Printfarm Management System

- Communication Technology 10^{th} semester -

Master Thesis

Alexander Aagren

Aalborg University
Communication Technology
Networks and Distributed Systems

This report is written in LaTeX and compiled in Overleaf. Figures are made using MATLAB R2021b, Diagrams.net 18.1.2 and Tikz. MATLAB R2021b has been used for the majority of the calculations performed.

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Printfarm Management System

Theme:
Master Thesis

Project Period:
Spring Semester 2023

Project Group:
CT10-1023

Participant(s):
Alexander Aagren

Supervisor(s):
Henrik Schiøler

Copies: 1

Page Numbers: 48

Date of Completion:
June 1, 2023

Abstract:

This thesis is written in collaboration with the 3D printing company *Create it REAL*. The challenges of managing a 3D printer farm are investigated, both from the hardware perspective as well as the network connectivity perspective. It was found that the print jobs sent over WiFi to a printer, using Hardware and Firmware made by Create it REAL, use AES128 encryption. A Printfarm Management System is designed to give the capability of queuing multiple print jobs to printers. The Printfarm Management System is designed in a way where it considers the printing time of a print job in order to know whether or not the print job can finish within working hours. It also takes the estimated amount of filament left on the printer into consideration in order to know if the administrator is to be notified about a needed filament change. A set of queuing algorithms is investigated and simulated in order to choose the one best suited for the system. The chosen queuing algorithm was Modified Maximum Urgency First, as it gives the possibility of prioritising specific print jobs, which can be used for error handling of failed print jobs.

Contents

Preface **vii**

Nomenclature **ix**

1 Introduction **1**

2 Problem Analysis **3**
2.1 Analysing the printing process . . . 3
2.2 Error handling . . . 6
2.3 Controlling the printers over the network . . . 7
2.4 Network discovery of 3D printers . . . 9
2.5 Transmitting a print job . . . 10
2.5.1 Security . . . 10
2.6 Queueing Algorithms . . . 12
2.6.1 First-In-First-Out . . . 12
2.6.2 Round Robin . . . 12
2.6.3 Shortest Job First . . . 13
2.6.4 Priority Queuing . . . 14
2.6.5 Earliest Deadline First . . . 15
2.6.6 Least Laxity First Scheduling Algorithm . . . 16
2.6.7 Maximum Urgency First . . . 18
2.6.8 Modified Maximum Urgency First . . . 19
2.7 Problem Formulation . . . 19

3 Requirements **21**

4 System Design **23**
4.1 The Combination of Queuing Algorithms . . . 23
4.2 Running out of filament . . . 25

5 Simulations **29**
5.1 Simulation Results . . . 31
5.1.1 Ranking the algorithms . . . 32
5.1.2 Simulations with random deadlines for print jobs with priority 2 . . . 33

6 Final Design **37**

7 Discussion **41**

8 Conclusion **43**

Bibliography **47**

Preface

This master thesis is written by a 10^{th} semester student, studying Communication Technology - Networks and Distributed Systems by the name of Alexander Aagren. It is expected that the reader has a general knowledge of communication systems.

Source references in this report appear according to the numerical source-referencing method. Sources are referenced by numerical order in which they appear. Figures, tables, and equations are inserted and numerated according to chapter and order of insertion. Explanatory text and references are inserted immediately before or after the figure, table or equation. Abbreviations or acronyms, that are not commonly recognised, are written in full followed by the abbreviation or acronym in parenthesis. After the first reference, the words are only referenced by the acronym or abbreviation.

Aalborg University, June 1, 2023

Alexander Aagren
<aaagre18@student.aau.dk>

Nomenclature

This page contains common abbreviations and terms used throughout this report.

Abbreviation	*Meaning*
ABS	Acrylonitrile Butadiene Styrene
AES	Advance Encryption Standard
DNS-SD	DNS Service Discovery
EDF	Earliest Deadline First
FCFS	First-Come-First-Serve
FDM	Fused Deposition Modelling
FIFO	First-In-First-Out
IoT	Internet of Things
LLF	Least Laxity First
mDNS	multicast Domain Name System
MLLF	Modified Least Laxity First
MMUF	Modified Maximum Urgency First
MUF	Maximum Urgency First
PLA	PolyLactic Acid
RM	Rate Monotonic
SJF	Shortest Job First
TPU	Thermoplastic Polyurethane

1 Introduction

There are many forms of 3D printing. One of the widely used 3D printing techniques is material extrusion-based 3D printing. This technique is widely used due to its low prices and the fact that the products can be fully functional parts.[1] Fused Deposition Modelling (FDM) is an example of material extrusion-based 3D printing. It was developed in the early 1990 and was the first example of material extrusion-based 3D printing.[1]

A common procedure for 3D printing is to import the 3D model into a program on the computer called a slicer. In the slicer, a user can then orient a model, add support for a model and more. The slicer is also the program used to set the settings for a printer, and when a user is done, the user can export the print file. This print file is often given in G-code format for FDM printers.
FDM works by having a filament that is heated up to a semi-liquid state and extruded on a print plate. If support is needed for the model, the 3D printer will extrude material for the support in a way that results in a weaker end result for the support in order to make the support possible to remove from the print.

As FDM can be used for fully functional parts and is relatively low cost, it has been looked into having multiple printers, also called a printer farm, in order to have the finished product faster, especially for a multipart product.
Another reason for having multiple printers can be to make a factory, where the amount of printers directly correlates with the number of finished parts that can then be shipped. Furthermore, using a 3D printer farm for manufacturing parts gives versatility in the sense of switching from one part to another is as simple as slicing the new part, adding the G-code to the storage drive, and starting the new print on the printer. If a part is to be printed in another material, then the material needs to be changed before starting a new print. However, this process is considerably faster than if the part for example was manufactured by the use of moulds.
These are just two examples of applications of a printer farm.

When talking about print farms, there is a challenge that the printers are made for manual work, which means that the majority of the printers require the user to save the g-code to a flash drive and then plug this into the printer before starting the print. When the print is complete, the user then has to remove the print from the print plate. When this is done, the user can then start the procedure again with the next part. Furthermore, the printed result might need post-processing, such as the removal of the supports.
As the technology improves, so has 3D printing. It is no longer only hard materials such as Poly-Lactic Acid (PLA) and Acrylonitrile Butadiene Styrene (ABS) that are being used as filament, but also softer materials such as Thermoplastic Polyurethane (TPU).

One of many companies that has 3D printers that can print with TPU is *Create it REAL*. Create it REAL is making the electronics, the firmware and the software for the 3D printers. Some of the electronics they make are their motherboards for 3D printers called Bluefin. [2] When looking into the software they make, they have an online slicer called REALvision Online, and they have an offline version called REALvision Pro.[3] Some of Create it REAL's customers are orthopaedic doctors making shoe insoles. These insoles are 3D printed and can be handed to customers in a relatively short time. A challenge that these orthopaedic doctors experience is recruiting manual labour to start the printers and removing the finished print. For that reason, an automatic print farm man-

agement system would be advantageous.

Thereby, an initial problem formulation can be constructed as:

> *How can a 3D printing farm be automated to reduce the amount of manual labour needed?*

2 Problem Analysis

This master thesis is written in collaboration with Create it REAL, who have already begun looking into the problem stated in Chapter 1. In order to automate a 3D printing farm it is important to understand which steps are usually present. From the common procedure mentioned in Chapter 1, a set of steps can be defined:

- Slice the model
- Transfer the G-code to the printer
- Start the print
- Remove the print when finished
- Verify that the print plate is empty
- Start a new print
- Post-process the finished print

The next section will be used for analysing the different steps and their possibilities for automation.

2.1 Analysing the printing process

When looking into 3D printers it is important to remember that not all printers are as advanced as the ones made by Create it REAL. If looking into the most used printer model in a Create it REAL slicer product, a printer by the name of *Ender 3*, made by the company *Creality*, runs with the title. One notable thing about the Ender 3 is that it does not have the capability of having WiFi or Ethernet.

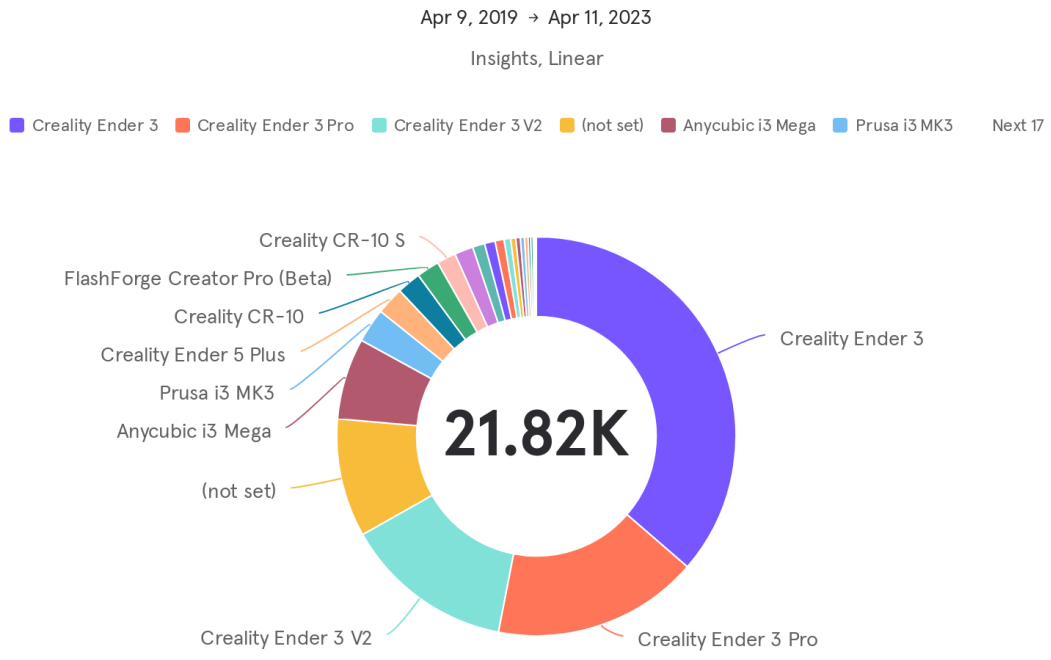


Figure 2.1: Diagram of the used printers in a Create it REAL slicer product from April 9, 2019, to April 11 2023.

Printer Model	Amount of users	Percentage of total
Creality Ender 3	7931	36.34%
Creality Ender 3 Pro	3647	16.71%
Creality Ender 3 V2	3001	13.75%
undefined	2096	9.60%
Anycubic i3 Mega	1438	6.59%
Prusa i3 MK3	602	2.76%
Creality Ender 5 Plus	492	2.25%
Creality CR-10	418	1.92%
FlashForge Creator Pro (Beta)	406	1.86%
Creality CR-10 S	337	1.54%
Ultimaker 2+	329	1.51%
Monoprice Select Mini V2 (Beta)	216	0.99%
IdeaWerk-Speed	186	0.85%
Monoprice Voxel (Beta)	159	0.73%
Monoprice Mini Delta (Beta)	114	0.52%
Monoprice Select Mini (Beta)	93	0.43%
Creality CR-20 Pro	80	0.37%
Ultimaker 2 Extended+	73	0.33%
Creality CR-10 Mini (Beta)	61	0.28%
Speedy	47	0.22%
TEVO Tornado (Beta)	42	0.19%
TEVO Tarantula Pro	34	0.16%
Dood S	21	0.10%

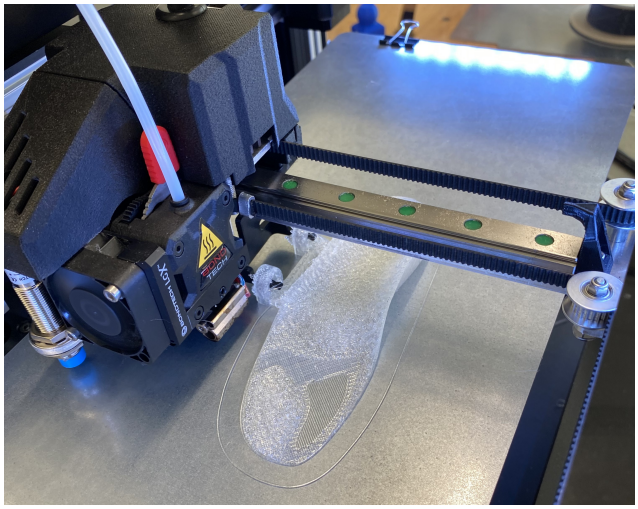
Table 2.1: A Create it REAL slicer product statistics from April 9, 2019, to April 11 2023.

It is unknown why some printers have been defined as "undefined". But if these were all added to the second most used printer, the Anycubic i3 Mega, the most used printer would still be the Creality Ender 3. The Creality Ender 3 and its different versions do not have the ability for WiFi or Ethernet without upgrading the hardware. [4]It is possible to make modifications to 3D printers, giving them the capability of network connectivity, however, this requires modifying the firmware and is not a simple plug-and-play solution in most cases. One solution for adding network capabilities to the printers was investigated in [5]. Their solution was to equip each printer with a Raspberry Pi 3, which has WiFi capabilities. This was chosen in order to have an *OctoPrint* instance for each of the printers.[5]
OctoPrint will be investigated further in Section 2.3.

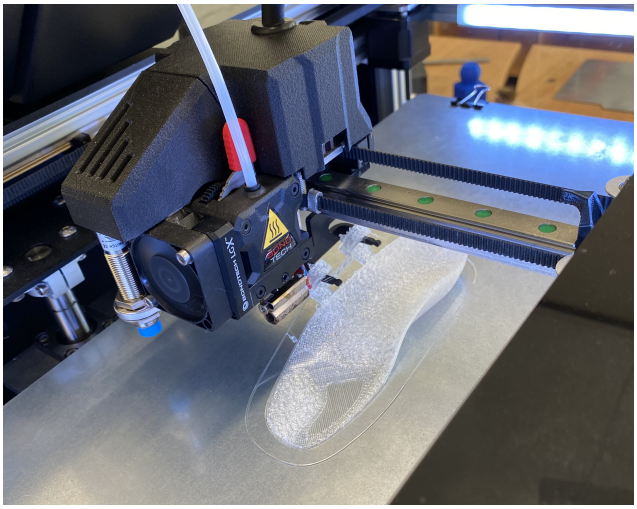
This is where Create it REAL stands out on the market as their motherboards already have WiFi enabled [2], giving the advantage of being able to send a G-code to the printer without getting up from a chair. Thereby, the process of transferring the G-code to the printer has been simplified, but not automated. At the time of writing, if all the printers are occupied, there is no system in place to handle the queue. The manual initiation of print jobs imposes a restriction on the maximum number of print jobs that can be executed overnight, as the absence of employees during that time precludes the commencement of print operations.
Taking this into consideration, one thing that could help to automate the printing process could be to design a queue manager. If a queue manager was designed and implemented, this could possibly help in ensuring that there is at least one print job per printer that is printing after the employees

have left the premises. However, when this print job has been completed, the finished print would have to be removed from the print plate, before the next print could be started.

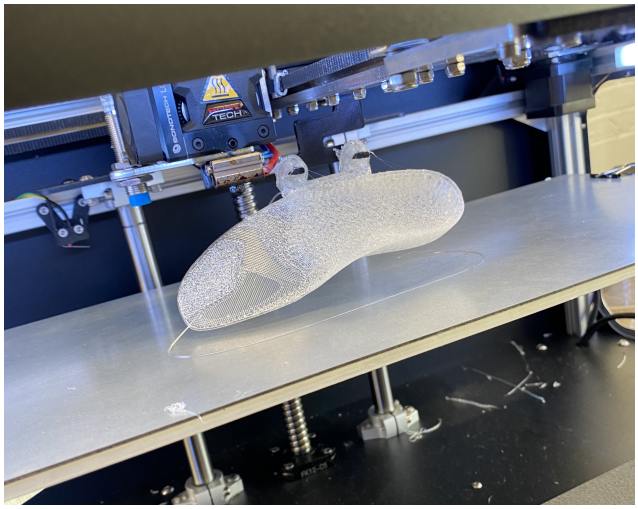
The process of removing a finished print from the print plate can be done in multiple ways. The manual process commonly consists of a person going to the printer and separating the print job from the print plate.
Create it REAL has invented its own procedure for removing the finished prints from the print plate. This is done by having a "fork" next to the print head, which after finishing printing, grabs the print, and lifts it up to ensure that the print detaches from the print plate. Then it moves the print to the edge of the print plate and lowers the finished print while moving the print head away from the finished print, making it fall off. A demo of this process can be seen in the video in [6]. Another demo can be seen in Figure 2.2



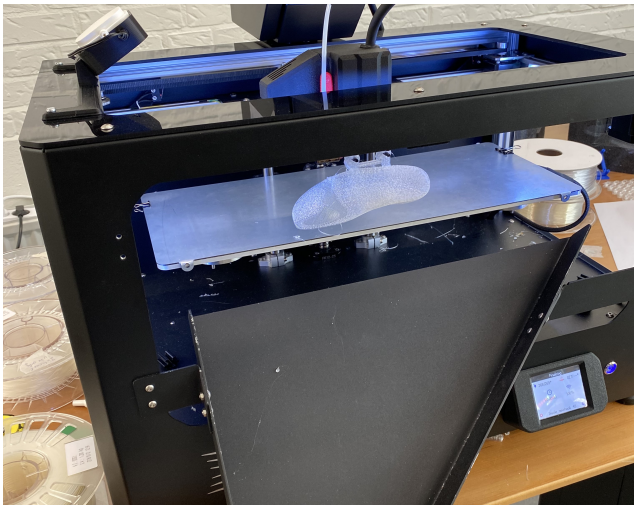
(a) Fork catching the print.



(b) The print plate starts to descend.



(c) The print is free from the print plate.



(d) The print head moves to the edge of the print plate.

Figure 2.2: The procedure of removing the print from the print plate

After the print has been removed from the print plate, it is necessary to verify whether or not the print plate is indeed empty. This can be done by either manually going to the printer and inspecting it, or by manually inspecting the printer remotely with the use of a camera mounted on the printer. A third way to do it could be to automate this by the use of computer vision and machine learning

together with a camera mounted on the printer.
A simple check using computer vision and machine learning could be to have an image of an empty print plate which is used for the reference model and then have the printer check if the feed from the camera is identical to the reference image. If they are not identical, the printer is then to report that the print plate is not empty.
More advanced versions of verification can be developed where the printer could possibly check if the print plate was empty and if not, whether or not this would affect the upcoming print, and only report an error if it is estimated that the leftover material is in the way of the next print. Another advanced version could be that the check could be done during printing, to ensure that the printing is progressing as expected. This would require significantly more computational power as the machine learning would need to calculate how the print was to look at the given time and then compare. At the time of writing Create it REAL does not have computer vision and machine learning implemented. However, they are in the process of developing it, and for that reason, this will not be the focus of this master thesis.

2.2 Error handling

The printing process is not flawless and printing errors can happen. One such error can be the print loosening itself from the print plate and becoming a loose object on the print plate, this can affect other parts on the print plate if the print job was a combination of multiple parts. Some slicer programs have the option of sequential printing, which means that if there are multiple parts in the print job, the printer will complete one part before printing the next part. This differs from the standard procedure where the printer prints one layer at a time on each part but does come with limitations in terms of the number of parts on the print plate as the software must allocate extra space per model to ensure that there is space enough for the print head.

Another error that can happen could be running out of filament before the print job is complete. One common way to mitigate the damage of running out of filament is to install a filament run-out sensor. These sensors can come in a variety of designs, but some of the simplest ones are literally just a switch that is being pressed down when there is a filament, such as the one seen in Figure 2.3, and can thereby be used to see when there is no filament.

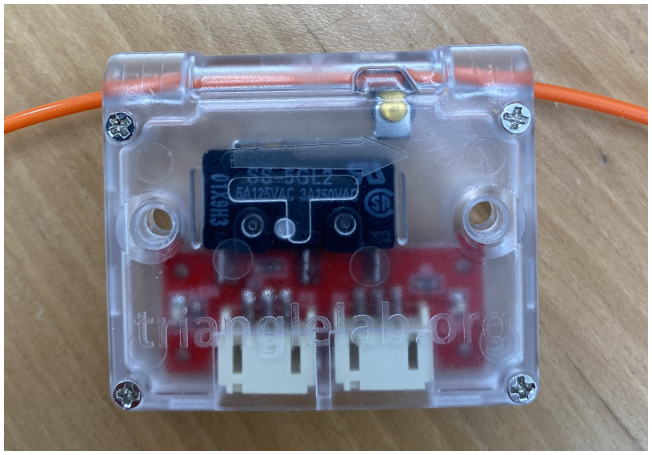
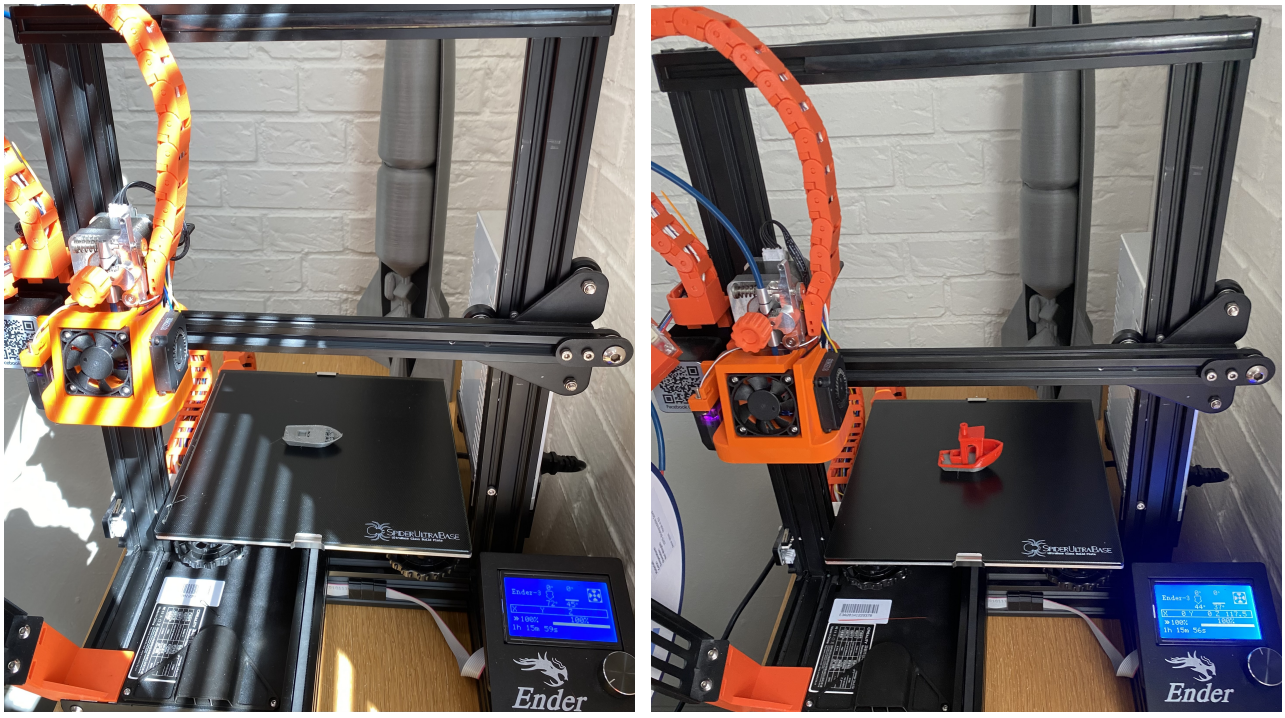


Figure 2.3: Triangle-lab filament run-out sensor with filament.

In the case of no filament, the printers can be programmed to pause the ongoing print, giving the

possibility of changing the filament before resuming the print and thereby ensuring that the print job does not critically fail. A critical failure could be the printer believing that the print job is complete even though the finished print job actually is only half finished due to running out of filament without the printer knowing and pausing.



(a) The printer did not notice it ran out of filament and believed it finished the print job, however, the print is only half finished.

(b) The printer did run out of filament, but it was detected and the filament was changed, so that the print job could finish.

Figure 2.4: A visualisation of the difference between having a filament run-out sensor and not having one.

2.3 Controlling the printers over the network

OctoPrint is an open-source software that is used to control a 3D printer on the local network. This printer is controlled over a web interface.[7] It is often installed on a Raspberry Pi or another single-board computer, by installing a specific Linux distribution called OctoPi. [5]
OctoPrint gives the possibility of accessing a webcam feed, in order to monitor the printer remotely. Other than visually being able to monitor the printer through the webcam it also gives constant feedback on the progress of the current print job, as well as gives information on the different temperatures and gives the possibility to change them during printing. It also gives the possibility of starting, stopping and pausing the current print job. [8]
Adding the Raspberry Pi with OctoPrint to a printer such as the Ender 3 makes it capable of the same as Create it REALs printers in terms of connectivity. However, Create it REAL has the printer connection integrated into their slicer, REALvision Pro, making it easy to send the sliced file to the printers. However, using OctoPrint, the user would first have to slice the file in a slicer, then save the G-code, open OctoPrint in the browser, and then send the G-code to the printer. The difference is also visible in that OctoPrint will only show you the information for a single printer, which means the user will have to open multiple windows to have an overview of all the printers that have a Raspberry Pi with OctoPrint connected. In REALvision Pro, the user can see all the printers on the WiFi, see their statuses as well as start, pause or stop print jobs.

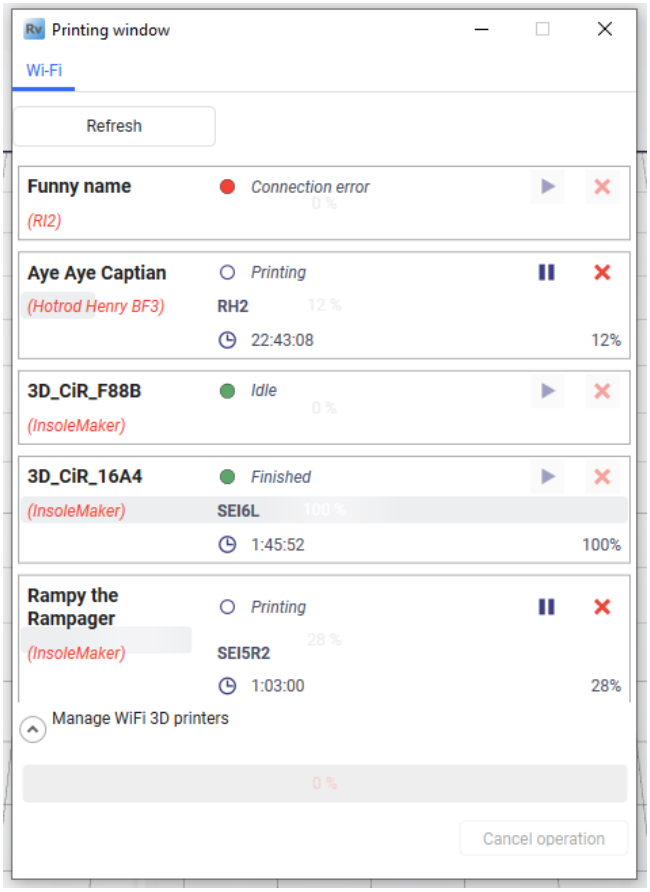


Figure 2.5: The WiFi menu in REALvision Pro. As the menu was opened without having sliced a model, the start button on 3D_CiR_F88B is greyed out, if a model had been sliced it would not have been greyed out.

One advantage of OctoPrint is that there is a community making plugins, one such plugin is called OctoPrint-Queue. [9] This plugin was designed to work in a library and gives the option of adding print jobs to a queue while sorting the print jobs depending on priority. The priority categories are: Urgent, Customer, Student, Internal, and Other. [9] As the printers made by Create it REAL all have WiFi connectivity, the main advantage of adding a Raspberry Pi to each printer, giving them WiFi connectivity falls to the ground. However, using a Raspberry Pi as a queuing server could possibly be a way to handle information of all the printers and the different client computers, it would however be recommended to run the queuing server on an already existing server if available. The reason for implementing on dedicated server hardware is due to the reliability of the hardware. A possible diagram for this can be seen in Figure 2.6. In the figure, it can be seen that the communication with each of the printers will be reduced compared to how the current system works, where each computer has to communicate with each printer. This can be seen in Figure 2.7.

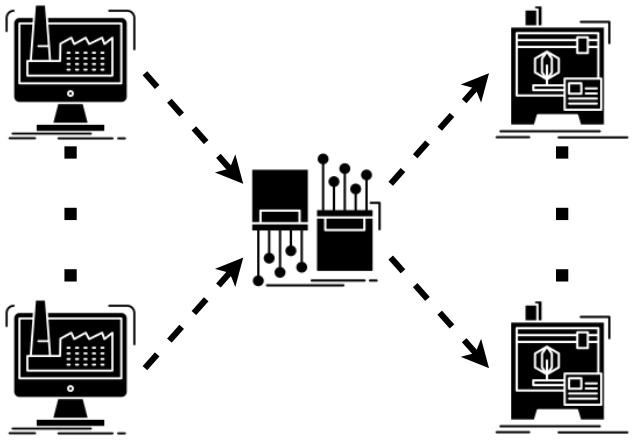


Figure 2.6: The figure shows the scenario where multiple computers communicate to a queuing server, which then communicates to the 3D printers.

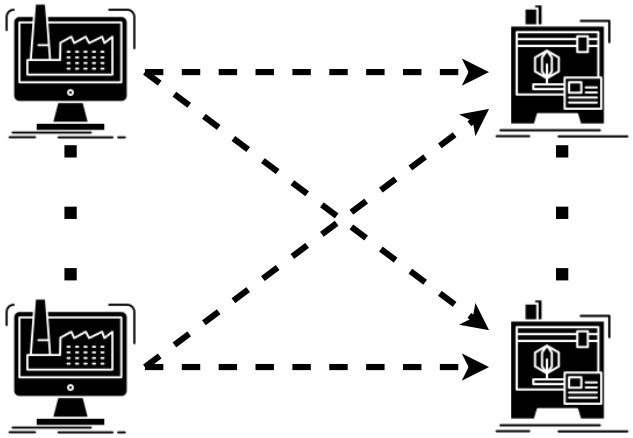


Figure 2.7: The figure shows how the current system works. Where each computer has to communicate to each 3D printer.

As Figure 2.7 visualises the current system, where all the computers communicate to all the printers and this is all done over WiFi, it is of interest to investigate how this communication actually takes place.

2.4 Network discovery of 3D printers

The current solution used by Create it REAL is using a multicast Domain Name System (mDNS) to discover the printers. One application where mDNS is also used is Bonjour. In Bonjour, it is used to enable computers to communicate ad hoc. [10] Bonjour is used when two Apple devices want to communicate by AirDrop, as it is used to create the peer-to-peer WiFi network. [11] This protocol is often used in Internet of Things (IoT) applications as it has proven to have a light footprint and good scalability. [12] Each of the printers in this case can be seen as an IoT device. As multicasting uses UDP packets for communication, there is a risk of packet loss. The effect of packet loss can be mitigated by re-transmitting the packets. The way mDNS functions is by a client sending a UDP packet to the IPv4 multicast address 224.0.0.251 or the IPv6 multicast address ff02::fb both on port 5353.[10]

Then the devices listening to that multicast IP address respond to the multicast IP address with their DNS resource records, resulting in the client receiving the record. [10] As the response was transmitted by multicasting, the other devices on the network also receive the records from the device that responds. Thereby they can update their records as well. This protocol is often used together with something called DNS Service Discovery (DNS-SD). [10] The combination of mDNS and DNS-SD works by using mDNS as the communication protocol and using DNS-SD for the service discovery and service description.[12]

In the implementation of mDNS that Create it REAL has made, they scan for devices running a specific service. The service they are scanning for is only running on their 3D printers, ensuring that they only discover the IP addresses of their own 3D printers.
The query for a given service is called a PTR Query. The response to this query is called an SRV Response. Due to confidentiality, an example of the request and response has been created and can be seen in Table 2.2.
In the table it can be seen that the IP address of 10.0.0.133 sends out a query to the multicast IP 224.0.0.251, expecting a multicast response from any devices running the service _aagren3dprinter._tcp.local. It can also be seen that there in fact are five devices running this service on the network. As the service is using UDP packets where there is a risk of packet loss, this request for devices is transmitted multiple times. The table only shows one retransmission of the request, however in reality Create it REAL requests a total of four times, in order to ensure that all the devices have had a chance to respond. As only one response is needed during the four attempts, and thereby the failure is only when there is no response from any of the four attempts, the calculation for the probability of a failure would be:

$$P(X = 0) = p^0 * q^4 = q^4 \tag{2.1}$$

where:

- P is the Probability of no answer for all the requests.
- p is the probability of an answer.
- q is the probability of no answer.

Thereby, if there is an 80% probability of an answer and 20% probability of no answer. Then the probability of no answer in all 4 attempts would be equal to 0.16%.

Now having established that the 3D printers made by Create it REAL has WiFi connectivity and they use mDNS to discover the printers. The next section will investigate further how the print jobs are sent to the printers, what kind of encryption is needed, and how a queuing system possibly could be implemented.

2.5 Transmitting a print job

As mentioned in Chapter 1, some of Create it REAL's customers are orthopaedic doctors, which makes it fair to assume that there could be some sort of personal data connected to the print jobs. For that reason, it is important to encrypt the print jobs. Furthermore, encrypting the print jobs has the advantage of complicating the theft of a print job.

2.5.1 Security

In the scenario where a competitor has gained access to the WiFi of an orthopaedic doctor and is listening on the network for print jobs, the encryption needs to be difficult enough to decrypt

T	*S*	*D*	*P*	*L*	*Info*
16.741	10.0.0.133	224.0.0.251	MDNS	84	Standard query 0x0000 PTR _aagren3dprinter._tcp.local, "QM" question
17.017	10.0.0.203	10.0.0.133	MDNS	239	Standard query response 0x0000 PTR AAGREN_3D_1094._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_1094.local TXT A 10.0.0.203
17.058	10.0.0.234	10.0.0.133	MDNS	189	Standard query response 0x0000 PTR AAGREN_3D_16A4._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_16A4.local TXT A 10.0.0.234
17.103	10.0.0.112	10.0.0.133	MDNS	210	Standard query response 0x0000 PTR AAGREN_3D_1AC0._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_1AC0.local TXT A 10.0.0.112
17.103	10.0.0.242	10.0.0.133	MDNS	213	Standard query response 0x0000 PTR AAGREN_3D_FA51._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_FA51.local TXT A 10.0.0.242
17.507	10.0.0.111	10.0.0.133	MDNS	239	Standard query response 0x0000 PTR AAGREN_3D_36E6._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_36E6.local TXT A 10.0.0.111
18.243	10.0.0.133	224.0.0.251	MDNS	84	Standard query 0x0000 PTR _aagren3dprinter._tcp.local, "QM" question
18.541	10.0.0.111	10.0.0.133	MDNS	239	Standard query response 0x0000 PTR AAGREN_3D_36E6._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_36E6.local TXT A 10.0.0.111
18.553	10.0.0.112	10.0.0.133	MDNS	210	Standard query response 0x0000 PTR AAGREN_3D_1AC0._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_1AC0.local TXT A 10.0.0.112
18.554	10.0.0.242	10.0.0.133	MDNS	213	Standard query response 0x0000 PTR AAGREN_3D_FA51._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_FA51.local TXT A 10.0.0.242
18.598	10.0.0.234	10.0.0.133	MDNS	189	Standard query response 0x0000 PTR AAGREN_3D_16A4._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_16A4.local TXT A 10.0.0.234
18.615	10.0.0.203	10.0.0.133	MDNS	239	Standard query response 0x0000 PTR AAGREN_3D_1094._aagren3dprinter._tcp.local SRV 0 0 80 AAGREN_3D_1094.local TXT A 10.0.0.203

Table 2.2: The table shows a snippet of how the communication could look. The services and hostnames have been anonymised by using a similar structure but with the author's surname. T = Time, S = Source, D = Destination, P = Protocol, L = Length.

without the pre-determined information, to ensure that the thief cannot manage to print the print job and contact the customer before the doctor has sold it to the customer. Encrypting the print job also complicates the theft of any printing technique used by Create it REAL that might be their intellectual property. The approach taken by Create it REAL to prevent a thief to print the print job immediately after a theft is that they encrypt all the print jobs by using Advanced Encryption Standard (AES) with a key size of 128, this is often written as AES128. Furthermore, each print job is encrypted separately, which results in that even if someone manages to crack the encryption, then that encryption key is only valid for that one print job and will be useless for the next print job. AES uses the encryption algorithm called Rijndael [13], the description and definition of Rijndael is described in [14].

2.6 Queueing Algorithms

One way to look at the system could be to look at the 3D printers as multiple processors. However, it is important to remember that the print jobs are physical products as soon as the printing process has started and for that reason when thinking of the print jobs as processes, they are not preemptive. Common queuing algorithms that will be investigated are First-In-First-Out (FIFO), Round Robin, Shortest Job First (SJF), Priority Queuing, Least Laxity First (LLF), Earliest Deadline First (EDF), Maximum Urgency First (MUF), and Modified Maximum Urgency First (MMUF).

2.6.1 First-In-First-Out

The first and simplest algorithm is called First-In-First-Out (FIFO), also called First-Come-First-Serve (FCFS). This algorithm is used when as the name gives, the first input would be the first output.[15] This algorithm could easily be used in a setup with just one printer as the first print job could be sent to the printer and the next print jobs would queue up to be sent.

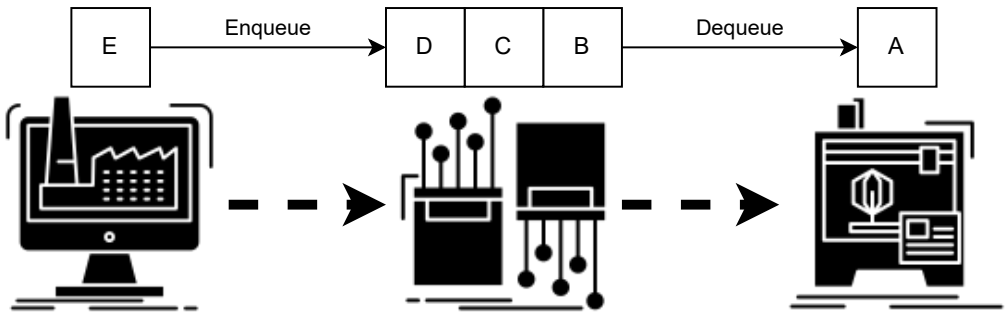


Figure 2.8: FIFO queue, where print job one has been sent to the printer, three print jobs have been queued already and a fifth print job is in the process of getting queued

In the case where there are multiple printers, FIFO can still be used, but would need additional instructions, as to which printer each print job should be sent to. One approach could be to use FIFO for the print jobs, while then use Round Robin for the printers.

2.6.2 Round Robin

This algorithm is meant to assign processes a time interval. [15] However, in the case of a 3D printer farm it is possible to look at it as the printers being assigned a time interval. Thereby, Printer A has X amount of time to receive print job 1 if it does not respond inside the time interval, then Printer

B has X amount of time to receive print job 1, and so on. At the time of writing the printer does have the possibility to respond while printing, but the WiFi communication has a low priority in the firmware to ensure that the communication does not cause errors during the printing. Thereby, the printers do not always respond. If Printer A manages to respond within the time interval then the algorithm would continue on to printer B with print job 2.
Thereby, making a Round Robin on the printers and a FIFO on the Print jobs. The challenges with using the Round Robin for the printers could be difficulties predicting when a print job, in reality, would be finished. Another issue could be determining the size of this time interval, if the window is too small, the printers will not manage to respond in time, and if it is too large, time might be wasted waiting for multiple printers to respond even though they are known to not answer as they are in the process of printing.
This could happen if the print job was sent to an empty queue, but where printers 1, 2, and 3 are printing and printer 4 is vacant. The print job arrived in the queue right after the time window for printer 4, thereby Round Robin would have the print job wait out the time for printers 1, 2, and 3 before it could send the print job to printer 4.

2.6.3 Shortest Job First

This algorithm assumes that run times are known in advance, and in fact, for 3D printing, there is an estimated print time which could be used as the run time. The algorithm is a batch algorithm, which means that it will run best if there is a batch of print jobs that is run through the algorithm. An example given in [15] shows how four jobs have different average turnaround times whether or not SJF is used or not. Using the five print jobs from Figure 2.8 and adding the expected print time for these print jobs, an example of this can be shown.

Hours:	1	4	6	2	3
Print jobs:	A	B	C	D	E

Table 2.3: The print jobs without using SJF, where A is the first and E is the last.

The turnaround time for each of the print jobs is calculated by:

$$TA_n = \sum_{m=0}^{n} T_m \tag{2.2}$$

where:

- TA_n is the turnaround time of print job n.
- T_m is the printing time of print job m.
- m is the first print job in the queue.
- n is the last print job in the queue.

Seen from Table 2.3, the turnaround time for print job A would be 1 hour, print job B 5 hours (the sum of the printing time of print job A and B), print job C 11 hours, print job D 13 hours, print job E 16 hours, giving an average of 9.2 hours. If SJF was used, then the order would be A, D, E, B, and C. This order would make the turnaround times for the print lower, as they would be: 1 hour for print job A, 3 hours for D, 6 hours for E, 10 hours for B, and 16 hours for C, giving an average of 7.2 hours. Mathematically, it can be shown that choosing the shortest job first brings the average print

time down. This can be seen by the formula for turnaround time given in [15]:

$$\begin{aligned} TA_{Avg} &= \frac{TA_A + TA_B + TA_C + TA_D}{4} \\ &= \frac{(T_A) + (T_B + T_A) + (T_C + T_B + T_A) + (T_D + T_C + T_B + T_A)}{4} \\ &= \frac{(4T_A + 3T_B + 2T_C + T_D)}{4} \end{aligned} \tag{2.3}$$

where:

TA_{Avg}	is the average turnaround time of the print jobs.
TA_A	is the turnaround time of print job A.
TA_B	is the turnaround time of print job B.
TA_C	is the turnaround time of print job C.
TA_D	is the turnaround time of print job D.
T_A	is the print time of print job A.
T_B	is the print time of print job B.
T_C	is the print time of print job C.
T_D	is the print time of print job D.

It can be seen that the first job has a significantly higher impact on the average than the fourth, which is why the largest job should be the last and the shortest be the first.
In the case of a 3D printing farm, the jobs are not expected to arrive in batches, which does cause some difficulties. In [15] it is mentioned that SJF is optimal only when all the jobs are available simultaneously. With this knowledge and with the assumption that all the print jobs are not sent to the queuing system in one big batch. It is possible to come up with a scenario where a print job keeps being pushed backwards in the queue.
The scenario goes: The printer is vacant, and the queue is empty. Then print job A is sent to the queue and the queue automatically sends to the vacant printer. Print job B estimated to have a print time of 5 hours is sent to the queue and as there are no other prints it is first in the queue. Then print job C is sent to the queue, print job C is estimated to take 3 hours and for that reason is put in front of B. Then print job D arrives in the queue and as it is estimated to take 3.5 hours it is put in front of B, but behind C. This can then continue as long as a print job with a lower estimated time than 5 hours arrives, and as long as the print job arrives in the queue before print job B starts.
For the reason of possibly trapping a print job in queue for eternities, this algorithm will not be considered viable for the print jobs as a standalone, but could possibly be used in combination with other algorithms.
One way this algorithm possibly could be used could be to choose the printer to which the next print job should be sent. It can be assumed that all the printers are discovered when a print is to be sent to a printer. Thereby the group of printers could be seen as the input batch to the algorithm, their individual times could be the remaining print time of the print jobs on the printers. If multiple printers are tied for the shortest time, then priority queuing could be used.

2.6.4 Priority Queuing

This algorithm stands out from the others as both FIFO, Round Robin, and SJF do not assume a certain print job is more important than another. SJF did prioritise one print job over another, but the prioritisation was done depending on the printing time of a print job, rather than a user-defined prioritisation. The idea of prioritising print jobs is rather simple. The base queuing could be done on for example FIFO or SJF, but where it is done in multiple sections. A scenario for this could be,

5 print jobs that are to be sorted based on priority and FIFO. The print jobs arrive alphabetically in the queuing system but are not sent to any of the printers before sorting. Print job A, C, and E have low priority, while print job B and D has high priority. Thereby, the order in the queue would be B, D, A, C, E. One case where priority could be needed would be in the case of failed prints. In this case, it is important that the print is restarted on the next available printer as to have the best chance of living up to the initial expected completion time.

If expanding the scenario to a print job F with high priority arrives after the prioritisation of the initial 5 print jobs, then it is needed to run the algorithm again. However, practically speaking, the print job would be put to the end of the high-priority queue, and the low-priority queue print jobs will have their estimated completion time extended.

Priorities can also be dynamically assigned as mentioned in [15]. The goal could be to print the print job within 24 hours.
The procedure would then be:

- Add print job to the queue, with a chosen priority.
- The algorithm would then queue it following the priority, but check whether the expected starting time added with the expected printing time is more than 24 hours after the print was added to the queue. If that is the case, move the print job to a higher priority.
- Every time a print job is added to the queue that moves the print jobs, then a check needs to be run.

This does come with the challenge that if the queue becomes too long, all print jobs might get moved to high priority, causing the priority queuing to be unnecessary. This can be mitigated by adding more printers to the print farm, or by loosening the goal to for example 48 hours. This way of dynamically assigning priorities is similar to the scheduling algorithms called Earliest Deadline First and Least Laxity First.

2.6.5 Earliest Deadline First

This algorithm is fairly simple, the base of the algorithm is priority queuing, but with dynamic priorities. EDF gives the highest priority to the jobs with the earliest deadlines. [16] An example with five print jobs and one printer can be made, the important part to remember is that print jobs are assumed to not be preemptive.

Print job	Arrival time	Execution time	Deadline
1	0	1	10
2	1	2	12
3	1	4	11
4	2	2	9
5	2	3	13

Table 2.4: The table shows the 5 print jobs with their different parameters.

If EDF is used to queue these print jobs, then the queue can be displayed in three steps.
Step 1:

1. Print job 1

As print job one comes in first and only takes one time unit it will be complete when step 2 starts and will not be shown further.
Step 2:

1. Print job 3
2. Print job 2

It can be seen that print job 3 will be prioritised higher than print job 2, as it has a lower deadline than print job 2. At step 3 two more print jobs arrive, however, as print job 3 is in progress this will stay as the highest priority.
Step3:

1. Print job 3
2. Print job 4
3. Print job 2
4. Print job 5

Thereby the total queue would have been:

Print job	Arrival time	Execution time	Deadline	Time to complete
1	0	1	10	1
3	1	4	11	5
4	2	2	9	7
2	1	2	12	9
5	2	3	13	11

Table 2.5: The table shows the finished print queue of the 5 print jobs with their different parameters.

If checking if all the print jobs are completed within their deadlines, it can be seen that they all do. However, this is a coincidence as it has been shown in [16], that it is not always the case that EDF delivers a queue where all the jobs complete within the deadline if they are not preemptive.
Furthermore, if the definition of deadline is set too simple, as in for example the deadline is always 24 hours after the print job is added to the queue, then EDF will basically become FIFO. Another way this could be used could be to have different deadline times depending on whether it is a premium customer or a standard customer, where the premium customer would have a deadline of 12 hours from the time the print job is added to the queue and the standard customer would have a deadline of 24 hours from the time the print job is added to the queue.

2.6.6 Least Laxity First Scheduling Algorithm

This algorithm is also known by the name *Least Slack Time First Algorithm*. The principle idea of this algorithm is that uses the deadline, the execution time, and the current time to calculate the *Laxity*. [17]

$$L = D - E - T_{now} \tag{2.4}$$

where:

L is the laxity of the job.
D is the deadline for the job.
E is the execution time of the job.
T_{now} is the time when laxity is being calculated.

Then the algorithm prioritises the print job with the lowest laxity. [16]
Then the updating of the laxity would be done at chosen times or could be done every time something is added to the queue.

Least Laxity First is also expecting the jobs to be preemptive, but can possibly be modified in the same way as EDF, so assuming that the print job in process is not preemptive, and using the example jobs from EDF, the steps of the Algorithm would be:

Print job	Arrival time	Execution time	Deadline
1	0	1	10
2	1	2	12
3	1	4	11
4	2	2	9
5	2	3	13

Table 2.6: The table shows the 5 print jobs with their different parameters.

Step 1:

1. Print job 1 which has a laxity of 9

Print job one comes in first and only takes one time unit it will be complete when step 2 starts and it will not be shown further. However, the time is still counted in for when calculating laxity for the coming jobs. Step 2:

1. Print job 3 which has a laxity of 6
2. Print job 2 which has a laxity of 9

It can be seen that print job 3 will be prioritised higher than print job 2, as it has a lower laxity than print job 2. At step 3 two more print jobs arrive, however, as print job 3 is in progress this will stay as the highest priority. Step3:

1. Print job 3 which has a laxity of 5
2. Print job 4 which has a laxity of 5
3. Print job 2 which has a laxity of 8
4. Print job 5 which has a laxity of 8

The example used happens to give the same queue order as EDF did. This could be due to that the print jobs are non-preemptive as soon as they have started.
The main difference between EDF and LLF is that EDF only uses the deadline to prioritise, while LLF also uses the Execution time to calculate the Laxity, and that way prioritises based on the laxity.[17]

2.6.7 Maximum Urgency First

There is an algorithm that combines the advantages of EDF, LLF and Rate Monotonic algorithm (RM). This algorithm is called Maximum Urgency First (MUF).[17]
RM is a fixed priority scheduling algorithm, where the tasks with the highest frequency are assigned the highest priority and the lowest frequency - the lowest priority.[17] In the case of 3D printing this algorithm could possibly be used if there are specific models that are sent to the print queue frequently, and some less frequently. However, this is not considered further as each print job is assumed to be unique and only needs to be printed once, and RM does not support dynamic priority [17].
As mentioned, MUF combines the advantages of EDF, LLF, and RM. It does that by mixing three levels of priorities, the first level is fixed priority and is called *Criticality*, the second level is dynamic and is based on LLF, and the third is fixed and is called *User priority*. This kind of priority is called *Mixed priority*. The order of precedence is Criticality over LLF over User priority. [17]
The assignment of criticality in the original design is done with RM [17], but in the context of 3D printing where each print job would be a task, the criticality could be determined by the user. The criticality values could possibly be:

1. Print error
2. Premium customer
3. Standard customer

Here the highest priority would be the lowest number. The user should only be able to choose between 2 and 3, while if the printer detects an error during printing and therefore re-schedules the print, then priority 1 should be used.
The assignment of user priority must also be chosen, this could possibly be the employee ID or the time it was added to the queue.
The assignment of the dynamic priority is done during run-time, but before adding print jobs to the queue some parameters must be set, these are:

- Desired start time.
- Deadline time.
- Worst-case execution time.

These parameters are used to calculate the laxity of the print jobs. [17] In the 3D printing setting, the desired start time could be automatically chosen to be the time the print job was added to the queue. The deadline time could be chosen to be 24 hours after the print was added to the queue, and the worst-case execution time could be found from the G-code as the estimated printing time.
The way that MUF selects which print job is to be printed is done in these steps:

1. Select the print job with the highest criticality, there by any print job with the value of 1 (Print Error).
2. If there are no print jobs with 1 then it will choose 2 and if none with 2 then 3.
3. If there are two or more print jobs with the highest criticality, then it will choose the one with the lowest laxity.
4. If there are two or more that have the same criticality and the same laxity, it will choose the print job with the highest user priority.

5. If there after running these three checks still are two or more print jobs, then it chooses by the FIFO principle

With the unique user priority it is claimed in [17] that the scheduler never reaches step 5 and is therefore a deterministic scheduling algorithm.[17]
MUF does stand strong as a contender for the queuing algorithm used for this system as it does provide a method of prioritising re-prints of a failed print job over newly added print jobs. Furthermore, it does provide a possible way of handling the prioritisation of print jobs with different sizes without the risk of suffocating one specific size of print jobs. Furthermore, the user priority gives the possibility of choosing priority depending on the time the print job was added to the queue, or simply by the employee ID.

2.6.8 Modified Maximum Urgency First

Having looked into MUF, it is interesting to look at a modified version of MUF. A Modified Maximum Urgency First (MMUF) algorithm was presented in [18]. They do show that MUF in some cases can fail to execute a task before the deadline, due to how LLF is defined.[18] The example they used was:

Task	Arrival time	Execution time	Deadline	Laxity
1	0	4	6	2
2	0	1	4	3

Table 2.7: The table shows the example from [18].

In this example task 1 will execute for four time units, causing task 2 to miss the deadline, this is due to when the LLF would update the laxity number.[18] Some of the modifications introduced were to use EDF or a Modified Least Laxity First (MLLF) instead of LLF for the second check. MLLF is in essence a non-preemptive version of LLF. [18] Another modification is that if there in any of the checks are a tie and one of the print jobs are already printing, this one will continue and the other will be printed afterwards. [18].

As the modifications in MMUF are introducing a non-preemptive version to MUF, and the main difference on the two are the second check, where MUF uses LLF and MMUF suggests to use either EDF or MLLF. This algorithm will be considered viable for the system as well as MUF.
Now having investigated some of the common queuing algorithms and challenges of automating a 3D printing farm, a problem formulation can be made.

2.7 Problem Formulation

How can a queuing system be made that considers the amount of filament left on each printer, the working hours of the company, and that multiple computers might want to add print jobs to the queue?

3 Requirements

In order to verify the design of the Printfarm Management System, a set of requirements is established.

Requirement	Explanation
The system must be able to group printers depending on their printer model.	This requirement is set with the assumption that all the printers of the same model have the same hardware and firmware installed.
The system must be able to group the printers depending on both the printer model and the installed filament.	This requirement is set in order to ensure that a print job meant for one material does not accidentally get sent to a printer with a different material.
The system must be able to queue a print job to a group of printers.	This requirement is set in order to give the queuing algorithm freedom to choose which printer of the chosen printer model can complete the print fastest.
The system should show the estimated print time for the print job on the chosen printer group.	This requirement is set in order to give the user an idea of when the print job earliest can be ready for delivery to the customer.
The system should be able to allocate the print jobs in queues for each separate printer.	This requirement is set in order for the system to make a more precise estimate of completion time.
The system must be able to change the queues, in order to adapt to errors during printing.	This requirement is set in order to do error handling. In the case of running out of filament or other errors that were not detected during printing and are only caught by the visual inspection, then the print job needs to be sent to another printer and started as soon as possible.
If a queue manager is on the network, this must be the only device communicating with the printers.	This requirement is set in order to ensure that no other device interrupts the queuing system.
If there is no queue manager on the network, the communication to the printers will work as previously.	This requirement is set in order to ensure that the present functionalities in REALvision do not rely on the queue manager being present.
The queuing algorithm should be able to handle multiple factors, such as priorities, opening hours, estimated print time, and the expected amount of filament left.	This requirement is set to provide an advanced queuing algorithm that provides the fastest print times, where the time for a change of filament is calculated into the print time.
The system must give an alert if an error happens during print.	This requirement is set in order to alert the administrator to change filament or other physical tasks.

Table 3.1: Requirements for the Printfarm Management System

4 System Design

The system will consist of two computers, one server, and three 3D printers. The computers and printers will be connected to a router by the use of WiFi, while the server will be connected by Ethernet. The computers will then communicate to the server, which will then communicate to the printers.

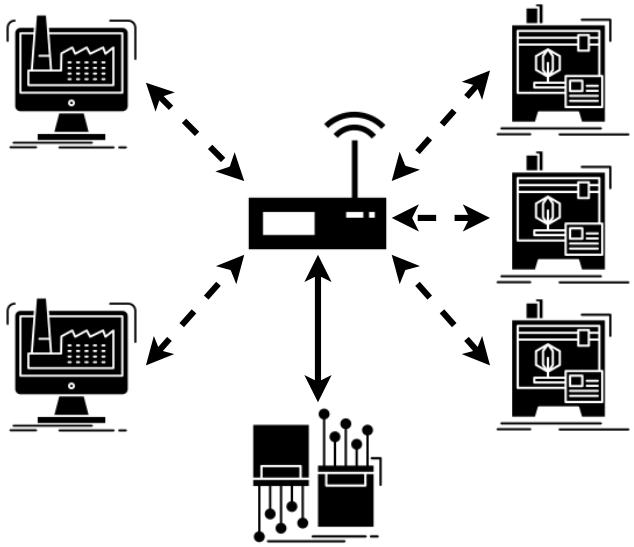


Figure 4.1: System Overview, with two computers, one router, one server and three 3D printers

The server will be running the queuing system and will be the centralised way to communicate with the printers.
The three printers chosen for this project are made by Create it REAL and are of the model Insole-Maker.

4.1 The Combination of Queuing Algorithms

When designing the system it is important to remember that there are two things that need to be sorted, the print jobs and the printers to print on.
A set of simple scenarios that can be seen as the fundamental parts are made. These scenarios must be handled by the algorithms for the system to be functional.

1. All printers are vacant.
2. Some printers are in the process of printing, but at least one printer is vacant.
3. All printers are in the process of printing, and there are no printers vacant, there are no print jobs in queue for any printer.
4. All printers are in the process of printing, and there are no printers vacant, there is a print job in queue for all printers.

The pairing of algorithms for the printers and print jobs for the four scenarios are as follows:

Scenario	Printers	Print jobs
1	Fixed priority, where the priority is equal to the model number.	FIFO.
2	Fixed priority on the vacant printers.	FIFO.
3	SJF, where the values used are the current print job's print time.	The chosen algorithm depends on the results from simulations.
4	SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.

Table 4.1: The pairing of algorithms used for the printers and the print jobs for the four basic scenarios.

In the first two simple scenarios, the choice of which printer the print job should be sent to is relatively simple as the important part is that a vacant printer is chosen. In scenarios three and four where there are no vacant printers, SJF can be used. As all the printers are printing a print job, SJF can be used to find the printer that is expected to become vacant first. The next time a print job is scheduled and SJF is run, the value for "time before vacant" for the previously chosen printer would be the sum of the print time of the print job in process and the print job in the queue for the printer.

For the more advanced case where the opening hours and the remaining amount of filament are considered, another set of scenarios can be made.

1. All printers are in the process of printing, and there is more time left of the workday than the print time.
2. All printers are in the process of printing, and there is not enough time left in the workday to complete the print on some of the printers.
3. All printers are in the process of printing, and there is not enough time left in the workday to complete the print on any of the printers.
4. All printers are in the process of printing, and there is more time left of the workday than the print time, but there is not enough filament to complete the print on some of the printers.
5. All printers are in the process of printing, and there is more time left of the workday than the print time, but there is not enough filament to complete the print on any of the printers.
6. All printers are in the process of printing, and there is not enough time left in the workday to complete the print, and there is not enough filament to complete the print.

With these advanced cases, the algorithm combination becomes complex and decisions have to be taken, however, the first one is similar to the third and fourth of the simple scenarios:

Scenario	Printers	Print jobs
1	SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.
2	Filter the printers so only those that can manage are used for the algorithm. Then SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.
3	SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.
4	Filter the printers so only those with enough filament are used for the algorithm. Then SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.
5	Notify the user that the printer will run out of filament during printing. Then SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer.	The chosen algorithm depends on the results from simulations.
6	Notify the user that the printer will run out of filament during printing, and that this will happen outside working hours. Then SJF, where the values used are the sum of the current print job's print time and the print time of all the print jobs queued for the printer. Send a reminder 10 minutes before closing time to change the filament, if it has not run out before.	The chosen algorithm depends on the results from simulations.

Table 4.2: The pairing of algorithms used for the printers and the print jobs for the advanced scenarios.

In order to be able to filter the printer depending on whether or not they have filament enough, calculations are made to determine the minimum length of filament on a roll.

4.2 Running out of filament

In REALvision, it is possible to see the estimated printing time, and estimated material use both in grams and meters. The rolls of filament used do not directly provide the length of the filament, but it does provide the weight, density and diameter, from this it is possible to calculate the length of

the filament with the formula given:

$$L = \frac{W/\rho}{\pi * \left((\frac{D}{2})^2\right)} \tag{4.1}$$

where:

L	is the length of the filament.
W	is the weight of the filament.
ρ	is the density of the filament.
D	is the diameter of the filament.

Thereby, the length of 1Kg filament with a density of 1.25 and a diameter of 1.75mm is around 332.60 meters, however, the precision of the diameter is $\pm$ 0.05mm. Thereby, the length is somewhere between 314.38 meters and 352.45 meters. With the imperfection of the diameter in mind, the algorithm should count for a roll of filament to be equal to 314.38 meters. This does bring the possibility of changing filament before running out and safeguards against running out of filament during the hours when there is no administrator present to change the filament. However, it does come with the challenge of rolls of filament that are not empty but not being used.

An approach to deal with these non-empty rolls of filament could be to use them next time a printer needs filament change during the morning, as these would be used up during the work day and the printers would then detect out-of-filament before the administrator goes home. This is fair to assume as the maximum amount of filament on these changed rolls would be 38.07 meters, and the sample insole provided by Create it REAL takes 2 hours and 7 minutes to print while using approximately 21.71 meters of filament. Thereby, it can be calculated that to print 38.07 meters approximately takes 3 hours and 43 minutes, which means that if the non-empty roll was changed to be used in the morning, then a new roll of filament could be changed to be used around lunchtime. Furthermore, a full roll of filament should be able to print for around 30 hours and 39 minutes, without counting time for transferring the print jobs, auto bed levelling, heating up, extracting the finished prints, and validating the empty print plate.
The calculation where these additional things are counted in for the estimated minimum time the printer can print with a new roll of filament could be:

$$T_{Total} = T_{Roll} + \lceil \frac{T_{Roll}}{T_{Print}} \rceil * T_{Trans} + T_{InitProc} + \left(\lceil \frac{T_{Roll}}{T_{Print}} \rceil - 1 \right) * T_{ContProc} + \lfloor \frac{T_{Roll}}{T_{Print}} \rfloor * (T_{Extra} + T_{Val}) \tag{4.2}$$

where:

T_{Total}	is the total time before a filament change is needed.
T_{Roll}	is the time it would take to print the whole filament roll in one go.
T_{Print}	is the time the demo insole takes to print.
T_{Trans}	is the time it takes to transfer the print job to the printer.
$T_{InitProc}$	is the time it takes for the initial process of auto bed levelling and heat up.
$T_{ContProc}$	is the time it takes for doing the continued process of auto bed levelling and heat.
T_{Extra}	is the time it takes to extract the print from the printer.
T_{Val}	is the time it takes to validate that the print plate is empty.

Using this, it is possible to estimate that if the transfer takes 3 minutes, the initial process takes 4 minutes, the continued process takes, 2.5 minutes, the extraction and validation each takes 0.5 minutes, and the printer can print for approximately 32 hours and 17 minutes. If the working hours

are from 8:00 to 16:00, and the change to the new roll of filament happens at 12:00, then the roll would run out of filament around 20:17 the next day. With that knowledge it could possibly be advantageous to change the filament at 8:00 as then the next filament change would be 16:17 the next day and the next after that would be around 00:34. Thereby, it can be seen that changing filament once per day would be needed in order to have as little downtime as possible.

If, however, Create it REAL had a device that could switch between multiple rolls of filament automatically, then the uptime of a printer could grow substantially. If the device used 3 minutes to change between the filament, then a time plan for the different filaments could look like this:

	Day 1	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Roll 1	08:00	16:17			
Change		3 min			
Roll 2		16:20		00:37	
Change				3 min	
Roll 3				00:40	08:57

Table 4.3: Time plan for use of 3 rolls of filament

From the time plan in Table 4.3 it can be seen that on Day 3 the administrator can at any point of the day change filament roll 1 out with a new roll, and if the administrator wanted to utilise that there are 3 rolls, then the administrator could wait until Day 4 and then change both Roll 1 and Roll 2, without disturbing the continuous printing.

Now knowing the minimum length of the filament on a roll, and how having the capability of changing between the rolls can contribute to less amount of manual interactions with the printers, it is time to determine which of the algorithms investigated in Chapter 2 is best suited for the sorting of the print jobs.

5 Simulations

In order to verify which of the algorithms investigated is best suited for a 3D printing farm, a set of simulations is made. The purpose of the simulations is to investigate the average turnaround time of the print jobs, the number of print jobs finishing within their deadline, and to investigate the total print time of all the print jobs.
The simulations will have these parameters:

Attribute	Value
Number of print jobs	100
Number of printers	3
Printing time	30min - 150min
Priority	2 - 3
Deadline	24 hours after the print job has been created
Initial Laxity	Deadline - Creation time - Printing time
Update Laxity	Deadline - Current time - Printing time
Employee ID	1 - 10
Print job creation interval	5 min

Table 5.1: Common parameters of all the simulations

The algorithms that will be tested are FIFO, SJF, Priority Queuing, LLF, MUF, and MMUF with EDF as the second check. As the deadline for each of the print jobs is 24 hours after being created, the EDF will not be simulated alone as the results would be the same as FIFO. Furthermore, SJF and Priority Queuing will have FIFO as the second check in case they are comparing to print jobs with equal values for their respective checks. It is therefore expected that Priority Queuing and MMUF with EDF as the second check will perform exactly the same, but will be simulated in order to confirm this.

The simulations will be run 100 times with each of the algorithms, and their results will be compared. The aim is to determine which algorithm has the lowest average turnaround time of the print jobs while having the highest amount of print jobs that finished within their deadline and used the lowest amount of time to print all the print jobs.
The results for each algorithm will be the average from all 100 times the simulation was run.
The structure of the simulations will be as follows:

1. Initial steps:
   (a) Create a print job.
   (b) Add the print job to the queue.
   (c) Sort the print jobs in the queue.
   (d) Check if there is a printer vacant.
   (e) If there is a printer vacant, send the first print job in the queue to the printer.
   (f) If there is no printer vacant, check if a print job has finished on a printer.
   (g) Increment the current time by 5 minutes.
   (h) Update the laxity of all the print jobs in the queue.

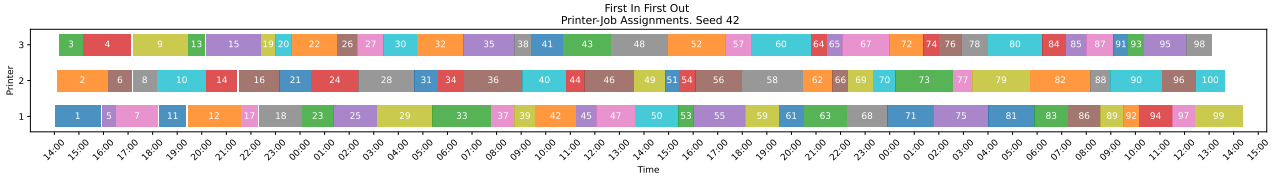
   (i) Sort the print jobs in the queue.

2. When all the print jobs have been created, and there are still print jobs in the queue:
   (a) Check if a print job has finished on a printer.
   (b) If a printer is vacant, send the first print job in the queue to the printer.
   (c) Increment the current time by 1 second.
   (d) Update the laxity of all the print jobs in the queue.
   (e) Sort the print jobs in the queue.
3. When all the print jobs have been created, and there are no more print jobs in the queue:
   (a) Check if a print job has finished on a printer.
   (b) Increment the current time by 1 second.

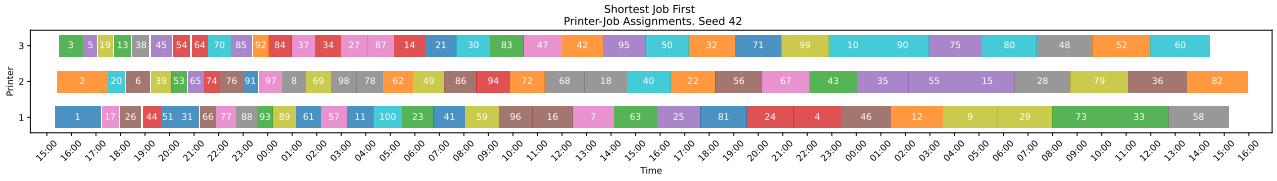
By generating the print jobs one at a time, and then checking if it is possible to send a print job to a printer, it is expected that some time might get lost between when a print job is finished and when a print job can be sent to the printer. The maximum amount of time lost per generation of print jobs would be 4 minutes. This is due to that a print job is generated every five minutes and the printing time of a print job is given in minutes. Thereby, if the printing time of a print job is 31 minutes and other print jobs are being generated every 5 minutes, then there will be a check at time 0, 5, 10, 15, 20, 25, 30, and 35. It is only at the check at time 35 that the check would be positive for that the print job finished even though it finished 4 minutes earlier. As soon as all the print jobs have been generated, the check comes every second, causing a maximum time lost of 1 second per check.

5.1 Simulation Results

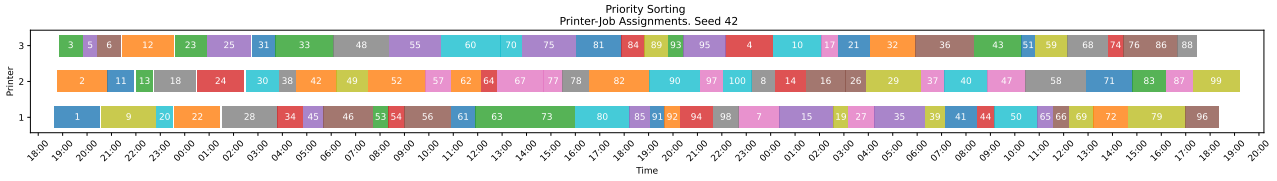
A visualisation of the result from a simulation of each of the algorithms can be seen in Figure 5.1



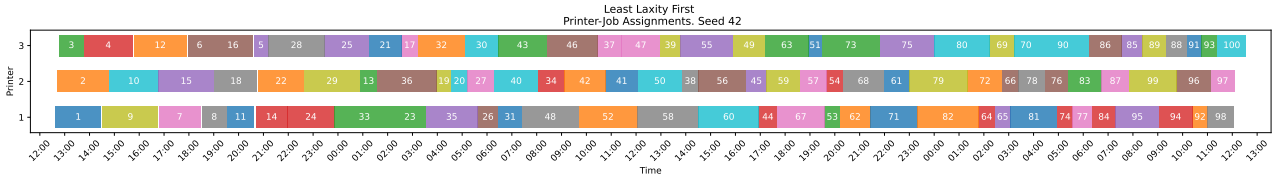
(a) FIFO scheduling.



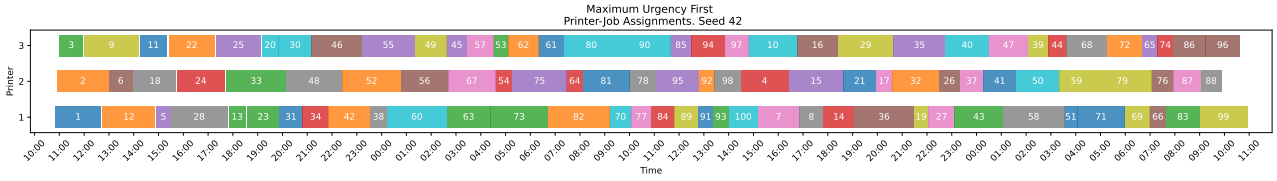
(b) SJF scheduling.



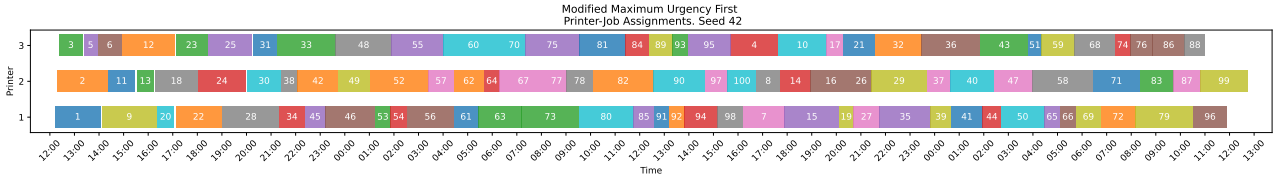
(c) Priority scheduling.



(d) LLF scheduling.



(e) MUF scheduling.



(f) MMUF scheduling.

Figure 5.1: A visual representation of the different algorithms with 3 printers and 100 randomly generated print jobs with a seed value of 42.

The average results of all the simulations of the algorithms were found and organised in Table 5.2.

Algorithm	Total printing time	# of print jobs missing deadline	Average turnaround time
FIFO	2 days, 2:55:11.04	44.96	21 hours 54 minutes
SJF	2 days, 3:17:12.32	32.22	16 hours 58 minutes
Priority Queuing	2 days, 2:48:24.01	47.46	21 hours 49 minutes
LLF	2 days, 2:29:07.44	47.86	23 hours 1 minute
MUF	2 days, 2:33:33.52	47.73	22 hours 22 minutes
MMUF	2 days, 2:48:24.01	47.46	21 hours 49 minutes

Table 5.2: Average results from 100 simulations of each of the algorithms used for queuing

Analysing the results from the simulations, it is possible to rank the algorithms and then after ranking all the algorithms for the three attributes sum the rankings up and give them an overall ranking.

5.1.1 Ranking the algorithms

When ranking the algorithms depending on the total printing time, the highest rank is the algorithm with the lowest total printing time.
Printing time ranking

1. LLF with a total printing time of 2 days, 2:29:07.44
2. MUF with a total printing time of 2 days, 2:33:33.52
3. MMUF and Priority Queuing with a total printing time of 2 days, 2:48:24.01
4. FIFO with a total printing time of 2 days, 2:55:11.04
5. SJF with a total printing time of 2 days, 3:17:12.32

It is interesting to see that SJF performed worse than FIFO on the total printing time, as FIFO was expected to perform worst.
When ranking the algorithms depending on the number of print jobs missing their deadline, the lowest value is again the best result and will be ranked the highest.
Print jobs missing their deadlines

1. SJF with an average of 32.22 print jobs missing their deadlines
2. FIFO with an average of 44.96 print jobs missing their deadlines
3. MMUF and Priority Queuing with an average of 47.46 print jobs missing their deadlines
4. MUF with an average of 47.73 print jobs missing their deadlines
5. LLF with an average of 47.86 print jobs missing their deadlines

It was expected that SJF would rank highest for jobs missing their deadlines as it prioritises the smaller print jobs and waits with the longest print jobs to the end, thereby it can be assumed that the majority if not all the print jobs that missed their deadlines are print jobs taking a long time to print. However, it was not expected that MMUF would perform better than MUF. As the main difference between the simulation implementation of MMUF and MUF was that MMUF was using the deadline to sort with instead of laxity which was used in MUF, it was expected that MUF would perform better.

When ranking the algorithms depending on the average turnaround time, the lowest value will again be ranked the highest.
Average turnaround time

1. SJF with an average turnaround time of 16 hours and 58 minutes
2. MMUF and Priority Queuing with an average turnaround time of 21 hours and 49 minutes
3. FIFO with an average turnaround time of 21 hours and 54 minutes
4. MUF with an average turnaround time of 22 hours and 22 minutes
5. LLF with an average turnaround time of 23 hours and 1 minute

Once again SJF runs off with the best ranking, this is expected though as SJF prioritises the shortest jobs, which affects the turnaround time for the next jobs as explained in Equation (2.3).
In order to visualise the algorithms and their rankings, a table can be made. The sum of the rankings can then be used to determine which of the algorithms would possibly be best for the system.

Algorithm	Printing time	Print jobs missing deadline	Turnaround time	Total ranking
FIFO	4	2	3	9
SJF	5	1	1	7
Priority Queuing	3	3	2	8
LLF	1	5	5	11
MUF	2	4	4	10
MMUF	3	3	2	8

Table 5.3: Rankings of the algorithms.

It can be seen from Table 5.3 that SJF is the best performing for the system, but taking into account the risk of starvation of print jobs with a long print time, this will not be chosen. Thereby, Priority Queuing and MMUF are the best scoring algorithms. The difference between the two is that the second check on Priority Queuing was the timestamp for the creation of the print jobs, while on MMUF it was the deadline for the print jobs. If the deadline had been randomly chosen between some time intervals, these two algorithms would not have performed the same. It is expected that the MMUF would perform better than Priority Queuing in that case, as it would sort the list depending on the deadline, rather than a FIFO approach.

5.1.2 Simulations with random deadlines for print jobs with priority 2

In order to verify which of the two algorithms actually performs best, another set of simulations is run. The difference between these simulations and the previous ones is that the deadline of each print job depends now on the priority. Thereby, if a print job has the priority of 2, the deadline is set to be:

$$Deadline = T_{Creation} + T_{Printing} + T_{MaxPrinting} + T_{Random} \tag{5.1}$$

where:

$T_{Creation}$	is the timestamp of when the print job was created
$T_{Printing}$	is the printing time of the print job
$T_{MaxPrinting}$	is the maximum printing time of a print job
T_{Random}	is a randomly chosen amount of time in hours between 1 and 24

The idea of adding the maximum printing time of a print job to the deadline calculation is to ensure that at least one other print job can be printed before the print job must be started for it to manage within the deadline. If a print job has a priority of 3, the deadline is set as in the previous simulation:

$$Deadline = T_{Creation} + T_{DeadlineHours} \tag{5.2}$$

where:

$T_{Creation}$	is the timestamp of when the print job was created
$T_{DeadlineHours}$	is 24 hours

The simulations were run for all the algorithms, but it is expected that the 2 of importance are Priority Queuing and MMUF. The average results of all the simulations of the algorithms were found and organised in Table 5.4.

Algorithm	Total printing time	# of print jobs missing deadline	Average turnaround time
FIFO	2 days, 2:39:30.47	52.02	21 hours 47 minutes
SJF	2 days, 3:03:45.69	38.38	16 hours 52 minutes
Priority Queuing	2 days, 2:42:20.15	66.18	21 hours 45 minutes
LLF	2 days, 2:44:10.64	51.94	21 hours 49 minutes
MUF	2 days, 2:42:01.85	55.69	21 hours 46 minutes
MMUF	2 days, 2:49:28.68	53.78	21 hours 20 minutes

Table 5.4: Average results from 100 simulations of each of the algorithms used for queuing

Once again the algorithms will be ranked where the lowest number is best.
Printing time ranking

1. FIFO with a total printing time of 2 days, 2:39:30.47
2. MUF with a total printing time of 2 days, 2:42:01.85
3. Priority Queuing with a total printing time of 2 days, 2:42:20.15
4. LLF with a total printing time of 2 days, 2:44:10.46
5. MMUF with a total printing time of 2 days, 2:49:28.68
6. SJF with a total printing time of 2 days, 3:03:45.69

It is interesting to see that FIFO in this run of simulations is the fastest, and only about 10 minutes slower than LLF in the previous simulation. However, SJF once again was the slowest just as in the previous simulations.

Print jobs missing their deadlines

1. SJF with an average of 38.38 print jobs missing their deadlines
2. LLF with an average of 51.94 print jobs missing their deadlines
3. FIFO with an average of 52.02 print jobs missing their deadlines
4. MMUF with an average of 53.78 print jobs missing their deadlines
5. MUF with an average of 55.69 print jobs missing their deadlines

6. Priority Queuing with an average of 66.18 print jobs missing their deadlines

Once again it is no surprise that SJF performs best here, and seeing that LLF performs better than the rest is also expected as the different print jobs now have different deadlines and the LLF algorithm can really shine. However, the difference between LLF, MMUF, and MUF is not significant as they all use the deadline at one point or another.

Average turnaround time

1. SJF with an average turnaround time of 16 hours and 52 minutes
2. MMUF with an average turnaround time of 21 hours and 20 minutes
3. Priority Queuing with an average turnaround time of 21 hours and 45 minutes
4. MUF with an average turnaround time of 21 hours and 46 minutes
5. FIFO with an average turnaround time of 21 hours and 47 minutes
6. LLF with an average turnaround time of 21 hours and 49 minutes

The algorithm that once again performs best is the SJF, which does not come as a surprise, however, what does come as a surprise is the difference or lack of difference for Priority Queuing, MUF, FIFO, and LLF the difference is insignificant.
As previously the algorithms' ranks will be summed to find the algorithm best performing.

Algorithm	Printing time	Print jobs missing deadline	Turnaround time	Total ranking
FIFO	1	3	5	9
SJF	6	1	1	8
Priority Queuing	3	6	3	12
LLF	4	2	6	12
MUF	2	5	4	11
MMUF	5	4	2	11

Table 5.5: Rankings of the algorithms with random deadlines for priority 2.

If the algorithm is to be chosen by the rankings alone, then SJF once again should be chosen, then FIFO. However, SJF and FIFO both face the challenge of dealing with print errors as there is no direct way of prioritising a failed print job over the queue that is already there. Thereby, FIFO and SJF are not seen as viable solutions. There are two algorithms tieing for the third place, these are MUF and MMUF. However, if the ranks for both deadline simulations are summed to a total ranking, and keeping in mind that the second simulation was run in order to verify which algorithm of either Priority Queuing or MMUF was to be used, the table would be:

Algorithm	Fixed Deadlines	Random Deadlines	Total Rank
FIFO	9	9	18
SJF	7	8	15
Priority Queuing	8	12	20
LLF	11	12	23
MUF	10	11	21
MMUF	8	11	19

Table 5.6: Total Ranking is a sum of the first simulation ranks and the second simulation ranks.

It can be seen from Table 5.6 that the first place goes to SJF, then FIFO and then MMUF. As MMUF gives the possibility to have priorities, which can be used for error handling of print jobs. As well as MMUF performed relatively well, this is chosen as the algorithm for the system.

6 Final Design

Now having investigated different algorithms for print job queuing, simulated them, and chosen the one best suited for the Printfarm Management System. The system design can be completed and a flowchart diagram can be made showing how the system should act:

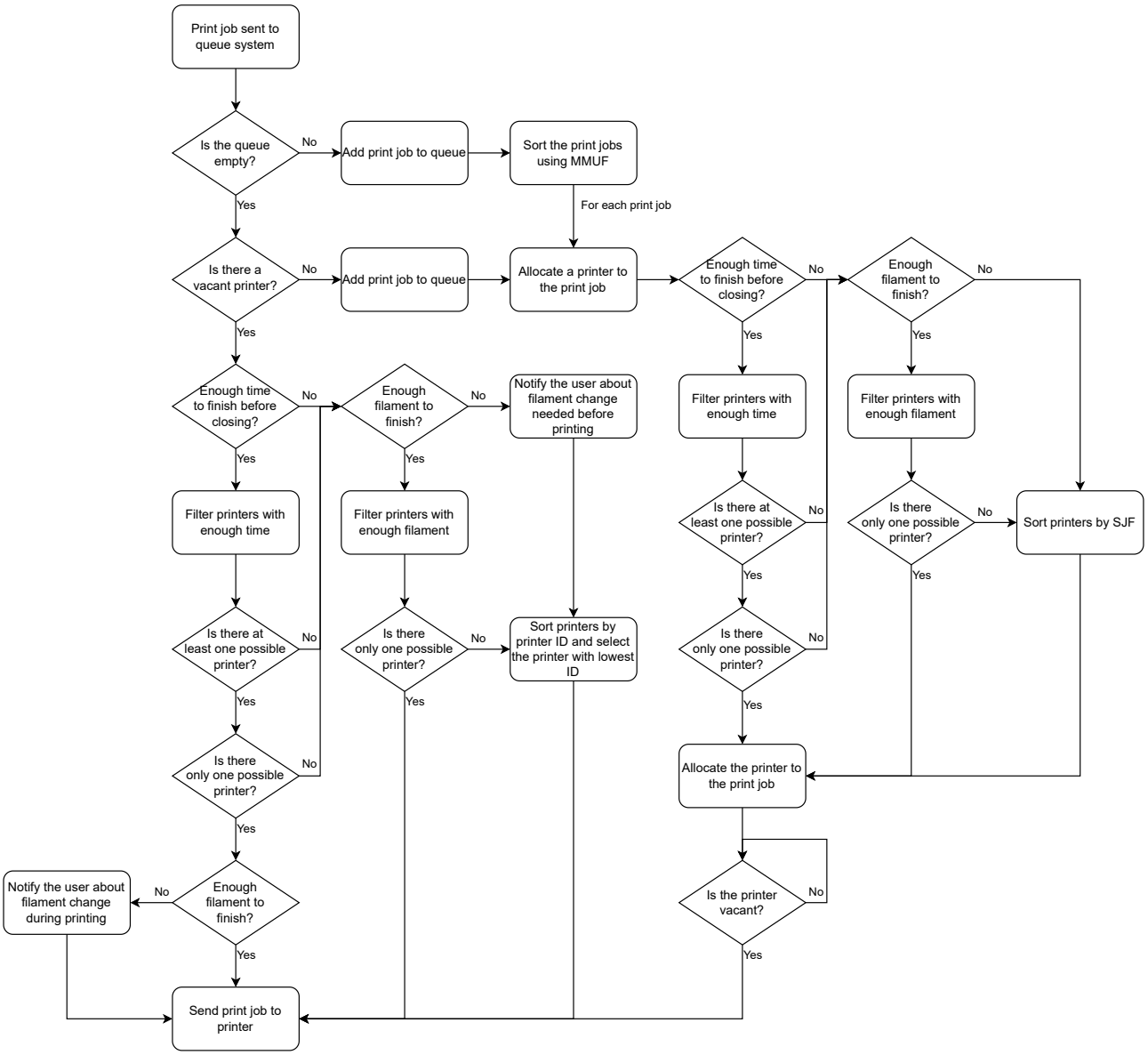


Figure 6.1: The figure shows a flowchart of how the queuing should act.

Seen from Figure 6.1, the first check is to verify if the queue is empty, as if the queue is empty there is no reason to sort the queue. Then the algorithm checks if there is a vacant printer, or if the print has to be added to the queue. The next check is to determine whether or not the print job can finish within working hours. This is important to know as the next check is to determine whether or not there is enough filament for the print job. If the print job can finish within working hours, but it is not expected that there would be enough filament to finish the print job, then the user or administrator could change the filament during printing. If, however, the print job cannot

finish printing within working hours and there is not enough filament to finish, then the user or administrator needs to change the filament before starting the print job.

The steps for when a print job finishes on a printer are relatively simple, and can be seen in the flowchart in Figure 6.2

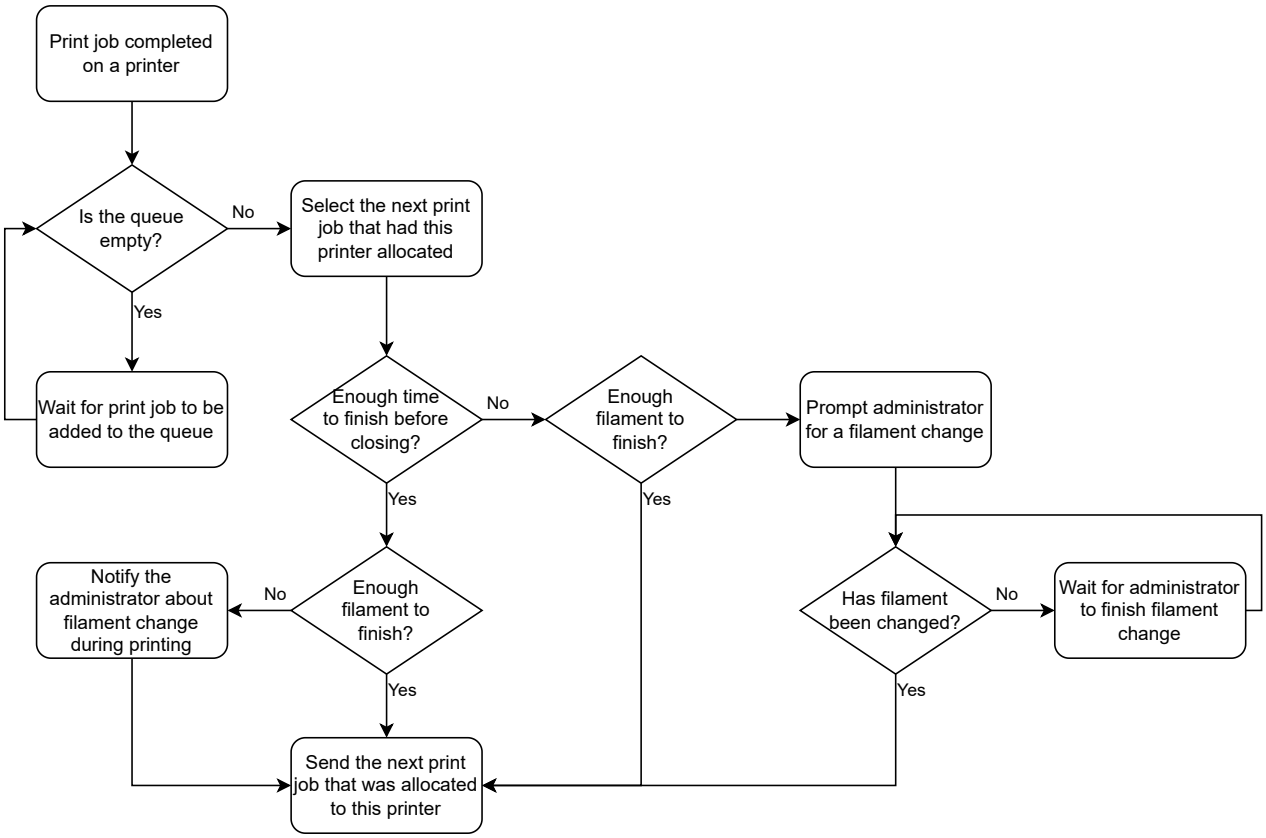


Figure 6.2: The figure shows a flowchart of how the Printfarm Management System is to act when a print job finishes on a printer.

As the system is to run task-driven, the algorithm needs to react when a print job is completed on a printer. If there are print jobs in the queue, then the print job that had this printer allocated will have to run through the same checks as in the previous flowchart. The first check is if there is enough time to finish the print job within working hours. Then if there is enough filament to finish the print job.

The design of MMUF used in the project will first be sorting the queue for criticality also called priority, then deadline, then employee ID, and then FIFO. A flowchart diagram of this process can be seen in Figure 6.3.

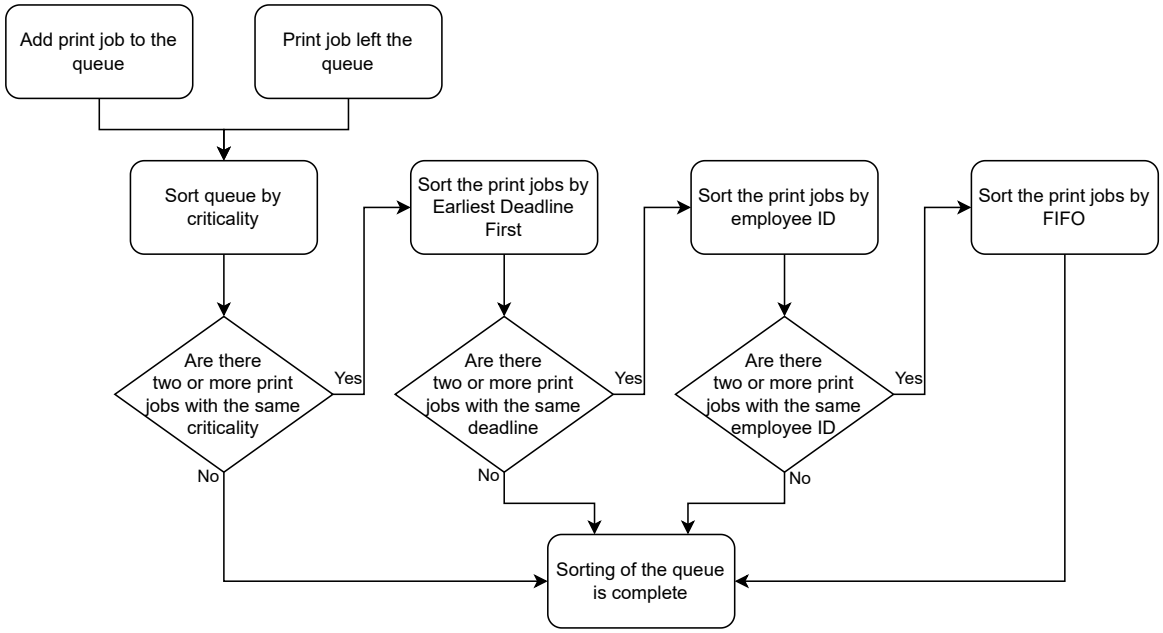


Figure 6.3: The figure shows a flowchart of how the sorting of the print job queue is to be carried out using MMUF.

The third step, where the queue is sorted depending on the employee ID of the print jobs, is chosen as it is expected that the chance of two print jobs having the same criticality, the same deadline, and being sent to the queuing system by the same employee is relatively low, but if it happens then these will be sorted using FIFO.

7 Discussion

Now having designed the Printfarm Management System it can be discussed whether or not the amount of simulations and the way they were set up is sufficient for taking the decision of queuing algorithm for the print jobs. Furthermore, it can be discussed if SJF actually would be the best queuing algorithm for the print jobs if there were more printers in the simulation. One of the main reasons why MMUF was chosen over SJF and FIFO was the possibility of having different priorities. These priorities could be used for standard customers and premium customers, but more importantly, could be used if an error happened during printing. So in order to better handle print errors, the only algorithms that were actually up for discussion were Priority Queuing, LLF, MUF and MMUF.
Prioritisation is important for handling printing errors, as a printing job that has failed on a printer needs to be restarted as soon as possible on another printer. This can be done by adding the print job to the queue again, but with a priority that ensures that no ordinary print job is being started before this print job is started. Another way to attempt this could be to add the print job to the queue again but with a laxity equal to 0. Thereby, LLF could start this print job again as one of the next print jobs. However, there is no guarantee that the print job will be prioritised over regular print jobs as these can also have a laxity of 0. If that is the case then the failed print job would actually be prioritised worse than the regular print jobs.
Another algorithm that could have been tested in the second run of simulations would have been EDF. However, as EDF does not take priority into account this was chosen not to be simulated. One could, however, implement error handling while still using EDF, by simply queuing the failed print job again, but with a deadline of 0 or the time when the print job was added to the queue.

When discussing error handling by either priorities or by deadline = 0, then it is important to note that this must be automated, in order to mitigate user errors. One way to mitigate user errors could be to ensure that the user cannot assign the same values as a print error would have and only allow the printers or the Printfarm Management System to allocate these values to a print job.

A user error that has not been handled in the thesis is if a user physically plugs in a USB drive and starts a print on a printer instead of using the Printfarm Management System. This error would cause the system to have to sort the print jobs in the queue again, and re-allocate print jobs to the printers that are available.
Looking at the initial problem formulation it can be discussed if a Printfarm Management System actually reduces the amount of manual labour compared to the current solution? In the current solution, it is possible to send a print job to the printer by WiFi, and the printer can automatically remove the print from the print plate. The improvements presented by the Printfarm Management System are that it is possible to queue up multiple print jobs causing the downtime of the printers to be lower as the downtime does not depend on the user interaction of sending another print job to the printer.
Furthermore, the Printfarm Management System is designed to notify the user about filament changes, which the user else would not know was needed unless the user physically inspected the printer and estimated whether or not there is enough filament for the print job. This, however, comes with the cost of changing out filament rolls that might have a significant amount of filament left. Thereby, the amount of manual labour of checking up on filament before sending the print job has been reduced.

As the simulations ran for the print job queuing algorithms did not calculate an estimate of the material usage of the print jobs, and did not consider working hours, it can be discussed if further simulations are needed in order to determine which algorithm is actually best suited for the system. However, those simulations would be closer to a simulation of the system as a whole than a simulation of the queuing algorithms. Furthermore, the difference in average turnaround time between MMUF and Priority Queuing in the second simulation case was 25 minutes, and the difference between Priority Queuing and LLF was only 4 minutes. It is with great confidence that MMUF was chosen as the algorithm for the print job queue. FIFO and SJF did perform better than MMUF in the simulations, but would not be able to satisfy the challenge of printing errors that would need to be reprinted.

When looking at the network communication of Create it REAL's 3D printers, it can be discussed whether or not mDNS is implemented optimally. One of the benefits of mDNS is that all the devices listening can update their DNS tables, which then could result in fewer retransmissions of the discovery process of devices. This could be done by having all the devices listening for the mDNS request also listen for responses, when a response is received, the device then stores the information. When the next request is sent the device can then respond with its DNS table, which includes information about itself and other devices. This could help in discovering devices that for one reason or another do not respond, but did respond on an earlier request. Thereby, the number of times a request is sent could be brought from four times down to possibly two times or even one time. However, this does come with the cost of memory on the devices, storing the DNS record, which at the time of writing is a sparse resource.

Furthermore, lowering the number of transmissions of the discovery request can possibly impact the amount of discovered devices for the first users sending their request, but as time goes on and the DNS records on the devices have been fully created, one request could possibly be enough to get the whole record as only one response is necessary from one of the devices in order to have the full record.

A possible way to utilise the capabilities of mDNS could be to have the computers on the network, that has REALvision open, listen for the request and send their DNS record as a response as well as the printers would respond, thereby, the computers could help each other and rely less on the printers to answer.

8 Conclusion

From this thesis it can be concluded that a Printfarm Management System has been designed to reduce the manual labour of running a 3D printer farm. The manual labour was reduced by designing a centralised queuing system, so there is no longer a need for manually plugging a storage device into the printers. The queuing system also gave the possibility of having the printers start print jobs outside working hours, potentially increasing the production with up to 3X the number of finished products.

Furthermore, it can be concluded that the design of the Printfarm Management System considers the estimated amount of filament left on each printer when selecting which printer each print job should be allocated to. The Printfarm Management System also considers the printing time of the print job, to determine whether a filament change is needed before starting the print job or if a change can happen during printing.
Moreover, MMUF has shown itself to be a favourable algorithm for sorting the print job queue, especially when looking at the average turnaround time of a print job. One of the key benefits of MMUF is the possibility of prioritisation, which can be used for error handling of print jobs.

It can also be concluded that by implementing the Printfarm Management System on a server and that being the only device communicating with the 3D printers, the amount of mDNS discovery requests for printers can be decreased dramatically as the Printfarm Management System can store the information about the 3D printers and would only need to make an mDNS discovery request if a printer was turned on and had not been on while the Printfarm Management System had been running. Furthermore, the computers on the network would only need to discover the Printfarm Management System, which should have a higher chance of responding than the 3D printers.

Future work could go into researching a way to ensure that a user cannot start a print job from a USB drive if there is a Printfarm Management System on the network. This must be investigated and implemented in a way such that the printers can still be used if the Printfarm Management System is nonoperational for a given amount of time.
Another direction further research could go would be looking into the firmware of the 3D printers and possibly finding another way for them to be discovered on the network, or to investigate whether or not the implementation of WiFi can be improved.

Acknowledgements

I would like to thank Create it REAL for the possibility of writing a thesis with them. It has been a learning experience in many fields both inside the topic of this master thesis, but also in other topics.

Furthermore, I would like to send a thank you to the first teacher believing in me, she saw a young boy who did not speak, read, or write any English and taught him how to believe in himself and taught him all the English he should have learned before she met him and what he should have learned that year. This teacher's name is Rohda Christensen.

Another person who shows a similar willingness to improve young people's lives is Henrik Schiøler. Henrik was willing to brainstorm the project proposal and support the idea of the project. He was there right when he was needed.

If more teachers and professors were like Rohda and Henrik, then the coming youth could go far in this world.

Bibliography

[1] N. Shahrubudin, T. Lee, and R. Ramlan, "An overview on 3d printing technology: Technological, materials, and applications," *Procedia Manufacturing*, vol. 35, pp. 1286–1296, 2019, The 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa, ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2019.06.089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978919308169.

[2] C. it REAL, *3d printer electronics*, Date visited 17-03-2023, 2022. [Online]. Available: https://www.createitreal.com/3d-printer-electronics/.

[3] C. it REAL, *3d printer slicer software - overview*, Date visited 26-04-2023, 2022. [Online]. Available: https://www.createitreal.com/3d-printer-slicer-software/.

[4] Creality, *Ender-3 s1 pro 3d printer*, Date visited 11-04-2023, 2023. [Online]. Available: https://www.creality.com/products/creality-ender-3-s1-pro-fdm-3d-printer?spm=..index.header_1.1.

[5] C. Thierauf, "Networking 3d printers with printfarmer," in *2018 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2018, pp. 1–4. DOI: 10.1109/URTC45901.2018.9244818.

[6] C. i. R. Lene Jensen, *Is 24-7 mass-customisation really possible with fdm print? -yes!* Date visited 11-04-2023, 2023. [Online]. Available: https://www.linkedin.com/posts/jensenlene_technology-3dprinting-activity-7042058820197457920-GUr-/?utm_source=share&utm_medium=member_ios.

[7] M. Radaviciute, "3d printing farm set-up: Printers and software," 2022.

[8] G. Häußge, *Octoprint*, Date visited 11-04-2023, 2023. [Online]. Available: https://octoprint.org/.

[9] P. L. S. Chris Hennes, *Octoprint-queue*, Date visited 11-04-2023, 2022. [Online]. Available: https://plugins.octoprint.org/plugins/queue/.

[10] R. Klauck and M. Kirsche, "Bonjour contiki: A case study of a dns-based discovery service for the internet of things," in *Ad-hoc, Mobile, and Wireless Networks*, X.-Y. Li, S. Papavassiliou, and S. Ruehrup, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 316–329, ISBN: 978-3-642-31638-8.

[11] A. Inc., *Ios security - ios 12.1*, Date visited 11-04-2023, 2018. [Online]. Available: https://web.archive.org/web/20200220214656/https://www.apple.com/chde/business/docs/site/iOS_Security_Guide.pdf.

[12] M. Stolikj, P. J. L. Cuijpers, J. J. Lukkien, and N. Buchina, "Context based service discovery in unmanaged networks using mdns/dns-sd," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, 2016, pp. 163–165. DOI: 10.1109/ICCE.2016.7430565.

[13] N. C. S. D. (CSD), *Fips 197, announcing the advanced encryption standard (aes)*, Date visited 19-04-2023, 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[14] J. Daemen and V. Rijmen, *The Design of Rijndael The Advanced Encryption Standard (AES)* (Information Security and Cryptography), eng, 2nd ed. 2020. Berlin, Heidelberg: Springer Berlin Heidelberg, ISBN: 3-662-60769-7.

[15] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, eng, 4rd edition. Pearson Education Limited, 2014, ISBN: 9781292061429.

[16] K. C. S. Murti, *Design principles for embedded systems* (Transactions on Computer Systems and Networks), eng. Singapore: Springer, 2022, ISBN: 981-16-3293-6.

[17] Stewart and Khosla, "Real-time scheduling of dynamically reconfigurable systems," in *IEEE 1991 International Conference on Systems Engineering*, 1991, pp. 139–142. DOI: 10.1109/ICSYSE.1991.161098.

[18] V. Salmani, S. Taghavi Zargar, and M. Naghibzadeh, "A modified maximum urgency first scheduling algorithm for real-time tasks," *Proc. Seventh World Enformatika Conference*, Jan. 2005.