the best fit. The one utilised in this research is a stepwise approach.

This approach iteratively tries to remove predictor variables form the model in an attempt to find the variables that do not add significance to the model. The assessment of which model is most fit with regards to the included predictor variables is determined by the Akaike Information Criterion (AIC). AIC is a measure of strength of a given model at describing the data, relative to other compared models.

$$AIC = 2p - \ln(L)$$

Where p is the number of coefficients being calculated in the model, and L is the maximised value of the likelihood function of the model. The inherent idea with AIC is to penalise a model for having too many predictor variables. The best fit with regards to the model, is the one with the lowest AIC. Another aspect of including many predictor variables, it that of multicollinearity. One does not want to have in the model, predictors that can be approximated as linear combinations of other predictors in the dataset (Agresti, 2002:216) The approach of utilising the Variance Inflation Factor (VIF), gives the researcher the ability to quantify the multicollinearity for any given variable. The VIF is calculated as

$$VIF_i = \frac{1}{1 - \mathcal{R}_i^2}$$

where VIF is the variable inflation factor for variable i. For predictor variables, the VIF is a function of the $\mathcal{R}^2$ value for a regression model prediction that variable against the other predictor variables. The $\mathcal{R}^2$ value is the percent of change in the dependent variable explained by the change in the predictor variables. This indicates that the higher predictive ability of other variables on one specific variable, the more variance of the coefficient of that variable. VIF values considerably larger than one (1), are indications of multicollinearity (Kutner et al., 2005: 406:409). The aim with the use of VIF is to decrease the existence of variance, thus making the model more robust.

To say something about how well a model performs one can utilise a Receiver Operating Characteristic (ROC) curve. The curve displays a plot between sensitivity as a function by (1-specificity). The area under the usually concave curve (AUC) connecting the point (0,0) and (1,1), represents how good the model is at predicting the outcome of the explanatory variable; the larger the area the better the predictions (Agresti, 2002:228-230).

Finally, there is a need of saying something about the odds ratio. The odds ratio is the probability of an event occurring divided by the event not occurring. That is the odds Ω are

$$\Omega = \frac{\pi}{1 - \pi}$$

The ratio for two odds Ω_1 and Ω_2 is as follows

$$\theta = \frac{\Omega_1}{\Omega_2} = \frac{\pi_1/(1-\pi_1)}{\pi_2/(1-\pi_2)}$$

π_1 and π_2, refers to the probability of success in the respective groups ($\Omega_1$and Ω_2). When the odds ratio, θ > 1 it indicates that the success of $\Omega_1 > \Omega_2$. Take for example an odds ratio, θ = 5, the odds for Ω_1 is four times greater than the odds of Ω_2 (Agresti, 2002:44-45). Taking the log of the coefficients from the resulting logistic regression gives the odds ratio.

4.2 Part I - Frontend vs backend

4.2.1 Data collection

In order to analyse the where users spend most of their time loading a web site, HTTP Archive and Google BigQuery has been utilised. The former is a site that collects metrics from about 500 000 web sites across the globe. The latter gives users the possibility to make queries on the data amassed from former. The large sample provided by HTTP Archive, must be considered relevant enough and serves the purpose of being the data foundation which will be utilised for the analysis in this chapter. The relevancy is tied to the fact that data has been collected since 2010 and that the database houses one of the largest data sets of its kind (HTTP Archive, 2017d).

A data set from March 2017, from the HTTP Archive, has been utilised to perform an initial analysis. The data is made available by Google and is reachable on Google BigQuery (Google BigQuery, 2017). The data set is called *2017_03_15_pages* and the query can be seen in **Appendix Query I**. The reasoning behind the choice of only one sample, relies on the fact that the web page size and the number of requests has increased in size over the years. This discussed in **chapter 2.3.**

The increase in average page size and the increasing number of page requests, goes hand in hand with the increase of bandwidth. This does not directly imply that the relationship between time spent on frontend and backend will be the same as Steve Souders refers to (Souders 2007:3-5). However, the increasing size web pages/sites and the increasing number of request must nonetheless yield in an increase in response time, after all more content equals more to download. Since an increase of performance in hardware and software is not restricted to specific domains, one most accept the fact that the underlying enhancements affects across the hardware and software spectra. Thus, implying that an increase on the backend side also is reflected on the frontend side.

Querying the data set in Google BigQuery enables for the choice of many variables as can be found under **appendix 10.3**. For the purpose of showing where the most of a user's time is spent when surfing to a web site, the choice of the following variables is made: URL, Time To First Byte (*TTFB*) and *onLoad*. The variables are already defined in previous sections, but for the sake of overview they are repeated.

- *URL* (here) acts only as an ID field for the rest of the variables, and represents the name of the URL measured.
- *TTFB*, represents backend time or the time it takes the server to get the first byte back to the client.
- *onLoad,* measures when the entire page is loaded including images, JavaScript and other included resources.

In **Table 8**, time references for three different software/frameworks is shown. The point being that there are different names for the same measures when one moves across the different types of software/frameworks.

Table 8 - Naming conventions for different references and frameworks

Timing reference	First byte is received by client browser	HTML content is loaded but not entire page (JS and images etc)	All resources are loaded (including JS, images and other resources)	All resources are loaded and no network activity has occurred for 2 seconds
Web page test	TTFB	Dom Content Loaded	Load Time (onLoad)	Fully Loaded
Navigation Timing	responseStart	domContentLoaded	domComplete	loadEventEnd
Dynatrace AppMon timing	responseStart	domContentLoaded	domComplete, also called Response Time.	loadEventEnd

The data collected by the HTTP Archive is collected by agents (synthetic) that are located in a data centre in Redwood City, CA. Each URL is loaded three times with an empty cache and the result form the median run, based on load time is added to the data base (HTTP Archive, 2017d). There are several dimensions why this is important to have in the back of one's head when moving forward with the data material. The first item to bear in mind is the location of the synthetic agents, namely Redwood City, CA. The location per se, affects the outcome of the data due to the fact that of the geographic position, as discussed in **chapter 2.2.7**. What this means is that web sites located geographically farther away from the Redwood data centre, will perform worse as a result of the limitations of the speed of light.

The second concern to bear in mind is the notion of loading a page with an empty cache. This means that results from these actions are such as if a user had visited a web site for the first time. More precisely this means that one does not take in consideration web page resources that get cached after a first visit; this is discussed in **chapter 2.3.2**. The implications of this don't have any effect on this first part of the analysis, since the objective is to show the frontend/backend time spent distribution. However, as a side note, there is a difference in the load time for recurring visitors. A user visiting a web site for the second time will experience faster load times, under the circumstances that the web site is set up in such a way that it enables for caching.

Finally, the last consideration that needs to be addressed is the ever so ominous case of bias in the Alexa Top 1 000 000 represented web sites. Since there is a possibility for data being skewed in the favour of one geographical region; this, third consideration ties in to the first one. Given the geographical position of the data centre and the possibility for an over representation of, say sites in India, the data set would have possibility for being skewed in favour of longer response times. Since we only can check how frequent a top-level domain (e.g. .com, .no, .in) is represented in the HTTP Archive data set, and not where the physical placement of the web server is, it makes little to no difference for the outcome of the research.

4.2.2 Data analysis

The results from the HTTP Archive, via Google BigQuery were exported to a .CSV file and then converted into a .XLSX file. See **appendix section 10.6** for details. The exported file and its contents were analysed to determine if any of the data behaved irregularities compared to the what is normal to the rest of the data set. These checks for irregularities include outliers and overrepresentation of certain top-level domains, as discussed in the previous sub chapter. The data was initially viewed and edited in Microsoft Excel and after that imported to the statistical software R. As part of the analysis, descriptive statistics were generated. These statistics included amongst others mean, median, mode and range for the different variables. Furthermore, density distribution and boxplot graphs were generated. After an initial view of the data set, there were no direct surprises in the distribution of the top-level domains. There is an overrepresentation of .com domain in the set, but that probably has to do more with historical reasons than with other factors. Nonetheless, this has little to no obvious effect on the data set as we don’t know where these domains reside physically.

Analysis of the distribution of the frontend and backend time yielded that there is a rather large skewness and kurtosis in the set, indicating long tails. This can have a large effect on the statistics but there are ways of compensating for the undesired effects. One simple way is to trim the data, thus removing a certain amount of data from the tails. Another way is to replace the extreme values with other less extreme ones, also known as Winsorizing. A third approach would be to set a limit and delete the data exceeding said limit. Due to the fact that both variables in the data set were right skewed and that in the initial state the data was in, the decision to see what the data would look like if one forced the data in favour of the backend data. This means that the extreme points, or outliers, of the *onLoad* variable are deleted from the data set. Doing so will lead to a smaller set of data but it would also give more emphasis for the backend segment of the data set. Finally, the procedure from before the altering of the data set, was repeated.

4.3 Part II – Performance and conversion

4.3.1 Defining the user journey

A user journey in the context of a website, has something to do with the path a user takes in order to reach a conversion. Inherent in the terminology is that there are several paths to take to come to the end of a journey – this is no exception. The goal for the user journey is to convert, or to purchase a (here) ticket. As commercial websites often give its visitors many choices on how to pay, if they want to customise certain outcomes and/or chose additional features for certain steps, the complexity grows. Norwegian's website is no exception to this fact. The user journey which has been utilised here is defined by Norwegian and is one of the journeys they utilise when analysing visits to the website. Furthermore, the journey looks different for all the different country sites e.g. the Norwegian site differs from the UK site, but has the same underlying structure. This fact has been utilised and contributed to the design of a slightly different journey for the thesis.

This original journey is split into seven steps as represented in the **Table 9,** below.

Table 9 - Original user journey for Norwegian's website

Step	Name	Info
1	Select flight	First page where user select flight
2	Login or create profile	User logs or creates a profile (not necessary)
3	Passenger	Passenger fills in personal details
4	Seat reservation	Choice of seats
5	Additional products	Additional products e.g. extra or special luggage
6	Confirm reservation	Payment methods and final confirmation
7	Booking	Booking is complete

Each of these steps in the user journey has a particular URL associated with it. The URLs for step 2 to step 7, are all underpinned the same coherent structure for each and every country site. For example, when you reach *Step 3 – Passenger,* a part of the URL would be 'resmake/resmakepax/', this is true for all country sites. This is however not true for the first step, *Select flight*. For this particular step, the URL is different for each and every country. An example of this is that Norwegian site has the address https://www.norwegian.no/booking/fly/velg-flyvning/. For the UK site the equivalent URL would be https://www.norwegian.com/uk/booking/flight-tickets/select-flight/. This makes it more difficult to make an analysis across the spectrum of different countries. Therefore, a choice of not including the first step as such as been taken. This does however not mean that the landing page for visitors is ignored. The choice to treat all initial page hits equal has instead been taken. The enables for a wider initial funnel and catches all visitors on Norwegian's web site, instead of those who solely land on the beforehand defined initial step. Furthermore, the last step has been eliminated due to the fact that this final outcome of a booking is managed by a binary metric called *Converted*. The resulting user journey looks as is depicted in **Table 10**.

Table 10 - Modified user journey for Norwegian's website

Step	Name	Info
1	Index	Whichever page the user lands on.
2	Login or create profile	User logs or creates a profile (not necessary)
3	Passenger	Passenger fills in personal details
4	Seat reservation	Choice of seats
5	Additional products	Additional products e.g. extra or special luggage etc
6	Confirm reservation	Payment methods and final confirmation

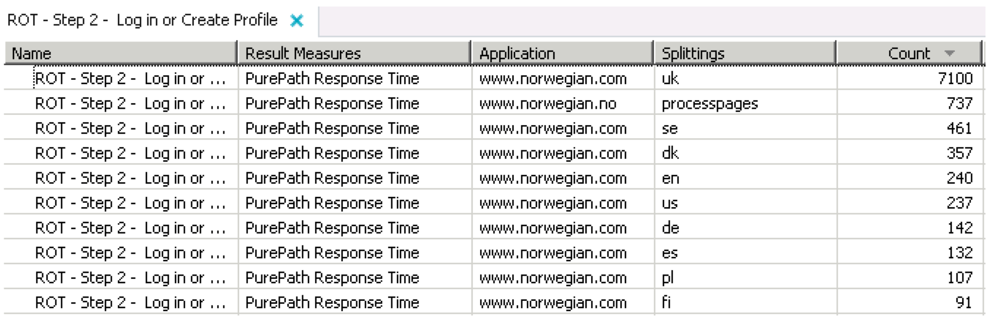
Although the data collection platform gathers all the raw data, one needs to instrument said platform in order to gather specifics. More precisely one needs tell the platform what one is interested in out of all the raw data that is available. This is done by the help of creating logical segmentation of the data. This is described in more detail in the **Business transactions** chapter, and the specifics are presented under the **appendix, section 10.4.** The business transactions created to represent the user journey is set up with the consideration of the underlying URL for that particular step. The exception for this is the first and initial step which has a lot more URLs included. In fact, the first step does not take in consideration which page the user lands on, as long as it is not one of the other pages in the user journey. If the initial page of the user journey is on the login step, then that visit will be accounted for in step two in the journey and not in step one.

The setup of the business transaction step is presented in **Table 11,** below.

Table 11 - Business transaction setup for user journey

Step	Name	Filter	Split result
1	-	-	-
2	Log in or Create Profile	URI ends with: resmake/resmakelogin/	First part of URI path: /([^(/;\?)]*).*
3	Passenger	URI ends with: resmake/resmakepax/	
4	Seat Reservation	URI ends with: resmake/resmakeseating/	
5	Additional products	URI ends with: resmake/resmakenonair/	
6	Confirm reservation	URI ends with: resmake/resmakeconfirm/	

The different business transactions in the above presented table, tracks all web request hits that have occurred for the given URI. If we take the second step for example, all users that have passed through the URI that ends with the filter (column three) resmake/resmakelogin/, will be marked as a hit for that specific business transaction. If users move further down the funnel, that information is recorded, and so on and so forth. The final column represents how one splits that outcome of the data. Since all the URIs of the user journey follow a specific pattern, one can split the URIs in order to gain (here) country specific information. If we continue with the example of the second step of the user journey, a visitor utilising the German site would have the address *norwegian.com/**de**/processpages/resmake/resmakelogin*. The equivalent for a visitor to the French site would be true, but with a slight difference, the address would be *norwegian.com/**fr**/processpage/resmake/resmakelogin*. The difference here being the country code (in bold text). The expression */([^(/;\?)]*).** enables for the splitting into these country codes of being performed. This is done by extracting what comes before the second '/' and after the first '/'. An output of this can be seen in **Figure 34.** The ability to capture and split by the help of business transactions aids in to control the data set. Furthermore, the business transactions can be utilised as filters for other queries. This exact feature has been utilised for the collected data.



ROT - Step 2 - Log in or Create Profile

Name	Result Measures	Application	Splittings	Count
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	uk	7100
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.no	processpages	737
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	se	461
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	dk	357
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	en	240
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	us	237
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	de	142
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	es	132
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	pl	107
ROT - Step 2 - Log in or ...	PurePath Response Time	www.norwegian.com	fi	91

Figure 34 - Splittings of the second step in the defined user journey

these country codes of being performed. This is done by extracting what comes before the second '/' and after the first '/'. An output of this can be seen in **Figure 34.** The ability to capture and split by the help of business transactions aids in to control the data set. Furthermore, the business transactions can be utilised as filters for other queries. This exact feature has been utilised for the collected data. In able to collect data for the first step of the user visit a feature in the Dynatrace Application Monitoring software called *Visits* has been utilised. This feature has been discussed earlier in **chapter 3.1.3.** As stated earlier, first step of the user journey does not take in consideration any particular address, as long as the visitor does not land on the second step of the user journey. Then again if the user starts her/his journey on the second step, the visit would be recorded either way.

The goal, as seen from Norwegian's point of view, if for users to convert, this is to buy a ticket. When a user has bought a ticket, this user lands on a page that has a URI that ends with 'resmake/resmakereceipt/'. This makes it possible to track which users that have converted and which users that has not. In the same way that the business transactions were set up, one can set up a specific business transaction for the conversion goal. This is exactly what has been done.

4.3.2 Data sampling

Choosing the correct sample is by itself a subject one could devote an entire master thesis to. As the correct strategy could yield in better answers it is evident that much time should be put the choices and rationale behind it. Having stated this, there are different approaches that suits this research better than others. Following convention, one could choose to utilise random or systematic sampling. A random sample would ensure a relatively high degree of representativeness, and due to the fact that the samples would be nearly similar which ever day one chose, this is a sound choice. Systematic sampling would also be a good approach but less random, and thus subject to a higher sampling error rate. The possibility for sampling rate errors are more or less impossible to avoid, as one can be subjected to one of many possible influencers. For the material at hand there are several obvious risks for fluctuations in the data set. Perhaps the most eminent of the risks is not to have a normalised data set. Since the Norwegian web site is commercial at its core, it is not improbable that user behaviour follows economics incentives initiated by the company. There are more or less always campaigns ongoing that have, at least the slightest possibility of affecting users of buying that plane ticket. Furthermore, there are risks of technical issues that could prevent users from purchasing tickets at the intended time. Thus, one must open up for the possibility of users leaving Norwegian's site, in favour for a competitor. There are after all several companies in the same segment as Norwegian.

Nonetheless, a random approach has been utilised in order to avoid as much bias as possible. This does however not state that the dataset is exempted from bias. Ongoing campaigns is one such bias, geographical and/or cultural arbitraries can of course effect the data set in several different ways.

4.3.3 Data collection

The approach for the data collection of this second part of the analysis has been to gather data that initiate in the user's web browser. The software utilised for collecting the data set is called Dynatrace Application Monitoring. As discussed in **chapter 3**, a JavaScript agent is loaded on the web site as the user performs a visit. This agent then collects user specific data throughout the entire user's visit. Doing so, has enabled for collection metrics that later on could be utilised as variables in the statistical data analysis. To gain deeper insight into the data, there was a need for the data to be structured in such a way that one could follow the user from start to goal, this was enabled by creating a user journey, as discussed previously, in **chapter 4.3.1.** Norwegian's market division was consulted in order to get an overview of existing user journeys, or conversion funnels as they also are called. Based on that information a new user journey was developed to fit the needs and restrictions of this thesis and the software utilised. Once developed, the new user journey was created with the help of business transactions.

Creating all these business transactions, as described in **Table 10**, allows for collection of data from each and every one of these steps. This means that one can collect individual data in such a way that one can say something about each and every step, and at the same time have the users entire journey, visit, in consideration.

The collected data is stored and made available in two different ways. The raw data is stored on a hard drive on what is called a *Session store*, this is the primary way data is collected. Based on this raw data, the software performs analysis and stores the results of that analysis on a long-term database. For the purpose of this thesis, solely the raw data has been utilised to perform statistical analysis, as discussed in the next section. The long term stored data has also been utilised to investigate the behaviour of the raw data. More precisely it has been utilised to gain an overall feel for the data. The data collection was made in different steps, each corresponding to the respective step in the user journey.

For the first step, no consideration has been taken to where the user lands. That means that the initial page can be different from visitor to visitor. The choice has been taken with the logic that the first page a user visits differs from time to other. One can for example have one approach on *normal* occasions, but a totally different approach when utilising the page during an active campaign. For the second step of the data collection, users that have passed the login page will be registered. This means that visitors from the first step can show up on the second. This procedure is repeated for the third and fourth step and so on and so forth, until data is collected for all the six steps of the user journey. For each step of the journey, data was exported as .CSV files and cleaned in order to omit metrics that were superfluous to the data set. The utilised metrics are displayed in **Table 12**.

Table 12 - Metrics used for the data analysis

Converted	Visit Duration
User Experience	Client Errors
User Actions	Failed Actions
Client Family	OS Family
Country	Landing Page Response Time
Start Time	

The metric *Converted* will be utilized as the dependent variable for the logistic regression tests later on. The value of this metric can be either 0 or 1. Where 0 stands for a non-converted visit and 1 represents the opposite.

User Experience, classifies the overall experience of a user visits as one out of three categories, namely satisfied (1), tolerating (2) or frustrated (3).

User Actions, represents the number of actions a user has performed during a visit. What classifies as a user action can be clicking a link, loading a page and entering characters into a text field.

Client family, states which browser the visitor utilises. In total, the metric has 12 categories. In the data cleaning process, chapter 4.3.4, several of the lesser represented categories were bundled into one category called *Other*. It must be stated that the software differs between versions of the different browsers, but that these have been omitted from in the data cleaning process.

- Chrome (1)
- Chrome Mobile (2)
- Edge (3)
- Edge Mobile (4)
- Firefox (5)
- Internet Explorer (6)
- Opera (7)
- Opera Mobile (8)
- Safari (9)
- Safari Mobile (10)
- Samsung Mobile (11)
- Other (12)

Country, contains information about which country a visitor was currently in, when performing the actual visit.

Start time, gives information on when the actual visit started. The software does not take in consideration the local time of the visit. The timestamp utilised is the one of where the servers are located (in Norway).

Visit duration, is the amount of time the visit of the user was. The time is represented in seconds.

Client errors, represents if a user has any JavaScript errors during the visit. The value can be either 0 (no errors occurred) or 1 (errors occurred)

Failed actions, represent the percentage of failed actions that occurred during the visit. This metrics ties together with the metric *User Actions.* If for example a visitor has two user actions for a visit and one fails, the failed actions percentage would be 50%. A failed action can be anything from entering the wrong password to performing a wrongful query.

OS family gives information about which operating system a user has. Once again several of the lesser represented categories were bundled into one category called *Other*. In the same fashion as for the *Client family* metric, version numbers have been omitted. That is visitors utilising Windows 98 and Windows 2000, will both be categorised ad visitors utilising Windows.

- Android (1)
- Chrome OS (2)
- iOS (3)
- Linux (4)
- macOS (5)
- OS X (6)
- Windows (7)
- Other (8)

Landing page response time, represents the amount of time it took to load the first page a visitor landed on. As stated earlier the time dimension metric onLoad, was utilised. The value is recorded in milliseconds.

The exported .CSV data sets, were imported into the statistical software R, where analysis was conducted.

4.3.4 Data cleaning

Some statistical tests assume that the data is normalised in order to perform tests. Since the data collection has been set up in such a way to include all hits on the website https://www.norwegian.no, one must except that there will be a large variance is the landing page response time. The very fact that travellers will be located in different countries when visiting the site will yield in different outcomes in measured time. The country of the visit is only one of the variables that effects the outcome. One other is the type of connection the user is utilising. This is one reason for the anticipation of a large variance in response time. In the **Table 13**, below one can see that the range of the Landing page response time in fact is very large.

Table 13 - Landing page response time visit distribution - initial data set

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0	2735	4420	8637	7155	3572000

As with most data sets, some data points were missing or not collected due to unforeseen circumstances. These data points were substituted into the value *NA*, since there are methods of handling the that type of values from within R. Some data points did not make any sense to include due to the very nature of them. The reason why certain data points do not make any sense to include, is rooted in the nature of a visit. The visit by itself is meaningless if one only loads the page and leaves it right away. This behaviour is often seen in visits where the visitor has the intention of gaining information about price for a certain trip. Usually thee visitors are not actual visitors, but scripts called screen scrapers, created with the explained intention, that is to fetch a certain price. The behaviour for such a visit is often very distinct as it only has one or a set of very few user actions and a very short visiting period. The screen scraper "visit" never has the intention of converting, but to amass information for certain web pages in order to gain some benefit (often economical). There is also the behaviour of “hammering” from these screen scrapers. The term refers to performing the same task many times in a very short time period. However, the screen scrapers are often more sophisticated than that as they alter IP-address and the type of browser the visit with. Hence it would be safe to remove visits with such behaviour from the data set. At the other end of the spectrum one has visits that are very lengthy in nature. The most extreme data point with regards to visiting duration measured 2h 51min 36 s. The behaviour is not to be considered average, where the user probably has been on and off on the website without being timed out.

One, of course, runs the risk of excluding visits with the intention of conversion. For example, one could navigate to the web site on the telephone when one receives a call, or the ones boss enters the room. The reasons could be many. The question one needs to answer is where one draws the line? How many user actions is considered unnatural? Visit duration? To be more certain than not, one would have to analyse all the data points or have knowledge of which visitors, based on the IP-address they are using, are the ones with specific intentions. Even then one could risk of excluding incorrect visits, since multiple users behind one specific network often use one public IP when traversing the Internet.

To accommodate for the short visits with few user actions, all user visits that were less than five (5) seconds in duration, were excluded from the data set. Equally, visits longer than 10 000s were removed, since the skewed the data set too much. Visits with a *Landing Page Response Time*, equal to zero milliseconds were also deleted from the data set, as there is no logic behind a loading time of nothing. Visits with only one actions and more than 120 actions are equally removed with regards to the logic above. Furthermore, the data set was comprised of with visits made by users with the *Client Family* “Google Bot” and “AdsBot-Google”. These visitors are bots/scripts with the intention of performing measurements, and have their origin in Google, as can be derived by their respective names. The reduced data set is displayed in **Table 14.**

Table 14 - Landing page response time visit distribution – cleaned data set

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
191	2737	4253	5163	6604	19997

All in all, the data set started out with 132 764 data points and ended up with 88 825 data points.

4.3.5 Data analysis

The exported data files from Dynatrace AppMon were in the .CSV format. These files were later imported into the statistical software R. Initial analysis was performed in the software and the analysis were saved in .R-files. Since more or less all datasets contain extreme data, as realised after the initial analysis, the data needed to be cleaned. The data sets were in fact, very skewed and contained many more outliers and irregularities that initially thought. Several odd behaviours such as a large number of visits that lasted only one seconds and contained only one user actions, could be registered. After the inconsistencies in the gathered material were handled the data set could be analysed further. Since the second part of the analysis is made up of two different parts, univariate and multivariate analysis, the approach followed that logic. If one can say anything about the hypnotised relationship between *Response Time* and *Conversion*, the sound approach would be to analyse this initial relationship. Only to delve deeper in the same relationship in the second part.

There are some underlying assumptions of normality when performing univariate analysis. An effort to abide by a normal distribution of the data has been both aimed and stretched towards. Due to the fact that one cannot remove all too many points from the data set has led to not being able to meet all assumptions of normality. The statistical analysis has nonetheless been conducted with what is believed of being true results.

The second part of part two of the analysis conducted a logistic regression (WHY?) analysis based on the same dataset as utilised earlier. Several models were tested in order to find out which one was best fit for the data. In order to find out the best fit two approaches were utilised. A stepwise approach tests different combinations of a full model to reach a conclusion. The approach checks the marginal difference of including or removing predictor variables from the model. The result from the different steps compares the Akaike Information Criterion for the different models and suggest the model with the lowest AIC number. This approach does however not take in consideration the eventual underlying multicollinearity, which had to be checked with a Variance Inflation Factor test. The predictors with high collinearity were removed and the stepwise approach was repeated until the optimal model (for the utilised dataset) was found.

The coefficients for the logistic regression model, were explored and explained with accompanying tables and graphs. Finally, additional information from the dataset has been included as an ending to the next chapter.

5 Results and findings

5.1 Part I – Frontend vs backend

This first part of the results and findings chapter, explores a HTTP Archive data set from May 20017. The aim for this analysis is to find out where most of the user or visitor time is spent when surfing web pages. The methods used to come to any result has been to focus on the time distribution for the two metrics *TTFB* and *onLoad.*

5.1.1 Before cleaning the raw data

The distribution of top level domains for the dataset *2017_03_15_pages,* is displayed below. There are 481455 data points in the sample from the HTTP Archive. In the data sample a total of 1398 top level domains is included (IANA, 2017). The data sample contains information from 591 out of the 1398 top level domains. In **Figure 35**, below, the top 15 ones are graphed.

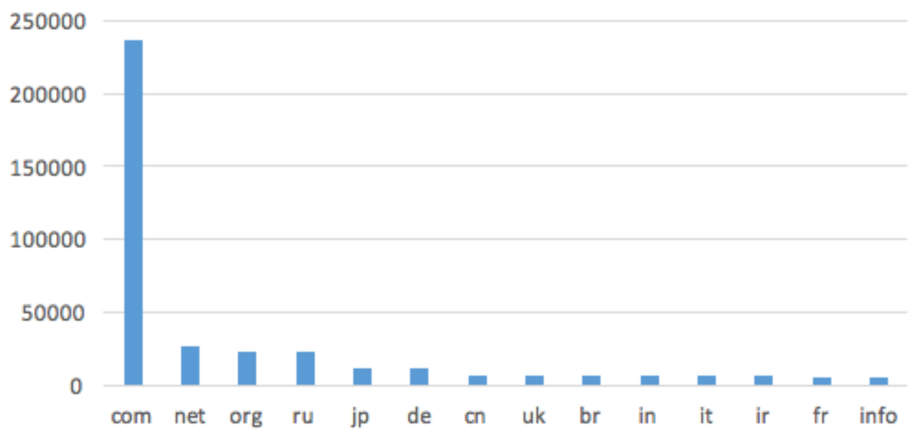


Figure 35 - Distribution of 15 most frequent occurring top level domains (15.03.2017)

The descriptive statistics in **Table 15,** has a total of 481454 observations with a very large spread for both the variables *TTFB* and *onLoad*. This is displayed clearly for both variables in the data set, where the skewedness and kurtosis is apparent in **Figure 37**.

Table 15 - Descriptive statistics before cleaning data

	n	mean	median	min	max	range	skew	kurtosis
url*	481455	-	-	Inf	-Inf	-Inf	-	-
TTFB	481455	1106,18	726	8	65535	65527	16,4	440,64
onLoad	481455	10838,14	8132	124	180129	180005	5,2	42,37

The distribution of where the average frontend and backend time resides is displayed in **Figure 36**, below. This result is not far away from Steve Souders results in 2007 and shows that most of the time loading a web page/site resides on the frontend domain (Souders 2007:3-5). Souders suggests that the distribution of frontend respectively backend data to have a relationship of 80/20. The percentages are averages of all the data points and they are in such a way that each and every in the dataset equals a total of hundred percent. That is frontend time + backend time = 100%.

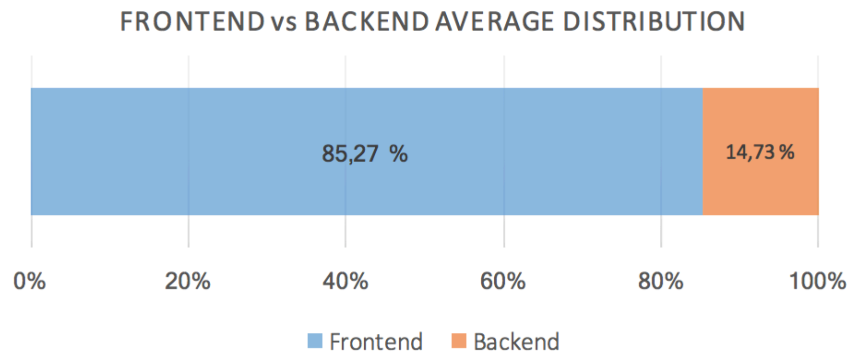


Figure 36 - Frontend vs. Backend average distribution (before cleaning data set)

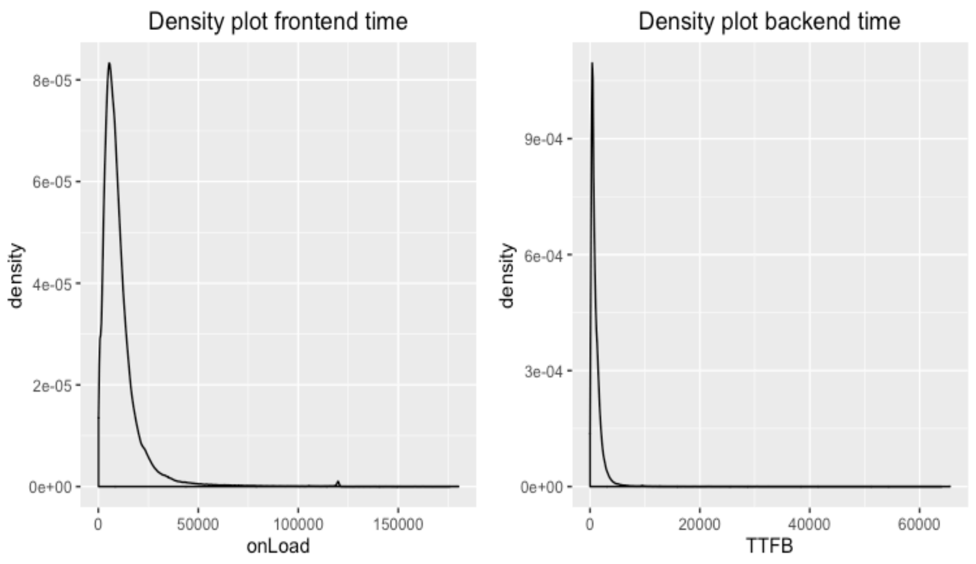


Figure 37 - Density distribution plots. Frontend and backend, cleaning data.

The graph above, **Figure 37,** shows the density distribution plots for the two variables *onLoad* and *TTFB*. As is evident are the long tails for both variables, indicating that we have a very spread dataset. The boxplot graph, **Figure 38**, acknowledges this fact.

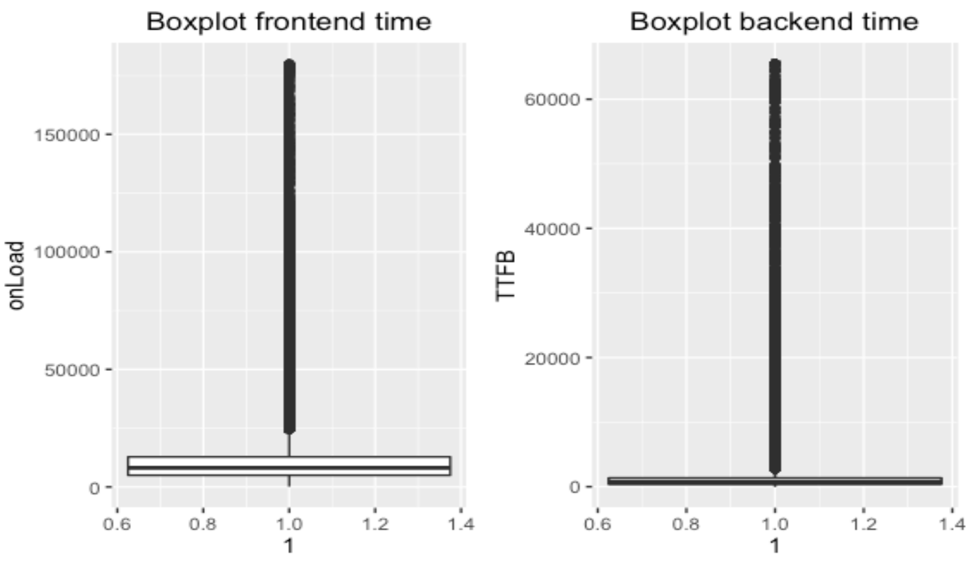


Figure 38 - Boxplots. Frontend and backend, cleaning data.

Moving forward, the aim is to reduce the extreme variance of the data points in the set. This is done in the next section.

5.1.2 After cleaning the raw data

A reduced data set with 435 787 data points, is presented below in **Table 16**. Here one can see that the skewness for is reduced a lot since we removed the most extreme *onLoad* data points from the data set. What one can see further is a reduction in the skewness and kurtosis for the *TTFB* variable. This is due to that the dataset includes three data points for each row, naturally all variables will be affected by removing data rows. Since the measures of the shape has altered to the lower, one can draw the conclusion that the rows with extreme data points in the variable *onLoad,* had equally stringent data points in the variable *TTFB*. This is confirmed by an approximate 70% reduction in range for the variable *TTFB*, and respectively a 90% reduction for the variable *onLoad.*

Table 16 - Descriptive statistics after cleaning data set

	n	mean	median	min	max	range	skew	kurtosis
url*	435787	-	-	Inf	-Inf	-Inf	-	-
TTFB	435787	997.96	711	8	20375	20367	3.78	28.60
onLoad	435787	8176.47	7488	124	20499	20375	0.56	-0.32

A graphical representation of the cleaned data set is displayed below in **Figure 39.** One can see that the *TTFB* variable still has a lot of outliers in the data set. What is hard to see from the graphs below is the somewhat altered distribution of frontend and backend average time. This is more evident in **Figure 40.**

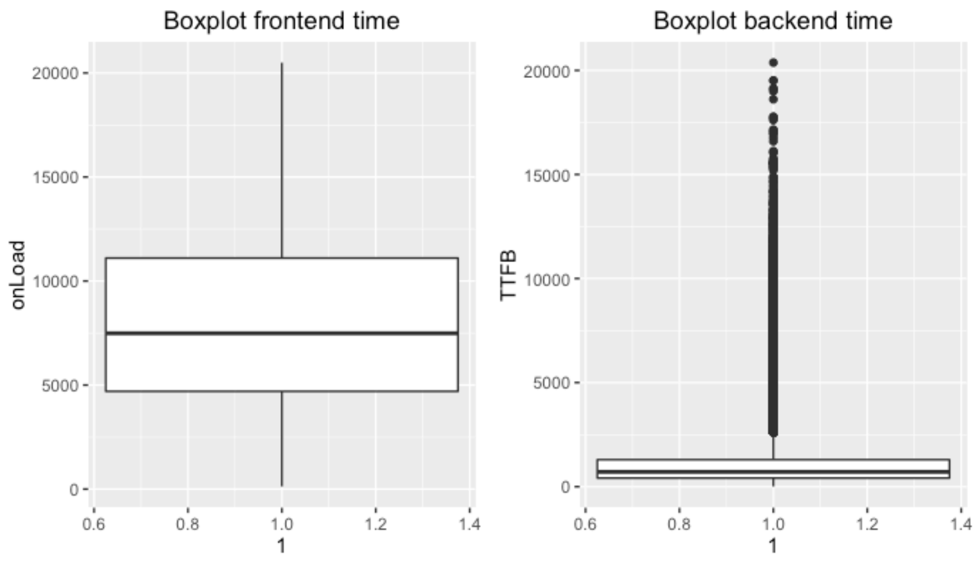


Figure 39 - Boxplots. Frontend and backend, after cleaning data.

After cleaning the data set, one can see that the average time distribution has shifted in the favour for backend time. This is by design, as explained in **chapter 4.2.2**. Still after designing for a shift in favour for backend time, one can see that the dataset hardly changes.

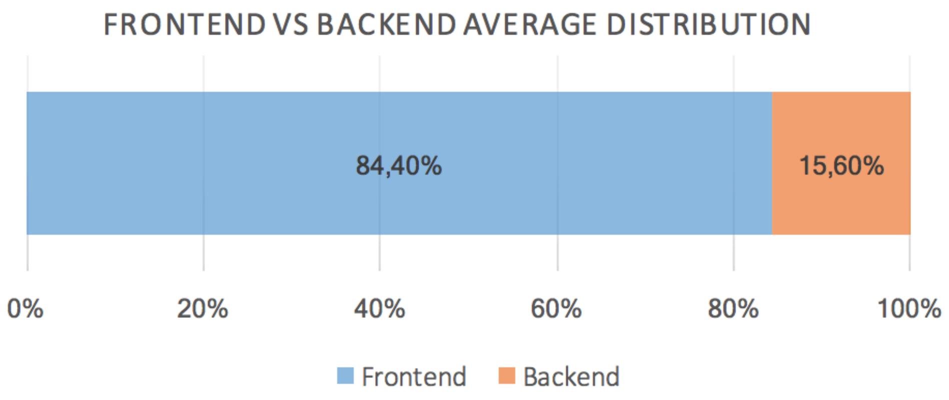


Figure 40 - Frontend vs. Backend average distribution (after cleaning data set)

5.2 Part II – Performance and conversion

After seeing that there is an evident amount of time spent on the front end as compared with the time spent on the back end, the task falls on examining the relation between *Response time* and converted visits. First of all, the aim is to see if there is any statistical substantiation between the two metrics. Moving forward from that, the research aims to explore the relationship between converted and non-converted visits. Finally, logistic regression is performed in order to see which independent variables can aid in the questions of which visits will be converted or not.

5.2.1 Univariate analysis

The data sets available for analysis are presented in **Table 17**. Each step represents a step in the user journey discussed earlier. In the table, one can see the number of data points available in each step and equally data set. Furthermore, one can see how many visits are converted or not and finally the percentage of converted visits for each step. Out of the 88825 visitors in the first step, 4346 converted. For each step in the user journey one can see that the relationship between non-converted and converted is in favour for the latter.

Table 17 - User steps and conversion rate

	n	Non-converted	Converted	% Converted
Step 1	88 825	84479	4 346	4.9%
Step 2	6 066	2687	3 379	55.7%
Step 3	5 327	1191	4 336	77.6%
Step 4	4 715	559	4 116	87.3%
Step 5	4 528	412	4 166	90.9%
Step 6	4 484	347	4 137	92.3%

An inspection of the distribution of *Landing Page Response Time* for the first step is displayed in **Figure 41**. The figure has the underlying data available in **Table 14**, in the **chapter 4.3.4**. What can be seen is the large spread between data points indicating that there is apparent skewness in the data. Most of the data points, visits to the web page www.norwegian.no, reside within the first seven seconds.

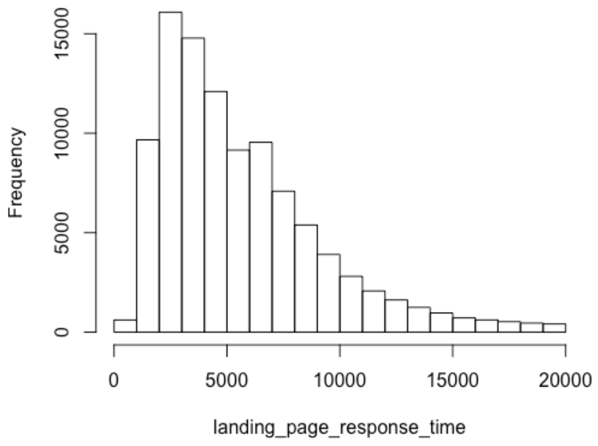


Figure 41 – Histogram of Landing page response time for Step 1

The distribution of the visits represented in the histogram above can be seen in **Figure 42**. In the boxplot, one can see that there still are outliers for both categories, both converted and non-converted. The median value for the non-converted visits is 4299ms, whereas the average is 5204ms. The values for the converted visits are 3232ms for the median and 4374ms for the average.

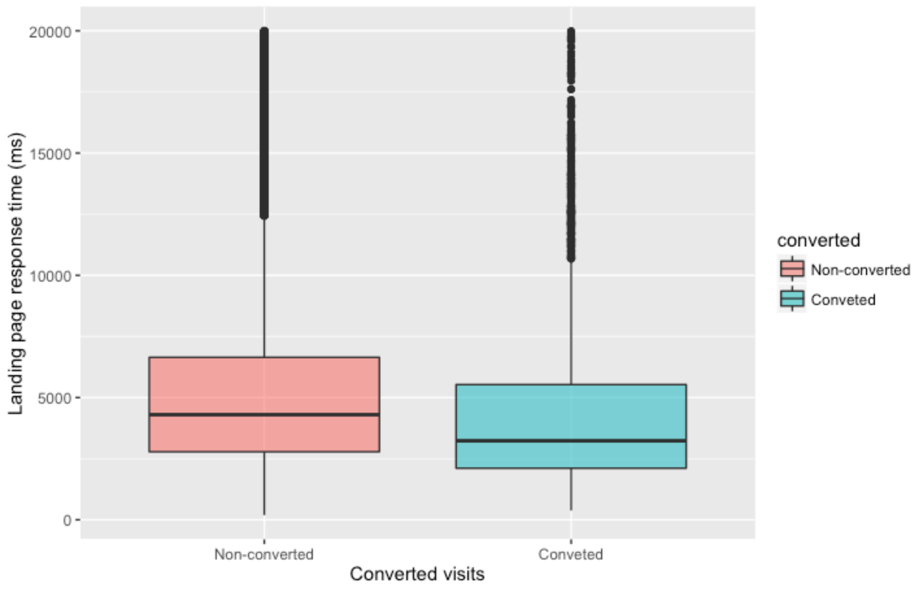


Figure 42 - Boxplot of response time for converted and non-converted visits

To test if these values hold statistically a two-sample t-test was performed with the following null and alternate hypothesis

$$h_0 \colon \mu_{lprt_{nc}} = \mu_{lprt\_c}$$
$$h_A \colon \mu_{lprt_{nc}} > \mu_{lprt_c}$$

The former states that there is not a difference in mean landing page response time for non-converted and converted visits. The alternate hypothesis states that there in fact is a difference in the two means. The output from the two-sample t-test below, confirms the difference in means with high significance as presented by the p-value (< 0,001). Thus, the null hypothesis is rejected in favour for the alternate.

t-test (Welch Two Sample t-test)
t.test(step1$landing_page_response_time~converted, alt="greater", conf=0.95, var.eq=F, paired=F)

```
 Data:  landing_page_response_time by converted
          t = 15.977, df = 4810.1, p-value < 2.2e-16
 Alternative hypothesis: true difference in means is greater than 0
                95 percent confidence interval:
                      743,9931      Inf
 Sample estimates:
               mean in group 0 mean in group 1
          5204,007                    4374,612
```

5.2.2 Multivariate analysis

As stated under the methodology chapter the main aim is to utilise the best model there is for the underlying data set. In order to do so the research is conducted such a way that a baseline is established through a generalised liner model which does not take in consideration any of the predictor variables involved. The output for this initial mode, *fit0*, can be seen below. The values per se does not tell us much if we don't compare them to other output values. A final comparison is made in **Table 22**.

fit0 – no predictor variables included
glm(formula = converted ~ 1, family = binomial(link = "logit"), data = step1)

```
Null deviance:          34703 on 88824 degrees of freedom
Residual deviance:      34703 on 88824 degrees of freedom
AIC: 34705
```

The next natural step is to utilise all variables available in the model. In doing so one can compare the initial null model with the output one receives for the full model. One must have in mind that there is a big risk of including too many predictor variables. Doing so can yield a model that is over fitted, and thus does not give an accurate picture of when one tries to predict. Furthermore, the more predictor variables one includes the more the model can be subjected to multicollinearity. Output for the model including all predictor variables can be seen in table **Table 18,** bleow.

fit1 – all predicted variables included

glm(formula = converted ~ user_experience_group + user_actions + client_family_group + visit_duration + client_errors_group + failed_actions + os_family_group + landing_page_response_time + time_frame_group, family = binomial(link = "logit"), data = step1)

Table 18 - Logistic regression model with all available predictor variables included

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-6,04	0,40	-15,09	< 0,001
user_experience_group2	0,23	0,07	3,28	< 0,001
user_experience_group3	0,91	0,12	7,50	< 0,001
user_actions	0,08	0,00	54,98	< 0,001
client_family_group2	-1,13	0,39	-2,87	< 0,001
client_family_group3	-0,09	0,08	-1,16	0,25
client_family_group4	-12,92	197,18	-0,07	0,95
client_family_group5	-0,12	0,07	-1,58	0,11
client_family_group6	-0,02	0,07	-0,24	0,81
client_family_group7	0,23	0,18	1,23	0,22
client_family_group8	-10,89	151,57	-0,07	0,94
client_family_group9	-0,32	0,07	-4,46	< 0,001
client_family_group10	0,89	0,38	2,36	0,02
client_family_group11	-1,99	0,52	-3,86	< 0,001
client_family_group12	0,87	0,32	2,74	0,01
visit_duration	0,00	0,00	10,81	< 0,001
client_errors_group1	-0,18	0,06	-2,83	< 0,001
failed_actions	1,41	0,51	2,78	0,01
os_family_group2	2,13	0,52	4,09	< 0,001
os_family_group3	-0,54	0,25	-2,12	0,03
os_family_group4	1,99	0,45	4,47	< 0,001
os_family_group5	2,40	0,40	6,00	< 0,001
os_family_group6	2,37	0,40	5,95	< 0,001
os_family_group7	2,06	0,39	5,22	< 0,001
os_family_group8	-11,21	544,85	-0,02	0,98
landing_page_response_time	0,00	0,00	1,24	0,21
time_frame_group12:00 - 18:00	0,19	0,06	3,29	< 0,001
time_frame_group18:00 - 24:00	0,27	0,06	4,80	< 0,001

Null deviance:	34703 on 88824 degrees of freedom
Residual deviance:	22402 on 88797 degrees of freedom

AIC: 22458

The result above yields a model that has ha lower AIC value than the initial model, implying that the new model is better than the previous one. However, the eventuality for multicollinearity has not been taken in consideration, nor is there any consideration taken to if the model could yield better results. The statement is simply that the full model, fit1, performs better than the previous one, fit0. Also notice that many of the included predictor variables hare of high statistical significance with p-values < 0,001. Moving forward one needs to check these results as the research has not taken into the picture, the possibility for multicollinearity and other models that could fit the data better.

As stated in the methodology **chapter 4.1.1**, it is advisable to test the model by a stepwise approach. Doing so gives the results as displayed in **Table 19,** below. The stepwise approach yields that the difference between including the variable *laning_page_response_time*, and leaving it out would result in a better overall fit. At the same time the AIC value does not differ the slightest between the two alternatives. This could be due to rounding of the AIC value. Nonetheless, the variable is included in the final model.

Table 19 - Stepwise approach for model fit1

	Df	Deviance	AIC
landing_page_response_time	1	22404	22458
<none>		22402	22458
failed_actions	1	22409	22463
client_errors_group	1	22410	22464
time_frame_group	2	22426	22478
user_experience_group	2	22457	22509
os_family_group	7	22479	22521
visit_duration	1	22515	22569
client_family_group	11	22541	22575
user_actions	1	25891	25945

To check for multicollinearity, the VIF test has been utilised. In the methodology chapter the effect of predictor variables being approximated by other predictors, has been discussed. For the model to be able to as accurate as possible in the regression coefficients, one needs to exclude the coefficients with a high GVIF value. Values considerably larger than one (1), are considered of being indicators of multicollinearity. The output in **Table 20**, shows two rows with high values, namely *client_family_group* and *os_family_group*. As compared to all the other output values, these must be considered as considerably higher than the others, and must therefore be omitted from the final model.

Table 20 - Variance Inflation Factor test for model fit1

	GVIF	Df	GVIF^(1/(2*Df))
user_experience_group	1.265813	2	1.060700
user_actions	1.693613	1	1.301389
client_family_group	429.111667	11	1.317232
visit_duration	1.572474	1	1.253983
client_errors_group	1.087474	1	1.042820
failed_actions	1.179644	1	1.086114
os_family_group	410.309534	7	1.536918
landing_page_response_time	1.114287	1	1.055598
time_frame_group	1.007549	2	1.001882

Taking in consideration the found multicollinearity, but ignoring the result from the stepwise analysis yields an optimised model, *fitFinal,* for the given data set. The output from said model can be found in **Table 21.**

fitFinal – optimised linear regression model

glm(formula = converted ~ user_actions + user_experience_group - client_family_group + visit_duration + client_errors_group + failed_actions - os_family_group + landing_page_response_time + time_frame_group, family = binomial(link = "logit"), data = step1)

Table 21 - Optimised logistic regression model - fitFinal

	Estimate	Odds ratio	Std. Error	z value	Pr(>|z|)
(Intercept)	-4,69		0,05	-87,59	< 0,001
user_actions	0,07	1,07	0,00	56,09	< 0,001
user_experience_group2	-0,25	0,78	0,06	-4,13	< 0,001
user_experience_group3	0,60	1,82	0,11	5,23	< 0,001
visit_duration	0,0004	1,00	0,00	18,90	< 0,001
client_errors_group1	-0,35	0,71	0,06	-5,66	< 0,001
failed_actions	2,05	7,80	0,41	5,07	< 0,001
time_frame_group12:00 - 18:00	0,27	1,30	0,06	4,79	< 0,001
time_frame_group18:00 - 24:00	0,36	1,43	0,05	6,61	< 0,001

Null deviance:	34703 on 88824 degrees of freedom
Residual deviance:	25284 on 88816 degrees of freedom

AIC: 25302

A comparison of all the different models is displayed in **Table 22**. The output shows that the model with the lowest AIC is indeed *fit1*. However, it has been shown that that particular model was subjected to multicollinearity and was not a viable model in the end.

Table 22 - Logistic regression model comparison

model	Null deviance	Residual deviance	AIC	ΔAIC
fit0	34 703	34 703	34 705	-
fit1	34 703	22 409	22 463	12 242
fitFinal	34 703	25 284	25 302	9 403

Utilising the stepwise approach and multicollinearity correction has reduced the initial model from twenty-seven (27) predictors, into eight (8) predictors that are statistically significant. The resulting logistic regression model is displayed below.

$$\begin{aligned} Log = \left[\frac{\pi_i}{1-\pi_i}\right] = &-4{,}69 + 0{,}07x_{user\_actions_i} - 0{,}25x_{user\_experience\_group2_i} \\ &+ 0{,}6x_{user\_experience\_group3_i} + 0{,}0004x_{visit\_duration_i} - 0{,}35x_{client\_errors\_group1_i} \\ &+ 2{,}05x_{failed\_actions_i} + 0{,}27x_{time\_frame\_gropu2_i} + 0{,}36x_{time\_frame\_group3_i} \end{aligned}$$

An assessment of how well the model preforms is displayed below, in **Figure 43.** The ROC curve shows the ability to classify those visits that have converted and the visits that have not converted. The result strongly indicates that the model *fitFinal* is able to predict the outcome of a randomly picked visit. The performance of the model rises well above the diagonal line, indicating that it is doing much better than a random guess. The Area Under the Curve (AUC), is the percentage of randomly drawn pairs which are true, that is the model correctly classifies visits into the correct category. With a value of 93,7% it must be said that the fit of the model is rather good at predicting the outcome of a visit.

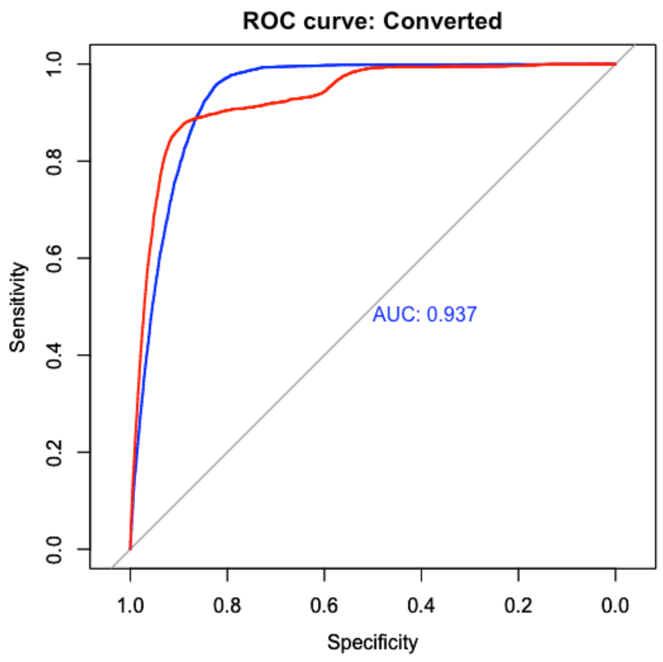


Figure 43 - ROC curve for conversion (models fitFinal (blue) and fit1 (red))

Based on the results of the logistic regression one can see that there are two predictors with negative coefficients. The variables are *user_experience_group_2* and *client_errors_group1*. The first of the two predictors tell us that when the user experience goes from 1 (satisfied) to 2 (tolerating), the chance of converting the visit is reduced. Likewise, if a visitor experiences client errors (errors = 1) as compared with no errors (errors = 0), the chance is equally reduced. All the other predictors have positive coefficients, indicating that they affect the outcome of a conversion in a positive manner. It must be noted that the predictor *visit_duration* has a very low coefficient, thus yielding little to no effect on the outcome. This predictor is included in the final model as the AIC value was considerably higher when excluding it. The remaining five predictors all have positive influence on the model. The predictor *user_actions* indicates that there is a 7% increase with each additional unit increase. When it comes to the predictors *user_experience_group3* and *failed_actions*, there is a quite large discrepancy in logic. For the former, the interpretation indicates that a change in user experience from tolerating (2) to frustrated (3), yields in an increase in chance by 82%. Right off the bat this makes little sense, but upon further analysis the answer is a bit more logic. The group wise distribution for the different user experience types (Satisfied, Tolerating and Frustrated) in **Figure 44**, shows that frustrated visitors are 2,6 times (10,06/3,86) more likely to convert as compared by the tolerating visitors. The explanation behind what the different User Experience levels mean, is available in **chapter 3.1.3.2.** For the 134 frustrated visits that converted, 54% had a *Landing Page Response Time* slower than four (4) seconds. The combination of having a slow response time and likewise convert goes against the current of the thesis. The explanation of this can be located in the domain of how a frustrated visit is categorised. Either the last action of the visit failed or was frustrating (slower than four seconds) or more than 50% of all actions were frustrating. Attaining the exact reason for a frustrated visit requires more analysis of the specific visits. The predictor *failed_actions*, is an anomaly as the odds ratio for the predictor is at 780%, indicating that an increase in failed actions would be the most certain predictor in the model.

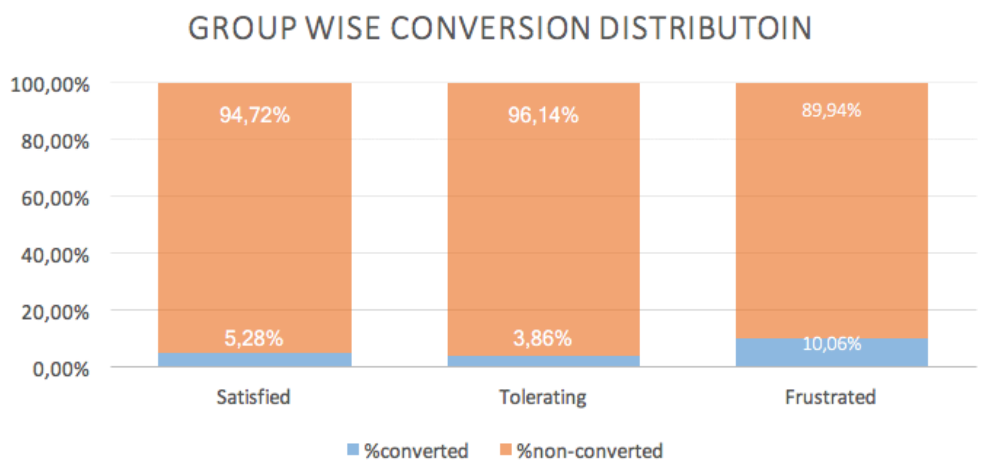


Figure 44 - Distribution of converted/non-converted for User Experience

The final two predictors time_frame_group(s) 12:00-18:00 and 18:00-00:00, also have positive values and an increase in odds ratio when moving from one group to the other. The chance of a conversion is 30% or 43% respectively for the two groups, as compared by the first group 07:00-12:00. The distribution between the different time bins is displayed in **Figure 45**, and shows a slight difference in the percentage of converted visits, as the day progresses. The number of visits for the respective bins is 16162, 35526 and 37137. These numbers can indicate the surfing behaviour of visitors as a lower number of conversions would be anticipated since people tend to have later habits during the summer months (ref data sample). Furthermore, the timestamp for the visit does not take account for the visitors' local time. The timestamps utilised are the ones that are local to the web site, and does not compensate for time differences for other time zones. This can of course skew the data in either direction depending on where visitors of the site reside at that particular instance in time.

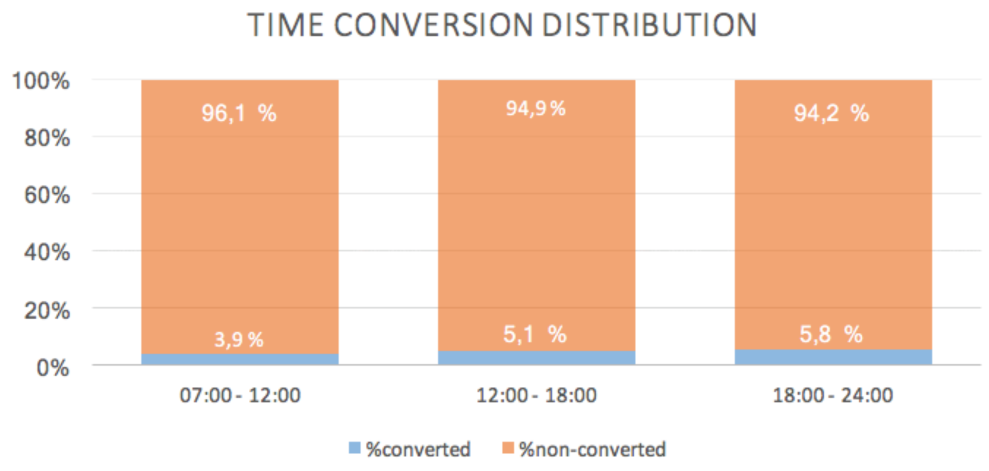


Figure 45 - Distribution of converted/non-converted for time periods

The timestamps utilised are the ones that are local to the web site, and does not compensate for time differences for other time zones. This can of course skew the data in either direction depending on where visitors of the site reside at that particular instance in

5.2.3 Additional results

There are reasons for exploring the data bit more as there is more value to be extracted from the existing data sets. In the following sections, precisely this is done.

The first graph represents the different time bins and the value of the conversion rate for each and every one of the time bins. There is a cut-off after the 15 first seconds, since it would have been superfluous to include all existing bins to the graph. Needless to say, the tail had been longer than displayed in **Figure 46**. One can see that the conversion rate for the four (4) first seconds is higher than for the rest of the presented data, the exception being the time bins "12-13s" and ">15s".

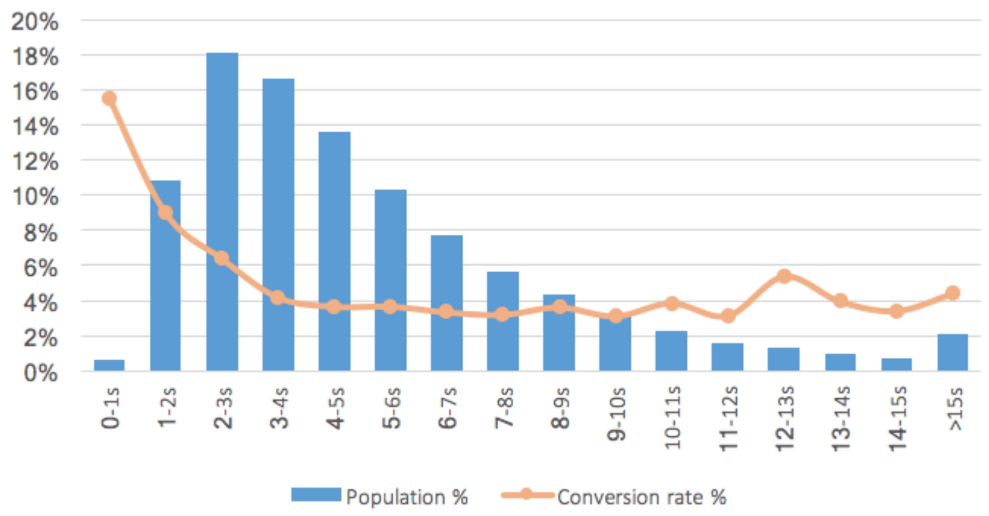


Figure 46 - Conversion rate percentage vs Response time

Figure 47, displays the conversion rate for each step in the user journey. The numbers indicate that the conversion rate increases for each step taken in the conversion funnel. The largest deltas can be found when moving from the first step to the second and on to the third.

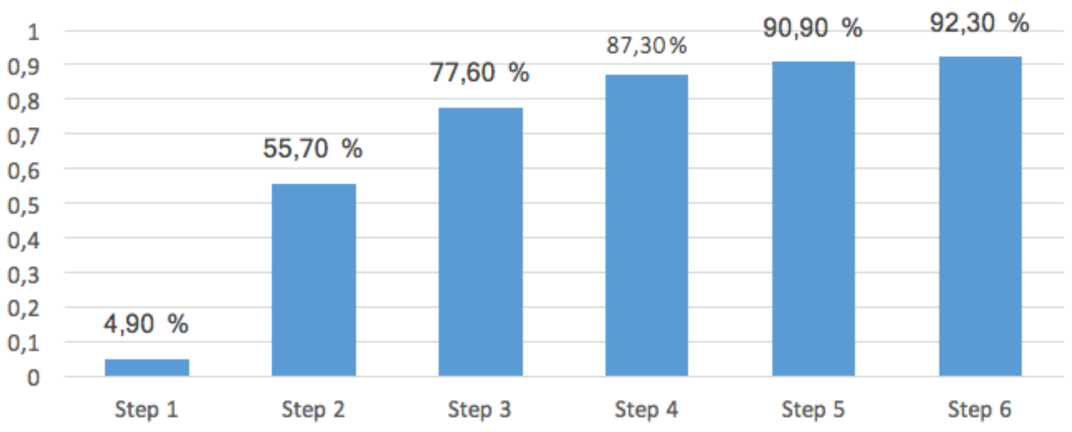


Figure 47 - Converted visit percentage for steps in user journey

Figure 48, displays the conversion rate with regards to the number of user actions performed by visitors. There is a big difference in the median value for the two different categories, where the number of user actions reside around seven (7) for non-converted visits and 28 for converted visits.

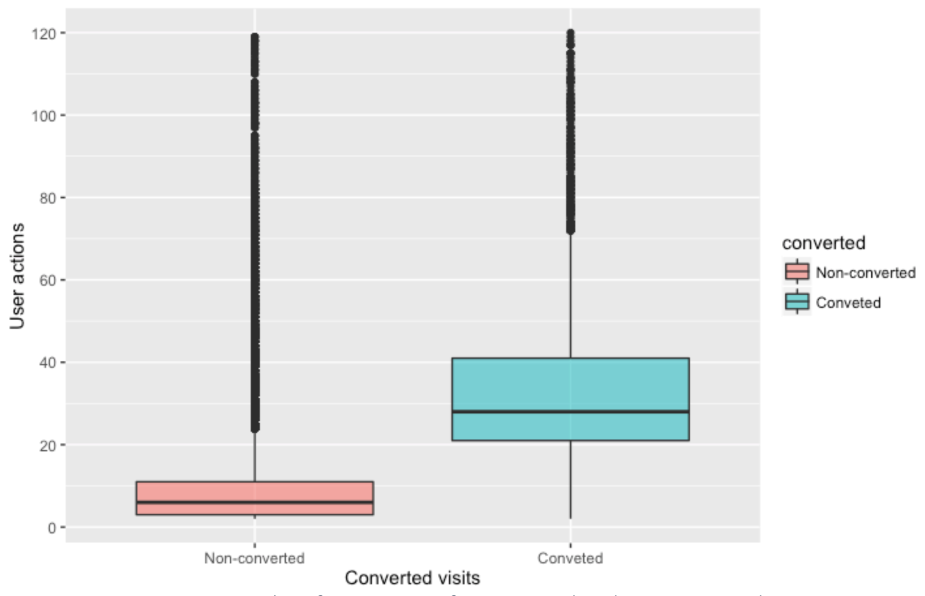


Figure 48 - Boxplot of user actions for converted and non-converted visits

Figure 49, displays conversion rate with regards to how long the visits are for converted and non-converted visits. Visitors that convert stay on the site around 18,5 minutes (~1100s), as comparted with non-converting visitors who hardly stay on the site. The latter is perhaps not a surprise.

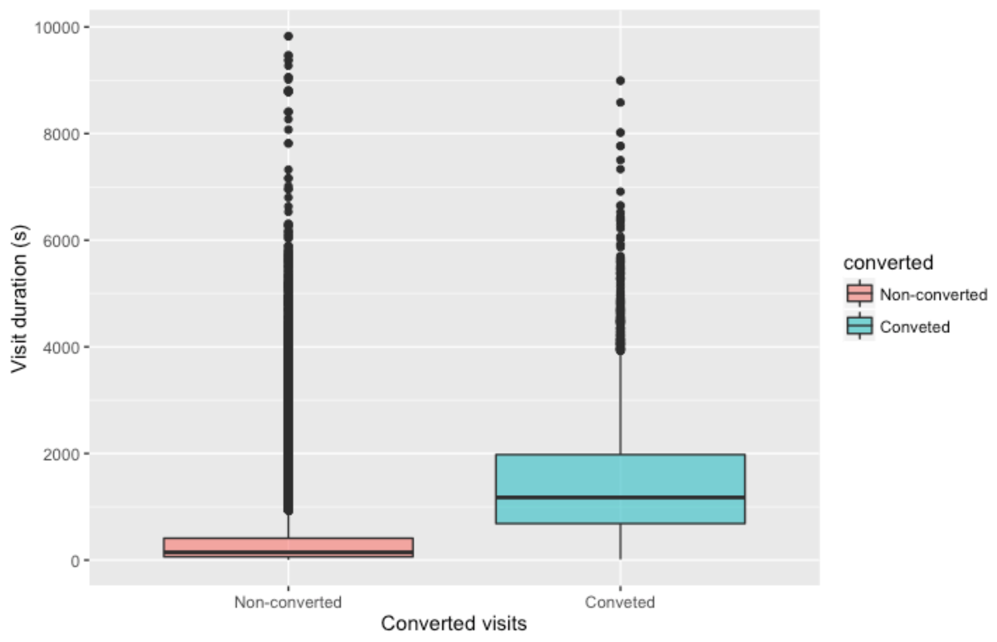


Figure 49 - Boxplot of visit duration for converted and non-converted visits

6 Discussion

6.1 Results part I - Frontend vs backend

The results from the first part of the analysis shows that there in fact is there is a large difference in the frontend and backend average response time distribution. Out of all this frontend and backend time around 80-85% resides in the former domain, and 15-20% in the latter. The hypothesis is also presented by Steve Suoders, although no method for reaching the outcome is presented (Souders 2007:3-5). One can see in both **Figure 37** and **Figure 38**, that the initial data set is heavily skewed for both the examined variables *onLoad* and *time to first byte (TTFB)*. If one considerers this particular skewness is not directly strange that it exists. Longer load times has no evident function or behaviour that will benefit the end user. Page load times, here *onLoad*, and *TTFB*, will therefore have a natural skewness towards shorter load times. To reduce the skewness, the dataset was altered in such a way that the most extreme points were removed from the dataset. The reduction of approximately 50 000 fewer data points resulted in a decrease of almost 70% in data point range for the variable *TTFB*, and respectively almost a 90% decrease for *onLoad*. Reducing the dataset is by itself not always a good idea as it can affect both the outcome and interpretation of the data. Although this is apparent that the data has been effected by the data cleaning procedure, it is still viable to assume that dataset is not unreliable. Since the both variables exist for each data point, one can draw the conclusion that the rows with extreme data points in the variable *onLoad,* had equally stringent data points in the variable *TTFB*. Comparing the average frontend versus backend distribution with help of the two variables, as shown in **Figure 36** and **Figure 40**, the difference before and after cleaning the data has little to say on the both the outcome and the interpretation. The conclusion to assume that the fact front end time is overrepresented as compared to backend time, is therefore safe. This can be universalised for web sites in general.

The research hypothesis has been tested and the findings from verifies this statement to be true for a large subset of Internet web sites. Of course, one cannot test all websites available, but the subset of around 500 000 web sites included in the test is to be considered a foundation that is more than good enough. Out of the large subset of websites almost 250 000 of these servers are .COM domains. One could argue that the overrepresentation skews the data in a specific manner, but there is no reason to assume that only .COM sites undergo any particular skewness that would favour frontend time as compared with backend time. The same reasoning would apply for both different international domains (e.g. .JP, .ES, .NO etc), and specific commercial segments. For example, one could reason that the aviation sector has a particular distribution of front and backend. There would not be any obvious reason to assume that aviation companies behave in other ways that any other segment in online business. The same test could be set up with the limitation of stratified samples, however that would only make the end result more inconclusive as this would not say anything about other strata. Following that logic, the skewed representation of .COM domain should have little to no impact on the dataset. The dataset utilised when testing the frontend versus backend time distribution, is limited to only one month of data. The choice could of course impact the final result and it is a drawback that such a limitation influences the test. Nonetheless strengthening the 80/20 hypothesis is the fact that the number or resources on and requests to any particular website have been increasing over the years, as displayed in. The larger amount of resources would of course result in more to load in the end users' browsers, thus resulting in longer load times. Figure 12 displays that the total transfer size has risen from 702 kB in 2010, to about 2 300 kB in the end 2016. With this particular fact at hand and that latency will increase due to the number of requested resources, it is more than safe to assume, that even though only one dataset it utilised, the result still is viable.

The variables that could affect the end user response time, are many in the number. In fact, there are some many that one needs to thin out the plethora of choices in order to get to the heart of the matter. Analysing the data after the fact that it has been collected, would nonetheless be rather devoid of value if one does not know the data one collects. In the theory chapter, several aspects of end user response time and latency it is comprised of is discussed. There it has been shown that latency can reside in several domains. Said domains can be comprised either components, load and/or geography. Starting at the data centre, where the servers that provide the end user with the bits and pieces that the web page is comprised of, exist; one speaks about back end time. As discussed, this time, *TTFB,* includes the time it takes from a user initiated request until the time the user sees anything in the browser). Also included in the domain in of backend time is the actual processing time of the packets traversing the data centre the time it takes for each node (switch, fire wall, load balancer amongst others) to process the response (data packets). Should the data centre be under a lot of pressure and not be able to handle all incoming request, such as in the case of queueing due to too many simultaneous sessions, the outcome for the end user would be experienced as a delay. Adding geography to the picture, users at farther distances from the web server, proving the content would require even longer time to receive an answer to the initial request. In the same way one has backend time, one has frontend time as well. Finding which variables to track thus is an essential part in order determine how to interpret the outcomes.

Having verified this first hypothesis, one can assume that the front and backend distribution is the general case for websites. In a world where hardware would be cheaper one could of course focus more on purchasing better hardware to improve the underlying requests (backend time). Although this is not reality, one cannot disregard looking in to and optimise the data centre/backend time as it represents 20% to the distribution. The distribution rather states that the benefits are larger if one focuses on frontend time.

6.2 Results part II – Performance and conversion

The results from the second part of the analysis shows that there is in fact an over-representation, almost 60%, of converted visits for end users that have a web page load time between 0-5 seconds. The rest of the converted visits are spread out over the rest of the time spectre, as can be seen in **in Figure 46**. This is strengthened by the univariate analysis that shows that there is a difference between median and average *Landing page response time* for converted and non-converted visits. The results shown in **Figure 42**, are in rather big favour of the converted visits, where the median time is approximately one second faster and average response time is approximately 900ms faster. Although the results are rather evident and not in need of further statistical analysis, due to the large data set, the results are also strengthened by a statistical Welch two-sample t-test. The numbers imply that one can expect to find more converted visits with faster page load times, as compared with slow page load times. The interpretation of the results from both figures, is that speed truly is an important factor for conversion rates.

Being a bit critical about the outcome, and one should, it can be said that the Norwegian web site is fast at a regular basis. As a matter of fact, the waterfall chart in **Figure 15**, shows that the load time from London is around five (5) seconds for www.norwegian.no. The site is hosted in Norway and thus will on average serve web pages in less than the five seconds for the web page visit originating in London. There however additional metrics to inspect in the waterfall chart. For example, one can rather than inspecting the *Document complete* metric, utilise the *Render Start* metric which is displayed as a vertical green in the waterfall chart. The metric represents when the end user can start interacting with the web page. The difference between the triggering of the metrics here is a mere 500ms, but even that small of a result has consequences. A study performed by the Financial Times shows an almost 5,0% drop in mean article views when the site speed is reduced by 1000ms (Financial Times, 2016). One could consider it be superfluous to be picky about the differences in metrics and which ones to choose but that is all but true. In the previous subchapter, it is highlighted that choosing the correct metrics in order to keep track of the underlying desired outcomes, is essential. If one for example want to keep users browsing several pages on site in order to get them to see adds, it would be important to measure the session depth. This in addition to measuring the speed of the individual pages of the site. In our case the desired outcome is to measure the conversions with respect to speed of the Norwegian web site. In doing so the choice, due to limitation is the software has been to focus on a metric that is labelled *Load Time*. This metric is perhaps not the most obvious choice as it is a measure of when a page is completed to the point where all the static content; images, stylesheets, scripts and text. The reason for the metric not being completely obvious is that pages display content in the web browser before the web page triggers the event that correlates to the metric *Load Time*. A better choice than utilising the mentioned metric would be to utilise the *Start Render* metric, mentioned above. In the context of responses taking too long time, how does one know when the end user is abandoning the slow version of the site as compared with the faster version of the same site? After all, on one side of the spectre one has conversion and on the other side one has end user dropping off. According to a study performed for Akamai by Forrester Research, 47% of the end users expect web pages to load in two (2) seconds or less (Akamai, 2009). This is a decrease from the four (4) seconds that was the threshold for end users in 2006 (Akamai, 2006). If we should consider the results from search engine experiment by Brutlag et al., the line in the sand is around 3 seconds, thus corresponding with the Akamai study in 2009 (Brutlag, Hutchinson & Stone, 2008). The results do however not give an understating about why this happens. To understand this notion better one must include the dimension of experienced time. According to Card et al. there is a concept of certain time constants in human-computer interactions. Some of these constants have been represented in **Table 6.** The constants are *Perceptual processing* (0-0.1s), *Immediate response* (0.1-1.0s) and *Unit task* (1.0-10.0s) (Card et al., 1991:67). Breaking things down more, Card et al. imposes that in the same way that you address a human being and expect an answer within a certain timeframe, so do you with computers. Card et al. further state that human flow resides within the time range between two (2) to five (5) seconds (Card et al., 1991:70-72). This statement somewhat goes against the findings of Brutlag et al. where the increased risk/chance of switching search engine increased by 1.5 times, already after three (3) seconds. The latter research does not say anything about the end users state of flow or not. Nor does it say anything about displaying any sort of indicator notifying the user about if the pages with longer page load time was actually loading. Which of course can affect an end user's choice, according to the several sources discussed already. The action of a user switching search engine or aviation company when a request takes too long time without the end user being notified, could be a result of an end user's impatience.

From this springs that one cannot generalise about conversion only based on speed as such. There needs to be a focus on breaking down time into smaller pieces. We have in **chapter 2.4**, been discussing both perception and tolerance with regards to response times. This opens up for at least three aspects to consider when working with load times and performance, namely:

- Load time
- Presentation dimension of time
- Experienced time

First off, one has the actual load time as such, which needs no further explanation. Secondly, one has a presentation dimension of time which cares for the usability of a web page/site. Thirdly, one has the aspect of the experienced time. All three aspects of course are interconnected, although the aspect of load time resides within a more technical domain than its two kins. Experienced time as it is needs some sort of reference in order to be subjected to interpretation, a web page not being an exception. In the early days of mobile browsing one would have the reference of things being fast within the technological limits of the day.

Comparing the browsing experience of 2010 with the one of 2017, would most probably be in disfavour of the former due to systematic asymmetry. This simple reason for this is that the expectation has shifted du to one's initial frame of reference (Hällström, 1985). This frame of reference acts ties in to the presentation dimension of time, and incorporates every aspect of usability during the user experience. There is a large downside to not including the final two dimensions when considering a site's speed and equally load time. The interconnectedness of the aspects really needs to be addressed as one cannot have one without the other. This also ties in to the quality work of the design and includes the actual design of site and the user journey. The discussed dimensions have been disregarded in the master thesis, as this was an initial limitation. Investigating these two dimensions could have been done by an A/B test or A/B/n test, all depending on the number of test/control groups one desires. In the A/B test one utilises two or more different layouts in order to compare the outcome of a hypothesis, e.g. utilising different font sizes for a heading. An example of an A/B test is shown in **Figure 50**, below.

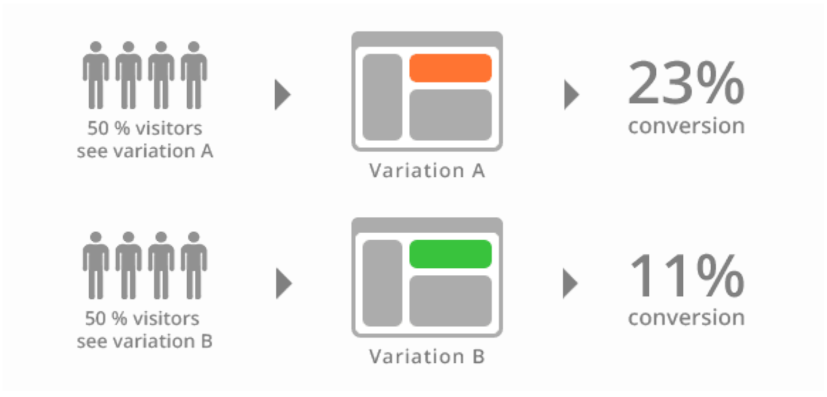


Figure 50 - Example of A/B testing

In the situation of trying to answer the problem statement of the master thesis this would have been possible by testing parallel versions of the same website. One of these web sites would be the regular one and the second one would have had a slight delay, latency, imposed on it. Utilising such an approach would have the possibility to give answer to additional dimensions for the given end user visits. In general, one can utilise an A/B test to answer if a version of a site functions good or better, it would still be immanent to have knowledge about the speed itself and its effect on conversion. Accordingly, an A/B test would not alone be suitable for researching the problem statement at hand. To nuance this more, one can consider all the time aspects and add the economics into the equation. A comparison A/B test between two specific pages, with both pages having the same web form, the exception being that one has one option less than the other, could act as an example. A sample calculation would be as follows:

The conversion rate for the original web from with ten (10) fields to fill in has a conversion rate of 10%. Likewise, the second page has nine (9) fields to fill in and a conversion rate of 11%. Additionally, one can consider a base line of 200 000 visits per year that complete the form and consequently contribute to the conversion rate. Removing one question from the web form, as with the second page, would result in an additional 2000 people completing the web form. Adding a value for the business for each web form completion and equally converted visit of for example 200 NOK, would yield in a cost of 400 000 NOK per year when asking that additional question.

One can of course regard the fact that the one additional question could yield in deeper insight about one's customers. Nonetheless, even if this is just an example it opens the door of considering how usability ties in with speed, and that one cannot consider one without the other. It thus, becomes important to consider the trade-offs between conversion rates and speed on one hand, and usability and conversion rates on the other hand; maximising the conversion rate or optimising the same. Even though that the thesis focuses on the finding if site speed has any effect on conversion rates, it is considered to be very important to highlight additional, and omitted dimensions that contribute to the overall picture.

As stated above, data has been collected at one particular collection point. From that data, a user journey has been outlined, as displayed in **Table 17.** The journey consists of users starting at different pages, continuing with filling in personal information and flight details and eventually purchasing a ticket. This particular user journey has been defined by Norwegian ASA. The data from **Table 17**, shows that the number of users across the user journey steadily go down from step two (2), to step six (6). Although the reasons for this could be many, this has most certainly more to do with natural browsing behaviour than slow response times. What furthermore is not directly surprising is the fact that there is a large drop from the first step to the second step in the user journey. The most likely contributor of this is that of window shopping. End users, simply want to browse around to see what is available.

Results from the stepwise approach in the multi variate analysis, reveals that some of the initial variables included in the model were superfluous going in to the analysis. This is perhaps not directly surprising as one can expect to have multicollinearity when utilising a multitude of variables. Variables that were omitted form the final logit model were such in nature that they represented *OS family* and *Client family.* In the case of testing the different logit models, these variables only provide meta information about the visit. There is no direct nor obvious correlation between utilising a specific operating system (OS) and/or any specific browser (client family/browser type). The only apparent browser types that differed from the rest was that of *client_family_group10*, which is Safari and *client_family_grou12*, which is the others group. This does however not indicate anything more than that many end users how convert utilise the browser Safari. It does not say anything about the web page speed per se. Analysing further it is a bit surprising to find the effect of *user_experience_group3* having such a high odds ratio. The variable corresponds to a visit categorised as being frustrating. This means the following:

- The last action performed, failed
- The last action was frustrating
- More than 50% of all actions were frustrating

This is described in more detail in **chapter 3.1.3.2**, but in short that somewhere along the user journey the user experienced long loading times and/or errors. In a separate context is also surprising that a good indicator for the logit model, due to the variables high odds ratio, that the number of *failed_actions* during the user journey actually increased the odds for at conversion. Putting *user_experience_group3* and *failed_actions* in the same context reveals that they are two sides of the same coin as many failed actions contribute to a frustrating visit, and frustrating visits, probably has failed actions in it. The failed actions can for example, be a result of the end user inputting incorrect values in a from and/or the Norwegian web site having issues in some way. The number of converted visits in the light of frustrated visits does in no way make any logical sense. Other possibilities one needs to consider is the effect of customer loyalty and/or price. A survey performed by KMPG show that price and/or promotions is the largest decision factor when performing online purchases (KPMG, 2017). As Norwegian is an aviation company that targets the low-cost, it would be surprising if not a part of its customer base choses the company for its prices.

Glancing at the variables *time_frame_group12:00-18:00* and *time_frame_group18:00-24:00*, reveals that there is an increased chance of converting if an end user performs the visit between 12:00-18:00. The odds are increased slightly more if the visit is performed between 18:00-00:00. The reasons for this could of course be many. Comparing the two first groups with the third group, between 00:00-12:00, one can anticipate that there is less traffic on the site during that particular timeframe. Having it that the data set only includes www.norwegian.no, it is hardly surprising to find that the number of visits is lower. This is most probably due to natural human habits, and is indeed emphasised by **appendix section 10.5.1**, which shows a dip in user visits between 02:00-07:00. The difference in odds ratio between *time_frame_group12:00-18:00* and *time_frame_group18:00-24:00,* 0,78 and 1,82 respectively, shows that the possibility for conversion, as seen from the web page, would be affected negatively for the 12:00-18:00 group. Why this is the case could be several, larger load on the web site, not many performing purchases at that particular timeframe, amongst others. Had the dataset been wider spread over several weeks, it would have been possible to do a more exact analysis on the end user behaviour. Finally, the 18:00-24:00 group has the highest conversion rate of all the three groups. Once again one could hypothesise about that particular outcome as with the previous groups, but with the limited data set it would be difficult. The final variable had a positive odds ratio is *user_actions*. The outcome is perhaps not directly surprising as the end user needs to perform a certain amount of user actions in order to complete the defined the user journey. An increased number of user actions would therefore be an intrinsic part of finalising a ticket purchase. Comparing the different logistic regression models, it is not that surprising, but nonetheless interesting to see that fewer predictor variables actually help explain the final conversion better. This is could be due to the very nature of some of the predictor variables that actually don't contribute with any specific speed data for the models, as discussed above. Facing this reality, it would have been to the advantage of the analysis if different predictor variables would have been selected. Knowing beforehand all the variables that can affect a conversion rate is not a simple task, since there is quite a few to choose from. For example, latency has been discussed in several of the chapter of the thesis. Including aspects of that particular variable, like for example geographic location of the end user would probably have an effect on the possibility of conversion. The simple logic behind this is that longer geographical distances are subjected to longer delivery times for the same data as compared with shorter distances.

6.3 Further discussions

The discussion has mostly has been about the technical details sounding the actual results. There is an equal need to augment content at a higher level and include that of the impact of the results. Examples from big companies show that it is important to consider technological gains/losses and the impact of these. Even though the focus has been on the aspect of performance, those of capacity and availability are equally important to consider. If for example one has really high load or an outage on one's platform, the effects would touch on all the three aspects. An example of the contribution performance has on capacity can be taken from a case study carried out by the NCC Group in 2015. In that study, the company under the loupe is Seatwave, had their website optimised through reducing page size and the number of sub requests for each page request. The reduced page sizes had a direct effect positive on the bandwidth, thus reducing the database CPU requirements by 75%. As a consequence, the concurrent user ceiling increased by 300% (NCC Group, 2015). Even though such an improvement is not the general case it shows that even small changes can have dramatic effects. An example of the consequences of a 15-minute availability outage for Amazon.com, has been estimated to cost the company an approximate $2.5 million in sales (Reuters, 2013). It is not clear if this had anything explicitly to do with performance. Consider the same scenario with recurring high load contributing to unavailability of the web site. The overall availability of web sites and the machines that host is still today a factor to consider when speaking about sales and the bottom line. Only in 2017, DNB, a Norwegian bank had 17 outages (e24, 2017). Likewise, Facebook have experienced several outages throughout its existence (The Guardian, 2016). What the cost of these outages is, may perhaps be more difficult to estimate for the Facebook, and less so for DNB, but this shows that no company has a free pass when it comes to the downside of availability. Further performance tests carried out by the Microsoft search engine Bing, shows that a 2000ms second delay on their website resulted in a -4.3% revenue per user and a decrease in user satisfaction by 3.8%. Also, interesting from the same study is that a metric called *time to click*, increases by 1900ms when the artificial delay increases from 500ms to 1000ms (O'Reilly, 2009). Even if the reason for the behaviour is not clear, one can assume that the end user is less engaged in the actual interaction. Or put in other words, they have lost their interest.

The economic downside of not paying attention to site speed, performance and availability can be grasped when evaluating the different examples above. It is rather safe to assume that almost no website is immune to the dimension of performance. If this is the case, the notion of controlling web site speed becomes a companywide challenge. The reason for the argumentation is that the underlying code for websites are produced by developers. The code is then served by web server which are maintained by people in the operations department. Also, the code is developed due to business functions that needs to be addressed. There are of course more entities that comprise a business, but the three included can be considered a part of any technology company. The upside is that once one has control of the variables that affect the various functions of a website, one can also control the effect of the underlying code has on the same website before new functions are deployed. In the case of retaining one's customer base, keeping them engaged and keeping the brand image alive, it becomes difficult not to focus on optimising web site speed and the elements that comprise it.

6.4 Method

The dataset utilised in the first part of the analysis has been retrieved from the HTTP Archive. There are no other similar public tools to utilise in order to gain information on a large set of websites. As the intention was to analyse the distribution of frontend and backend time, the alternative to the utilised dataset would have been to perform manual performance test for a large enough set of web sites. Since the HTTP Archive does this already, it has been considered the best alternative.

The data collection for the second part of the analysis has focused on collecting data with the help of Dynatrace AppMon. More precisely the UEM part of the software has been utilised. There are several software alternatives to the existing choice, such as Google Analytics, App Dynamics and New Relic. These software alternatives, all provide more or less the similar outcome as Dynatrace UEM, and could have been utilised for the same. The selected type of software has been utilised as it provides the dataset with individual end user data for each and every visit. An alternate approach to the data collection would be to harvest data through what is called passive monitoring. This is done by replicating data from certain physical locations in a data centre. In doing so one would also get a good enough overall picture of the traversing traffic. Though this particular method was considered at first, it has the downside that it provides no individual end user data.

The user journey that has been utilised in the assignment has been defined by Norwegian ASA and corresponds to already internally established guidelines. The choice of software has its drawbacks as it is not specialised in facilitating for this specific research, as a contrary to Google Analytics which has it prime focus on collecting front end data. The user journey as such imposes no specific limitations to the research. Rather it gives the possibility for the additional dimension of obtaining and correlating data for each step of the journey. Thus, providing deeper understanding of the customer base. One particular dimension of an end user visit has been omitted from the actual research, namely what resided in the segment of the user experienced time. The Appdex scale somewhat compensates, as it says something about the user experience and thus also implicates the experienced time. Within this segment there are multiple subjects of time which one could include in the overall analysis. These subjects are amongst others the end users experienced time and the presentation dimension of time. Even though this subject has been omitted from the practical research it has been included in to the thesis, both in theory and discussion, due to its importance and addition to the total picture of end user visits.

Since the dataset for the thesis is rather limited due to challenges in the deployment of the solution, an additional source that could confirm the collected dataset would be preferred. Luckily such a source exists for Norwegian AS, namely Google Analytics. Had the additional source not existed, it would be difficult to verify the strength of the data set as such. A more extensive dataset would need to be collected. Furthermore, the data collection and additional data cleaning could contribute to the dataset being erroneous, thus possibly contributing to an equally erroneous outcome. However, when comparing the accumulated data set with numbers from Google Analytics for Norwegian ASA (www.norwegian.no), the outcome is almost the same. The numbers for the sibling dataset from Google Analytics shows that the conversion rate number is 4.82%, as represented in appendix section 10.5.2. Furthermore, the initial dataset before going through any cleaning was comprised of approximately 132 000 visits. The equivalent number from Google Analytics is approximately 131 000 user sessions, as represented in appendix section 10.5.1. The expressions visits and user sessions are mere naming conventions between the different software. The similarities between the different software, implies that the method of measuring the conversion rates, fall and land not far from each other.

7 Conclusion

In this thesis, the problem statement has been to research the relationship between web site speed and conversion rates. Since web site speed is comprised of many interconnected metrics, it has been necessary to investigate these in order to understand and later disregard the less favourable ones. In this process, it has been found that there is a difference between how much time is spent by web servers delivering a request (backend time) and the browser handling said request (frontend time). To explore the dynamics, a data set from the HTTP Archive has been analysed. The analysis shows that there is in fact a large difference between the two domains and that the distribution is to be consider universal. Following that result it has been deemed logical to explore how frontend time contributes to conversion rates. The selected approach includes collecting individual end user data from each step in a six-step user journey. For the selection of metrics to be collected and later analysed, it has been highlighted that there are several different ones that are viable to utilise each one contributing to their part of the picture. Due to restrictions in the utilised software the metric *Load Time* has been utilised. Collection of the data has been limited to software form the Dynatrace software stack, as they are the monitoring tools utilised in Norwegian ASA. From this stack, the software called Dynatrace AppMon User Experience Monitoring was utilised.

As site speed plays only a part of the entire picture as discussion about several aspects of web site speed or performance have been discussed and evaluated. It has been shown that that there are multiple dimensions of time that needs to be addressed in order to give an answer to the effect speed has on conversion. These dimensions, exempt from actual delivery speed, include perceived time and how usability influences the same.

Univariate analysis has been performed to analyse the relationship between conversion and load times. Furthermore, multivariate analysis has been performed to explore the relationship between the relationship between the many variables that comprises and end user visit. This includes the amount of time a user spends on the site, how many interactions the user has with it. Visit data and conversion data has been compared with datasets from Google Analytics for www.noregian.no. The datasets from the two different sources are not far from identical.

Key findings from the dataset in the thesis reveal that there is a clear relationship between the performance of www.norwegian.no. and its conversion rates. The relationship between the two variables plays out in such a way that there is quite a large difference between the average and mean time between converted and non-converted visits. It is found that the chance to convert is increased the longer the end user proceeds into the defined user journey. Furthermore, analysis of the dataset shows that the conversion rate goes down considerably after four (4) seconds.

8 Further work

There are several paths that this type of research could take. As a primary step, it would be recommended to set up a more extensive data collection and for a longer period of time. Doing so would allow for a more nuanced picture of the data set. This would help to pinpoint any eventual biases and/or anomalies. As the notion that web site speed affects conversion rates, has been shown, it would be advised to further explore the effects on more variables as related to speed. The direction of utilising a lager set metrics to be more accurate to how the user is actually perceiving a web site, would give this added dimension to the work.

The different aspects of time have been elaborated, but only on a quite superficial plane. Thus, to gain deeper understanding, there is a need to include a qualitative dimension of the user experience to understand the broader picture. Furthermore, it would be very interesting to combine the quantitative data with qualitative research in order to examine how different aspects of time is influences conversion rates.

As most of the analysis part was manually conducted, there is room for automating this part of the research. There is no reason why machines cannot display performance data in the same manner as they do visit data. Doing so would allow for simultaneously testing of several different models. Trying different logistic models where one tries to pinpoint predictors that affect the outcome of visits, would be valuable. Being able to set value of performance or the lack thereof for several aspects of the web site and the underlying code that comprise it, would also be favourable to investigate further. The reason for that is that it compels for a structure where developers, web site operators and the business are forced to communicate.

9 References

Agresti, A. (2002). Categorical Data Analysis

Akamai. (2006) https://www.akamai.com/uk/en/about/news/press/2006-press/akamai-and-jupiterresearch-identify-4-seconds-as-the-new-threshold-of-acceptability-for-retail-web-page-response-times.jsp

Akamai. (2009) https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp (viewed 22.09.2017)

Appdex. (2007). http://apdex.org/documents/ApdexTechnicalSpecificationV11_000.pdf

Card, S et al. (1991). The information visualizer: An information workspace, available from: http://www2.parc.com/istl/groups/uir/publications/items/UIR-1991-01-Card-CHI91-IV.pdf (viewed 22.07.2016)

Chu, J et al. (2013). Increasing TCP's Initial Window – RFC 6928, available from https://tools.ietf.org/html/rfc6928 (viewed 27.06.2016)

Church, R. M. et al. (1994). Application of Scalar Timing Theory to individual trials, available from Journal of experimental psychology: Animal Behaviour Processes vol. 20, no. 2.

Davis, F et al. (1989). User acceptance of computer technology: A comparison of two theoretical models, available from Management science vol. 35.

Dynatrace. (2017a). https://www.dynatrace.com/support/doc/appmon/appmon-reference/appmon-platform-overview/architecture/

Dynatrace. (2017b). https://www.dynatrace.com/support/doc/appmon/appmon-reference/appmon-platform-overview/purepath-explained/

Dynatrace. (2017c). https://www.dynatrace.com/support/doc/appmon/application-monitoring/usage-of-business-transactions/

Dynatrace. (2017d). (https://community.dynatrace.com/community/pages/viewpage.action?pageId=221381305)

Dynatrace. (2017e). https://www.dynatrace.com/support/doc/appmon/user-experience-management/how-does-uem-work/user-action-timings/

Dynatrace. (2017e).

https://www.dynatrace.com/support/doc/appmon/user-experience-management/how-does-uem-work/w3c-resource-timing-metrics/

e24. (2017) https://e24.no/naeringsliv/dnb/omfattende-nett-troebbel-for-dnb/24144869 (viewed 21.01.2018)

Field, A. (2009). Discovering Statistics Using SPSS

Fielding, R et al (1999). Hypertext Transfer Protocol – HTTP/1.1 https://tools.ietf.org/rfc/rfc2616 (viewed 24.02.2017)

Financial Times. (2016) http://engineroom.ft.com/2016/04/04/a-faster-ft-com/ - (viewed 22.09.2017)

Fitts P. M. & Seeger C. M (1953). S-R Compatibility: Spatial characteristics of stimulus and response codes, available from http://e.guigon.free.fr/rsc/article/FittsSeeger53.pdf (viewed 25.07.2016)

Google (2009). Speed Matters, available from: http://googleresearch.blogspot.no/2009/06/speed-matters.html (viewed 28.04.2016)

Google (2010). Using site speed in web search ranking, available from: https://webmasters.googleblog.com/2010/04/using-site-speed-in-web-search-ranking.html (viewed 28.04.2016)

Google BigQuery. (2017). https://bigquery.cloud.google.com/table/httparchive:runs.2017_03_15_pages (viewed and utilised 03.2017)

Google Developers https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model?hl=en (viewed 05.2017)

Grigorik, I. (2013). *High Performance Browser Networking*.

Habib, A and Abrams, M (2000). Analysis of Sources of Latency in Downloading Web Pages, available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5961&rep=rep1&type=pdf (viewed 04.04.2016)

Hick, W. E. (1952). On the rate of gain of information. Quarterly Journal of Experimental Psychology 4:1, 11-26.

HTTP Archive. (2017a). http://httparchive.org/trends.php

HTTP Archive. (2017b). http://httparchive.org/urls.php - (viewed 03.2017)

HTTP Archive. (2017c). http://httparchive.org - (viewed 03.2017)

HTTP Archive. (2017d). http://httparchive.org/trends.php?s=Top100&minlabel=Mar+15+2011&maxlabel=Dec+15+2016#bytesHtml&reqHtml (viewed and utilised 03.2017)

HTTP Archive. (2017e). http://httparchive.org/about.php#datagathered (viewed 07.2017)

Hällström, Å (1985). The time-order error and its relatives: mirrors of cognitive processes in comparing. Psychological Bulletin vol. 97.

IANA, Internet Assigned Numbers Authority. http://data.iana.org/TLD/tlds-alpha-by-domain.txt (viewed 07.2017)

Information Sciences Institute, University of Southern California (ISI) (1981). Transmission Control Protocol – RFC 793, available from: https://tools.ietf.org/html/rfc793 (viewed 26.06.2016)

J. D. Brutlag, Hutchinson, H and Stone, M (2008). User Preference and Search Engine Latency.

Jupiter research (2006). Retail web site performance. Consumer reaction to a poor online shopping experience.
Kanai, R et al. (2006). Time dilation in dynamic visual display. Journal of vision vol. 6.

Khan, F (2015). The cost of Latency, available from: http://www.telx.com/blog/the-cost-of-latency/ (viewed 28.04.2016)

KPMG. (2017) https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2017/01/the-truth-about-online-consumers.pdf (viewed 21.01.2018)

Kurose, J. F. & Ross, K. W. (2013). Computer Networking, A Top-Down Approach. Pearson.

Kutner, H, M et al. (2005). Applied Linear Statistical Models

Liddle, J (2007). Amazon found every 100ms of latency cost them 1% in sales, available from: http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/ (viewed 28.04.2016)

Loftus, E. F. & Palmer, J. C. (1974). Reconstruction of automobile destruction: An example of the interaction between language and memory. Journal of Verbal Learning and Verbal Behaviour 13.

National Telecommunications & Information Administration (NTIA), Institute for Telecommunication Sciences (ITS), available from: http://www.its.bldrdoc.gov/fs-1037/dir-031/_4641.htm (viewed 11.05.2016)

NCC Group. (2015) https://www.nccgroup.trust/globalassets/resources/uk/case-studies/web-performance/website-performance--seatwave---case-study-0615.pdf (viewed 21.01.2018)

Nielsen Jacob (1998) Nielsen's Law of Internet bandwidth available from: https://www.nngroup.com/articles/law-of-bandwidth/ (viewed 15.05.2016)

O'Reilly (2009) http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html (viewed 21.01.2018)

Peterson, L. L. & Davie, B. S. (2012). *Computer Networks: A Systems Approach 3rd edition*

Reuters. (2013) https://www.reuters.com/article/net-us-amazon-website/amazon-com-back-up-after-going-down-for-u-s-canadian-users-idUSBRE97I0UT20130819 (viewed 21.01.18)

Sadre, R., & Haverkort, B. R. (2008). Changes in the web from 2000 to 2007.

Seow, S. C. (2005). Information Theoretic Models of HCI: A Comparison of the Hick-Hyman Law and Fitts' Law. Human-Computer Interaction, vol. 20.

Seow, S. C. (2008). Designing and Engineering Time: The Psychology of Time Perception in Software

Shniederman, B (1984). Response time and display rate in human performance with computers, available from Computing surveys vol. 16, no. 3.

Sidney L. S. & Jane N. M. (1986). Guidelines for designing user interface software EDS-TR-86-278, available from: http://www.dfki.de/~jameson/hcida/papers/smith-mosier.pdf (viewed 25.07.2016)

Souders, Steve. (2007). High Performance Web Sites

Sternberg, Robert and Karin (2012). Cognitive Psychology. Science.

Tanenbaum, A. S. (1996). *Computer Networks*. *World Wide Web Internet and Web Information Systems* (Vol. 52).

The free dictionary
http://www.thefreedictionary.com/latency (viewed 27.05.2016)

The Guardian. (2016) https://www.theguardian.com/technology/2015/jan/27/is-facebook-down-outages (viewed 21.01.2018)

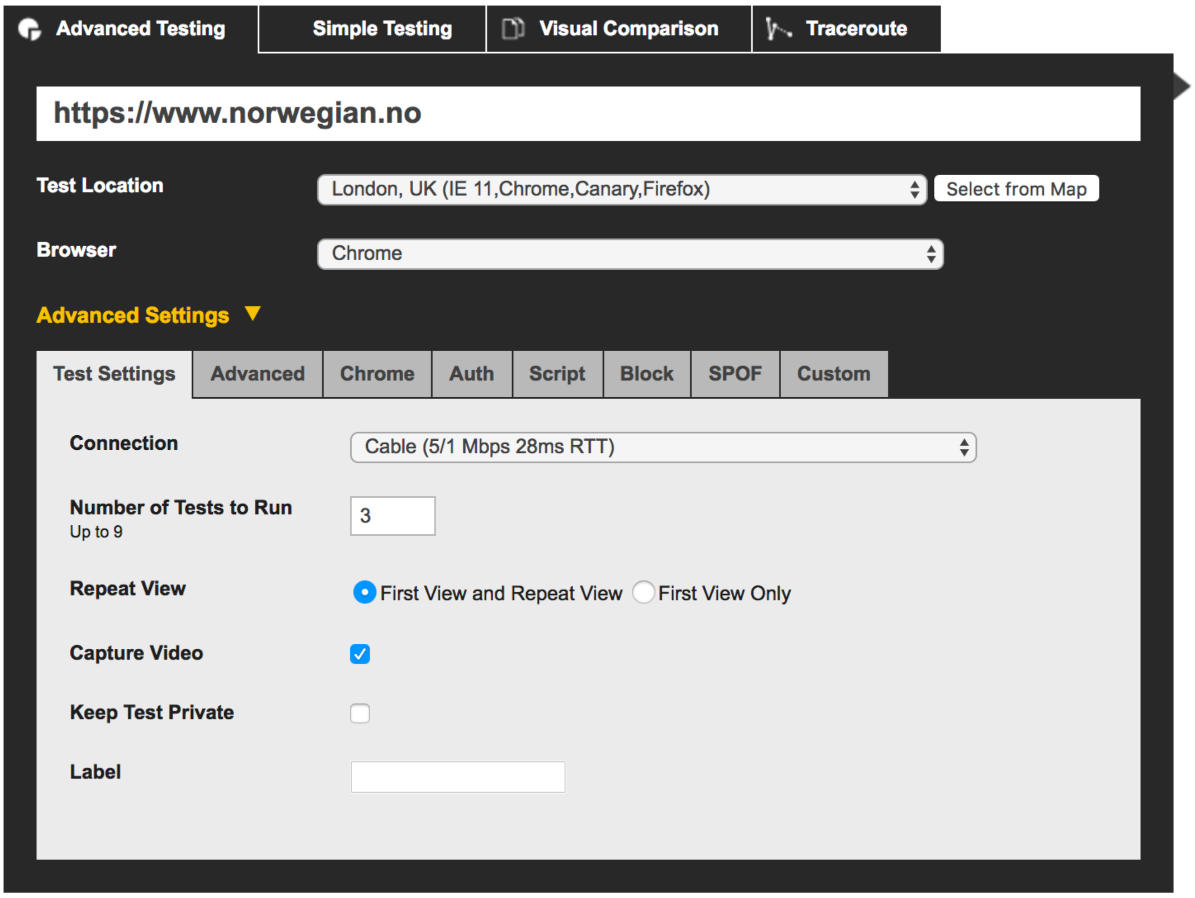
UCalgary - University of Calgary. (2014).
http://wiki.ucalgary.ca/page/Courses/Computer_Science/CPSC_441.W2014/Chapter_1:_Computer_Networks_and_The_Internet (Viewed 03.2017)

Webpagetest. (2017) https://www.webpagetest.org/ (viewed 05.2017)

Wonder network, Global ping statistics available from:
https://wondernetwork.com/pings/Oslo/London (viewed 27.06.2016)

10 Appendix

10.1 WebPageTest settings



https://www.norwegian.no

Test Location	London, UK (IE 11,Chrome,Canary,Firefox)
Browser	Chrome
Connection	Cable (5/1 Mbps 28ms RTT)
Number of Tests to Run Up to 9	3
Repeat View	[✓ First View and Repeat View] First View Only
Capture Video	✓
Keep Test Private	☐
Label	

10.2 Navigation timings

These navigation timing metrics were copied form https://www.w3.org/TR/navigation-timing/.

navigationStart	This attribute must return the time immediately after the user agent finishes prompting to unload the previous document. If there is no previous document, this attribute must return the same value as fetchStart.
unLoadEventStart	If the previous document and the current document have the same origin [IETF RFC 6454], this attribute must return the time immediately before the user agent starts the unload event of the previous document. If there is no previous document or the previous document has a different origin than the current document, this attribute must return zero.
unLoadEventEnd	If the previous document and the current document have the same same origin, this attribute must return the time immediately after the user agent finishes the unloadevent of the previous document. If there is no previous document or the previous document has a different origin than the current document or the unload is not yet completed, this attribute must return zero. If there are HTTP redirects or equivalent when navigating and not all the redirects or equivalent are from the same origin, both unloadEventStart and unloadEventEnd must return the zero
redirectStart	If there are HTTP redirects or equivalent when navigating and if all the redirects or equivalent are from the same origin, this attribute must return the starting time of the fetch that initiates the redirect. Otherwise, this attribute must return zero.
redirectEnd	If there are HTTP redirects or equivalent when navigating and all redirects and equivalents are from the same origin, this attribute must return the time immediately after receiving the last byte of the response of the last redirect. Otherwise, this attribute must return zero.
fetchStart	If the new resource is to be fetched using HTTP GET or equivalent, fetchStart must return the time immediately before the user agent starts checking any relevant application caches. Otherwise, it must return the time when the user agent starts fetching the resource.
domainLookupStart	This attribute must return the time immediately before the user agent starts the domain name lookup for the current document. If a persistent connection [RFC 2616] is used or the current document is retrieved from relevant application caches or local resources, this attribute must return the same value as fetchStart.
domainLookupend	This attribute must return the time immediately after the user agent finishes the domain name lookup for the current document. If a persistent connection [RFC 2616] is used or the current document is retrieved from relevant application caches or local resources, this attribute must return the same value as fetchStart.
connectStart	This attribute must return the time immediately before the user agent start establishing the connection to the server to retrieve the document. If a persistent connection [RFC 2616] is used or the current document is retrieved from relevant application caches or local resources, this attribute must return value of domainLookupEnd.
connectEnd	This attribute must return the time immediately after the user agent finishes establishing the connection to the server to retrieve the current document. If a persistent connection [RFC 2616] is used or the current document is retrieved from relevant application caches or local resources, this attribute must return the value of domainLookupEnd If the transport connection fails and the user agent reopens a connection, connectStart and connectEnd should return the corresponding values of the new connection. connectEnd must include the time interval to establish the transport connection as well as other time interval such as SSL handshake and SOCKS authentication.
secureConnectionStart	This attribute is optional. User agents that don't have this attribute available must set it as undefined. When this attribute is available, if the scheme of the current page

	is HTTPS, this attribute must return the time immediately before the user agent starts the handshake process to secure the current connection. If this attribute is available but HTTPS is not used, this attribute must return zero.
requestStart	This attribute must return the time immediately before the user agent starts requesting the current document from the server, or from relevant application caches or from local resources. If the transport connection fails after a request is sent and the user agent reopens a connection and resend the request, requestStart should return the corresponding values of the new request.
responseStart	This attribute must return the time immediately after the user agent receives the first byte of the response from the server, or from relevant application caches or from local resources.
responseEnd	This attribute must return the time immediately after the user agent receives the last byte of the current document or immediately before the transport connection is closed, whichever comes first. The document here can be received either from the server, relevant application caches or from local resources.
domLoading	This attribute must return the time immediately before the user agent sets the current document readiness to "loading".
domInteractive	This attribute must return the time immediately before the user agent sets the current document readiness to "interactive".
domContentLoadedEventStart	This attribute must return the time immediately before the user agent fires the DOMContentLoaded event at the Document.
domContentLoadedEventEnd	This attribute must return the time immediately after the document's DOMContentLoaded event completes.
domComplete	This attribute must return the time immediately before the user agent sets the current document readiness to "complete". If the current document readiness changes to the same state multiple times, domLoading, domInteractive, domContentLoadedEventStart, domContentLoadedEventEnd and domComplete must return the time of the first occurrence of the corresponding document readiness change.
loadEventStart	This attribute must return the time immediately before the load event of the current document is fired. It must return zero when the load event is not fired yet.
loadEventEnd	This attribute must return the time when the load event of the current document is completed. It must return zero when the load event is not fired or is not completed.

10.3 HTTP Archive variables

Variable name	Variable type
pageid	INTEGER
createDate	INTEGER
archive	STRING
label	STRING
crawlid	INTEGER
wptid	STRING
wptrun	INTEGER
url	STRING
urlShort	STRING
urlhash	INTEGER
cdn	STRING
startedDateTime	INTEGER
TTFB	INTEGER
renderStart	INTEGER
onContentLoaded	INTEGER
onLoad	INTEGER
fullyLoaded	INTEGER
visualComplete	INTEGER
PageSpeed	INTEGER
SpeedIndex	INTEGER
rank	INTEGER
reqTotal	INTEGER
reqHtml	INTEGER
reqJS	INTEGER
reqCSS	INTEGER
reqImg	INTEGER
reqGif	INTEGER
reqJpg	INTEGER
reqPng	INTEGER
reqFont	INTEGER
reqFlash	INTEGER
reqJson	INTEGER
reqOther	INTEGER
bytesTotal	INTEGER
bytesHtml	INTEGER
bytesJS	INTEGER
bytesCSS	INTEGER
bytesImg	INTEGER
bytesGif	INTEGER
bytesJpg	INTEGER
bytesPng	INTEGER
bytesFont	INTEGER
bytesFlash	INTEGER
bytesJson	INTEGER
bytesOther	INTEGER
bytesHtmlDoc	INTEGER
numDomains	INTEGER
maxDomainReqs	INTEGER
numRedirects	INTEGER
numErrors	INTEGER
numGlibs	INTEGER
numHttps	INTEGER
numCompressed	INTEGER
numDomElements	INTEGER
maxageNull	INTEGER
maxage0	INTEGER
maxage1	INTEGER
maxage30	INTEGER
maxage365	INTEGER
maxageMore	INTEGER
gzipTotal	INTEGER
gzipSavings	INTEGER
_connections	INTEGER
_adult_site	BOOLEAN
avg_dom_depth	INTEGER
document_height	INTEGER
document_width	INTEGER
localstorage_size	INTEGER
sessionstorage_size	INTEGER
num_iframes	INTEGER
num_scripts	INTEGER
doctype	STRING
meta_viewport	STRING
reqAudio	INTEGER
reqVideo	INTEGER
reqText	INTEGER
reqXml	INTEGER
reqWebp	INTEGER
reqSvg	INTEGER
bytesAudio	INTEGER
bytesVideo	INTEGER
bytesText	INTEGER
bytesXml	INTEGER
bytesWebp	INTEGER
bytesSvg	INTEGER
num_scripts_async	INTEGER
num_scripts_sync	INTEGER
usertiming	INTEGER

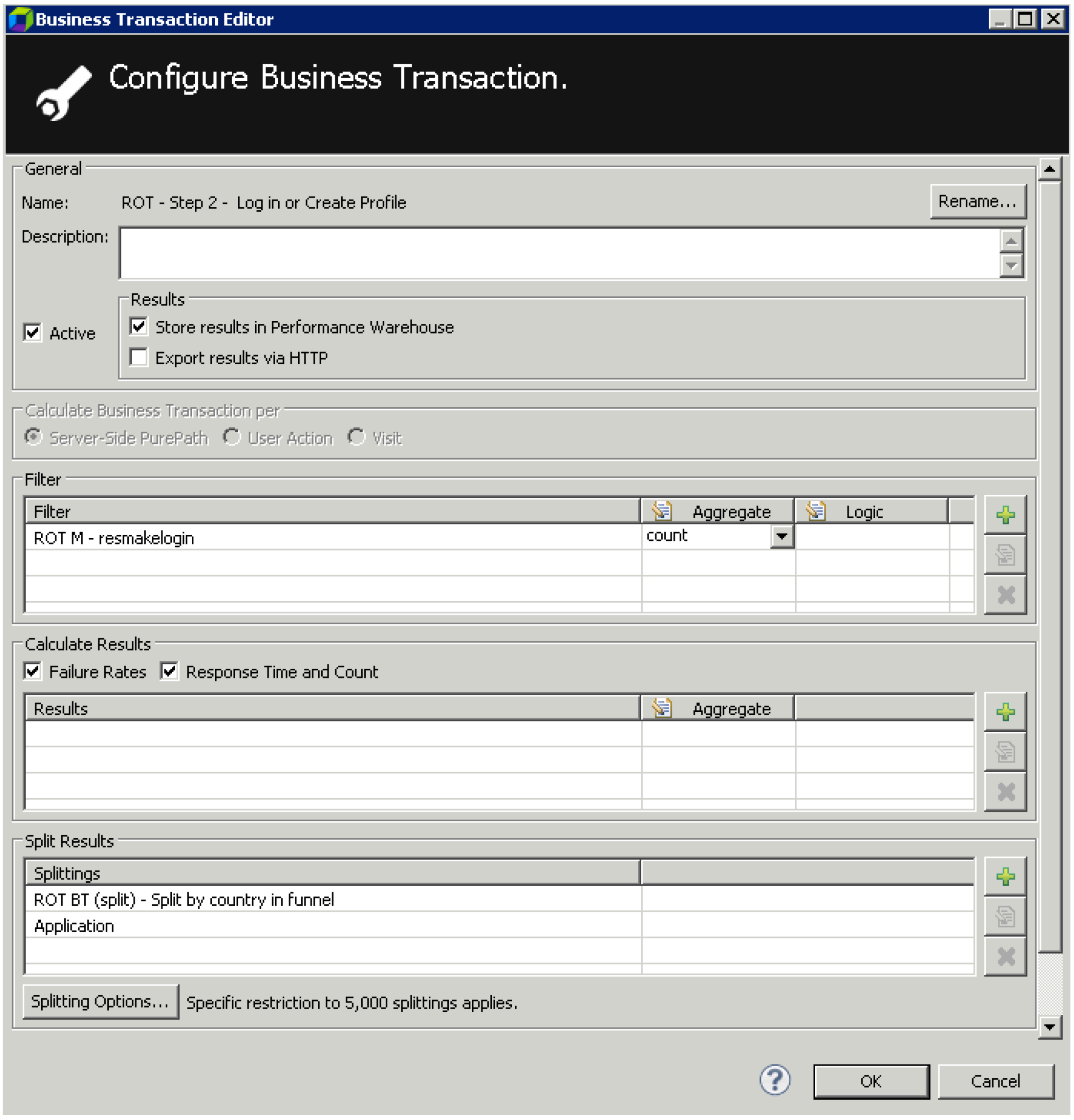
10.3.1 Query I

SQL query

```
SELECT
        url,
        TTFB,
        renderStart,
        onLoad,
        fullyLoaded
FROM
        httparchive:runs.2017_03_15_pages
```

10.4 Business transactions

Step 1



General

Name: ROT - Step 2 - Log in or Create Profile

Rename...

Description:

[✓] Active

Results

[✓] Store results in Performance Warehouse

[ ] Export results via HTTP

Calculate Business Transaction per

[● Server-Side PurePath] ○ User Action ○ Visit

Filter

Filter	Aggregate	Logic
ROT M - resmakelogin	count	

Calculate Results

[✓] Failure Rates [✓] Response Time and Count

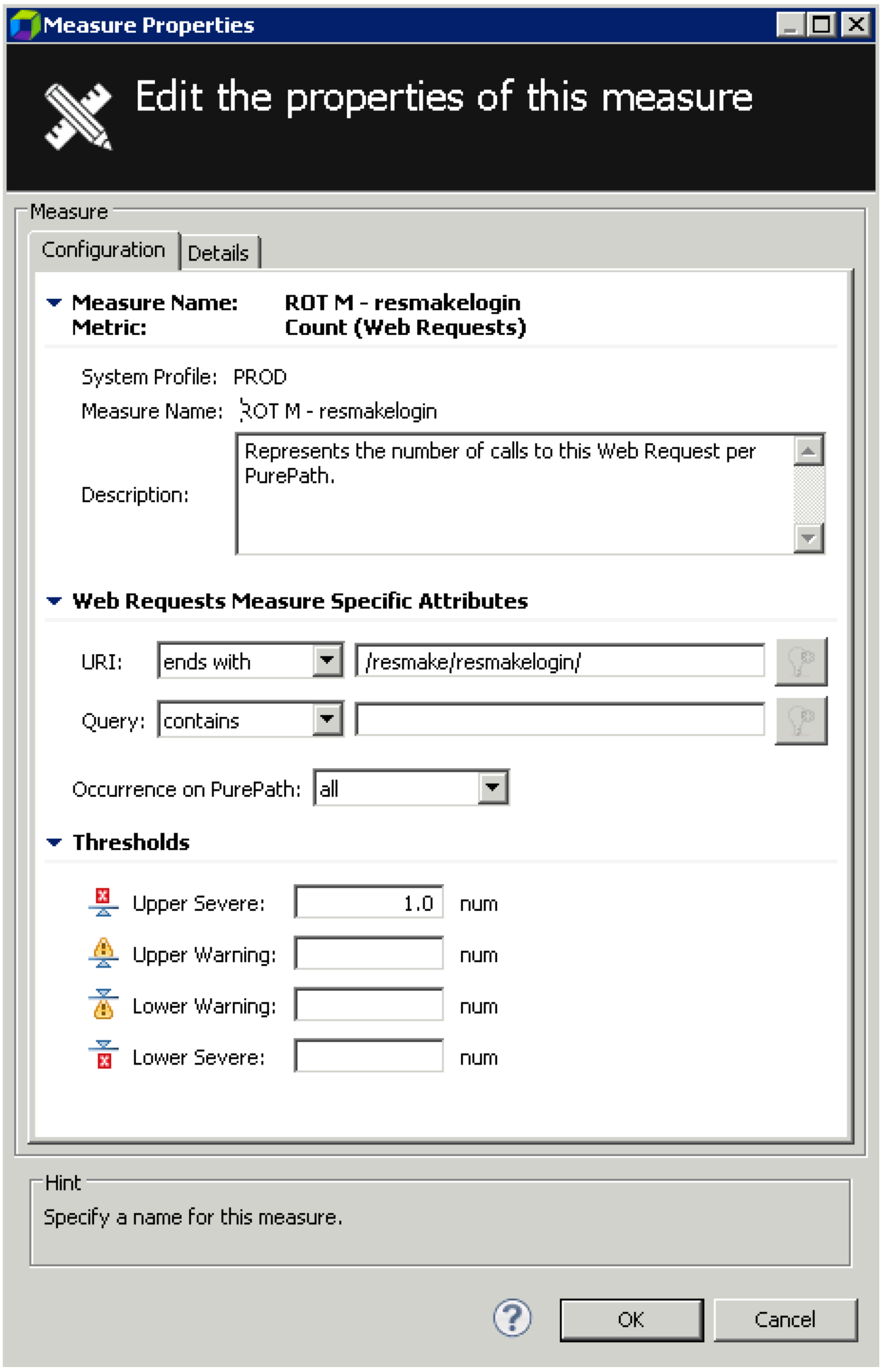
Results	Aggregate	

Split Results

Splittings	
ROT BT (split) - Split by country in funnel	
Application	

Splitting Options... Specific restriction to 5,000 splittings applies.

OK Cancel


Measure Properties

Edit the properties of this measure

Measure

Configuration | Details

Measure Name: ROT M - resmakelogin
Metric: Count (Web Requests)

System Profile: PROD
Measure Name: ROT M - resmakelogin
Description: Represents the number of calls to this Web Request per PurePath.

Web Requests Measure Specific Attributes

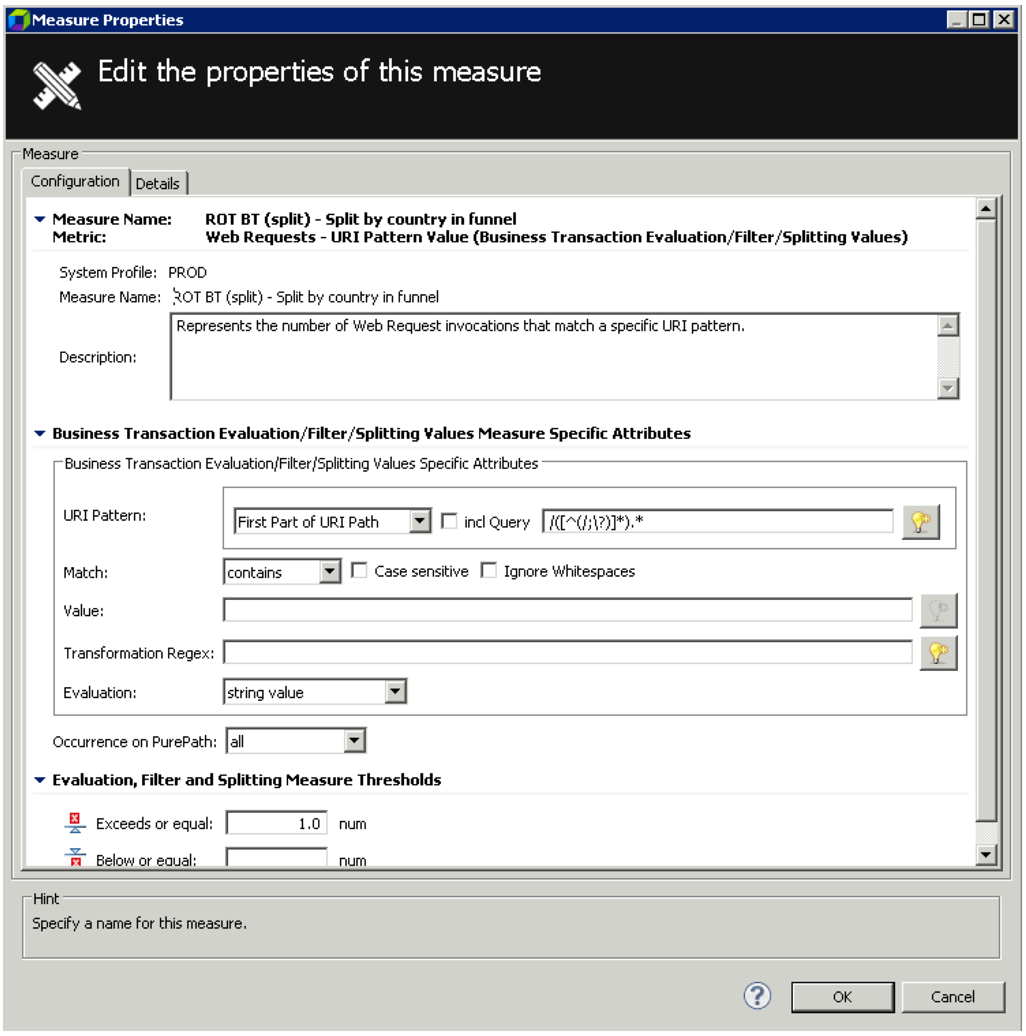
URI: ends with /resmake/resmakelogin/
Query: contains
Occurrence on PurePath: all

Thresholds

Upper Severe:	1.0	num
Upper Warning:		num
Lower Warning:		num
Lower Severe:		num

Hint
Specify a name for this measure.

OK Cancel



Measure Properties

Edit the properties of this measure

Measure

Configuration | Details

Measure Name: **ROT BT (split) - Split by country in funnel**
Metric: **Web Requests - URI Pattern Value (Business Transaction Evaluation/Filter/Splitting Values)**

System Profile: PROD
Measure Name: ROT BT (split) - Split by country in funnel
Description: Represents the number of Web Request invocations that match a specific URI pattern.

Business Transaction Evaluation/Filter/Splitting Values Measure Specific Attributes

Business Transaction Evaluation/Filter/Splitting Values Specific Attributes

URI Pattern: First Part of URI Path ☐ incl Query /([^(/;\?)]*).*
Match: contains ☐ Case sensitive ☐ Ignore Whitespaces
Value:
Transformation Regex:
Evaluation: string value

Occurrence on PurePath: all

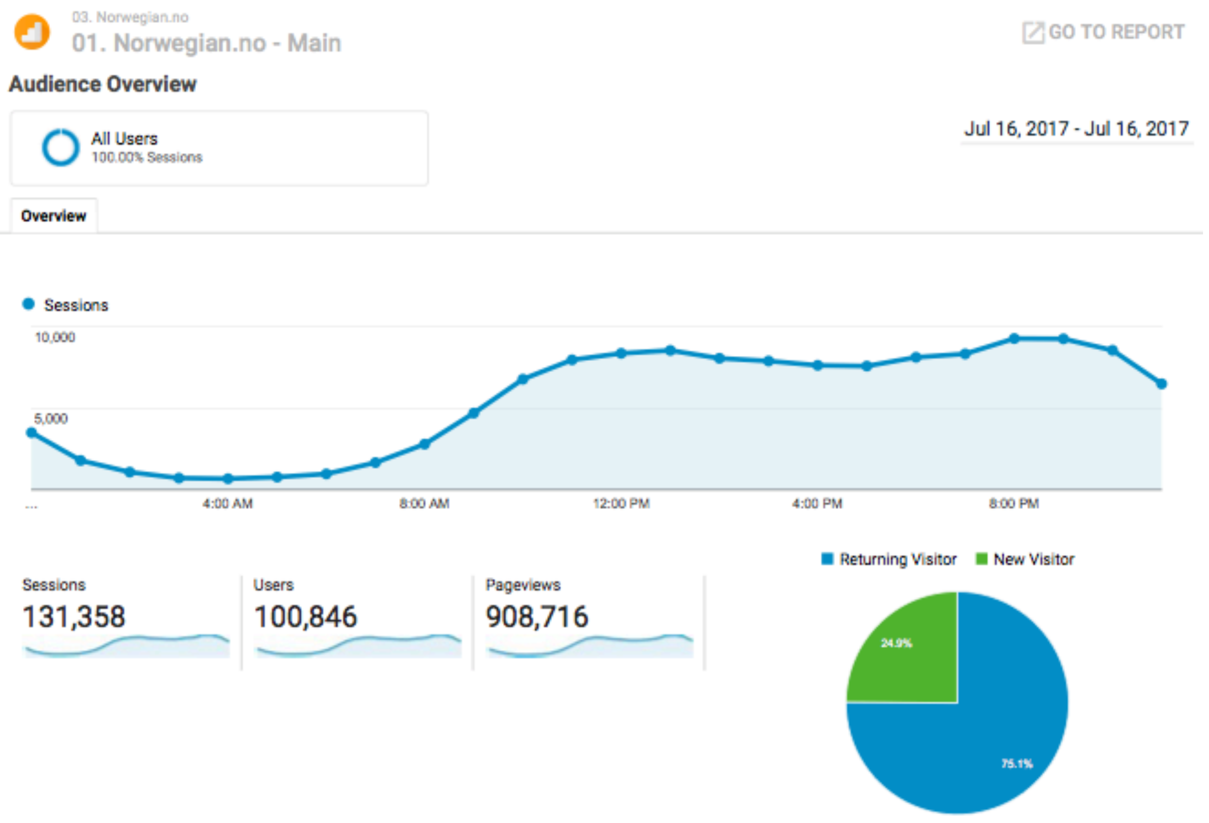
Evaluation, Filter and Splitting Measure Thresholds

Exceeds or equal: 1.0 num
Below or equal: num

Hint
Specify a name for this measure.

10.5 Google Analytics Export

10.5.1 Overview of user sessions for July 16th, 2017



03. Norwegian.no

01. Norwegian.no - Main

GO TO REPORT

Audience Overview

All Users
100.00% Sessions

Jul 16, 2017 - Jul 16, 2017

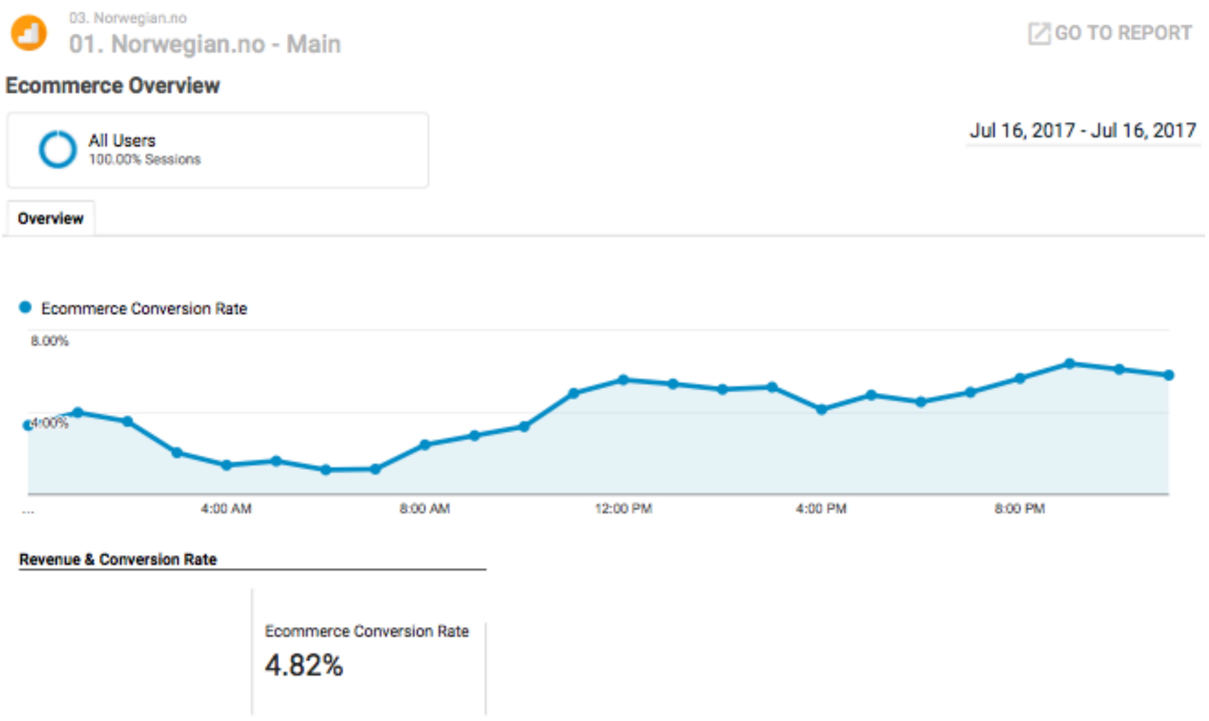
Overview

Sessions
131,358

Users
100,846

Pageviews
908,716

10.5.2 Overview of conversion rate July 16th, 2017


03. Norwegian.no

01. Norwegian.no - Main

GO TO REPORT

Ecommerce Overview

All Users
100.00% Sessions

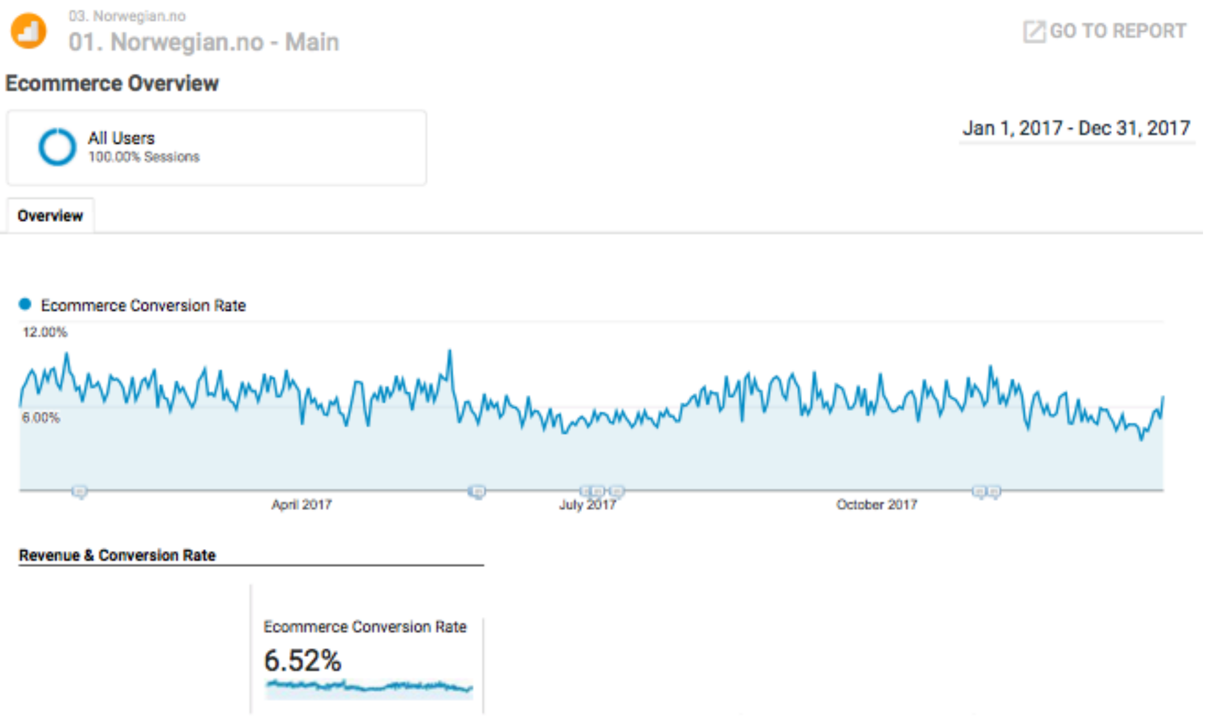
Jul 16, 2017 - Jul 16, 2017

Overview

Revenue & Conversion Rate

Ecommerce Conversion Rate
4.82%

10.5.3 Overview of conversion rate for www.norwegian.no, 2017



03. Norwegian.no

01. Norwegian.no - Main

GO TO REPORT

Ecommerce Overview

All Users
100.00% Sessions

Jan 1, 2017 - Dec 31, 2017

Overview

Revenue & Conversion Rate

Ecommerce Conversion Rate
6.52%

10.6 File structure

Root

- **Part 1**
  - FE vs BE results.xlsx
  - Master Part1.Rproj
  - Part1.analysis.R
  - **Functions**
    - Multiplot.R
- **Part2**
  - Master Thesis.Rproj
  - dataNorway.Rdata
  - 00 Load data.R
  - 01 T-test.R
  - 02 Logistic regression.R
  - **Data**
    - **Cleaned**
      - Step 1 - Visits.xlsx
      - Step 2 - Log in or Create Profile Visits.xlsx
      - Step 3 - Passenger Visits.xlsx
      - Step 4 - Seat Reservation Visits.xlsx
      - Step 5 - Additional Products Visits.xlsx
      - Step 6 - Confirm Reservation Visits.xlsx
    - **Original**
      - Step 1 - Visits.csv
      - Step 2 - Log in or Create Profile Visits.csv
      - Step 3 - Passenger Visits.csv
      - Step 4 - Seat Reservation Visits.csv
      - Step 5 - Additional Products Visits.csv
      - Step 6 - Confirm Reservation Visits.csv
    - **Removed**
      - 5s-visits.xlsx
      - Lprt-gte 20000 ms.xlsx
      - Step2-6.xlsx
      - Transform time to secs.xlsx
  - **Excel**
    - Calculations.xlsx
    - Frequency distribution conversion.xlsx