UNIVERSITY OF LINCOLN

PACS Network Optimisation

Dissertation (BSc)

By Liam T. Berridge

BER14475946

April 2017

Abstract

The following technical report investigates the applied design science of computer networks, with the aim of establishing scalable worst-case efficiency formulas for small-world network topologies. These formulas may then be considered and utilised for future implementations of picture archival communication systems (PACS) networks to increase network performance, fault tolerance and overall quality of care.

Acknowledgements

I would like to take this opportunity to thank my friends, family and university peers for their continued support, guidance and companionship over the course of my academic career. It is unlikely I would have progressed to this stage without them and as such I owe them all a great deal.

I would also like to take the time to thank my parents directly, whom have provided all of the above for as long as I can recall, and none of this would have been possible without them.

Additional thanks go out to Scott Millar & Adam D. Walker for their assistance when understanding key concepts of mathematics, as well as Nick Jennett for ensuring this paper was checked for consistency and grammatical errors.

Author's Declaration

I declare that this paper and the associated artefact have been written and constructed by myself, and that all work except where specified otherwise in text is my own. This work has not been submitted for any other degree or qualification/s.

DATE: ____

NAME:____

SIGNATURE:___________________________________

Contents

List of Figures

List of Tables

Definitions

❏ EMR	-	Electronic Medical Record
❏ ER	-	Emergency Room
❏ FDD	-	Feature Driven Development
❏ FOSD	-	Feature-Oriented Software Development
❏ HPC	-	High Performance Cluster
❏ ICU	-	Intensive Care Unit
❏ IT	-	Information Technology
❏ LAN	-	Local Area Network
❏ MAC	-	Media Access Control
❏ NS2/3	-	Network Simulator 2/3
❏ NSF	-	National Science Foundation
❏ OR	-	Operating Room
❏ OSI	-	Open System Interconnection
❏ PACS	-	Picture Archival & Communication Systems
❏ RTT	-	Round-Trip Time
❏ TM	-	Traffic Matrices
❏ Topology	-	A network hierarchy or structure
❏ USB	-	Universal Serial Bus
❏ XP	-	Extreme Programming

Introduction

Due to the growing demand for modern medical services and care, medical facilities of all sizes are implementing advanced, often complex computer networks, with substantial demand for superior speed, bandwidth and inter-site communication.

As stated by Hirad (2010) “The field of Information Technology and Network Infrastructure Management has become crucial components within the healthcare industry.”

As a result of this, healthcare providers are investing significantly more in IT and communication networks; however it is still of critical importance that network implementations are designed to deliver superior stability and efficiency, whilst also taking advantage of the full range of benefits newer technologies offer, especially when considering costs for implementation and long term sustainability.

Problem statement

As health-care networks increase in size, the resources available require greater distribution to meet demand; When the demand is not met it results in a substantial decline, if not an absolute failure, in quality of care for patients. In turn this may also result in a decline in the general operation of, as well as the financial support available to, the medical facility.

In locations where radiology is a subsidised, separate site, this can also result in the effective “loss” of a department until the issue is resolved; This can be detrimental to quality of care as resources then rely on third-party means of transportation, such as physical delivery of radiology images or assets through USB pen drives or compact disks.

All of the above issues are compounded by the fact that these issues may not be department-specific, and may rely on network administrators to identify, track and isolate the problem as well as resolve it. This can take significant additional time, especially in complex, clustered networks with intricate topologies.

As a result of the above, medical facilities may experience delays or lapses in service, which may be critical in a performance oriented environment; Examples of these mission-critical environments are ER, OR and ICU services.

Hirad (2010, cited Keene & Auger, 2007) notes this also, as “The continuity of Business is vital in any setting and even more so in hospitals where the product is literally life and death”.

This is not to mention the financial implications of the above, where the repeated absence of required health-care services may substantially increase the cost to private and government funding sources. Most of these issues may be avoided with fault tolerance measures, such as backup networks and services, however various topologies also offer innate fault tolerance.

As noted by Hirad (2010) “Ultimately, rather than allocating private and government funding toward increased quality of patient care, hospitals are directing this much needed funding to their IT and administrative departments to resolve these problematic occurrences.”

Proposal

The aim of the research undertaken in this project is to produce a set of worst-case efficiency formulas, which may be used to calculate the efficiency of a topology given its scale.Whilst such formulas may be theoretically calculated, this paper also seeks to gain scientific proof that such formulas are accurate. The topology set used in this research is based on those discussed by Stewart (1992) in his paper “PACS Mini Refresher Course”, which discusses the use of basic network topologies in PACS medical networks, as well as identifying the benefits and drawbacks of each.

Each topology will be constructed using the NS-3 network simulator, an open-sourced network simulation software released under the GNU GPL_V2 license and supported by the NSF in the U.S. The simulation will feature an abstracted network simulation, focusing on the fifth through first layers of the OSI model as shown in *Figure 1*.

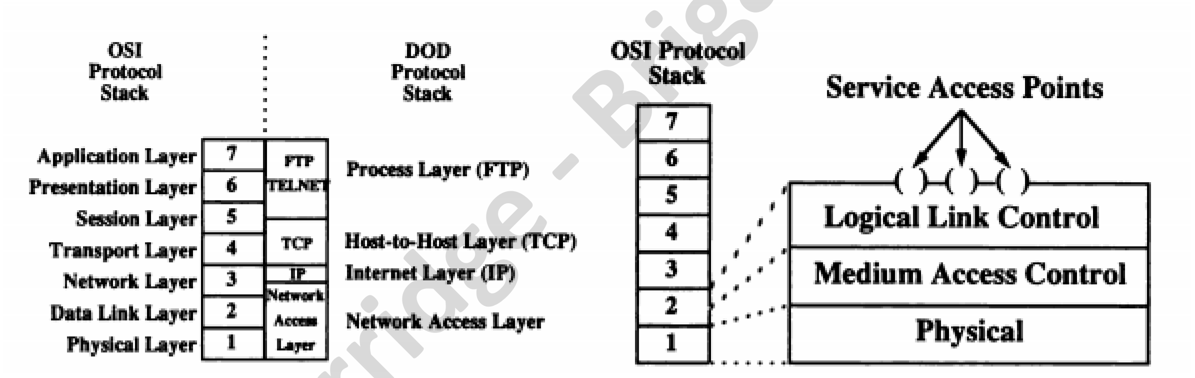


Figure 1 - OSI Model (Stewart, 1992)

Aim & Objectives

Aim:

The aim of this research is to investigate the worst-case efficiency of small-world PACS network topologies through empirical means. For each topology a single efficiency formula should be produced, where an efficiency can be determined through only the node requirement of the network.

Objectives:

1. Research an appropriate network simulation tool.
2. Create an accurate simulation of small-world topologies.
3. Use an appropriate means to simulate network communication between nodes.
4. Allow data transfer across network topologies using efficient routing algorithms.
5. For each topology, gain an accurate measure which may be used as a scientific proof for efficiency formula creation.
6. Compare the performance data gained, evaluating the topological benefits or limitations.

Background research

Whilst there have been various studies into the implementation of different topologies, most focus on a singular topology, with little analysis or reflection on why the proposed topology is inherently better than another. Most of these papers focus on larger multi-site networks also, such as the Dual-MAN proposed by Hirad (2010), but perform no in-depth analysis as to whether other topologies fare as well at larger scales. A study conducted by Gibbs & Quillen (2007) compares three different spatial scales of network, specifically NAS, MAN and WAN networks, however it only employs two topologies during testing, implementing tree and mesh networks and comparing the two. No wider set of topologies is tested or reviewed for use in larger networks, and no solid reasoning is given as to why.

The research presented by Ribeiro, Costa & Oliveira (2012) presents a different approach to handling PACS, using a cluster based peer-to-peer network. The paper suggests that by streaming a DICOM file from several machines at once the fault tolerance and access speed both increase exponentially. Whilst it does not focus on the topology of PACS networks, the suggestion that DICOM files may be streamed from and to several locations asynchronously presents an interesting alternative to creating a high-speed, high-availability network.

Literature review

To provide a further insight into the field and to also provide a comparison for the research performed, previous research into medical network systems will be utilised; additionally the past research will also aid in accentuating the benefits and drawbacks of implementing specific topologies.

Past research

An article written by Faggioni et al. (2011) suggests the following:
"*A key factor to fully harness the benefits of PACS is its integration[sic] with other systems and workflows already present in the hospital.*"
The statement delivered directly addresses that PACS systems are currently underutilised, often failing to integrate with the wider workflow of medical systems in place. As a product of this, PACS systems not only affect workflow, but also the medical networks they are attached to; This is also due to the excessive filesize many DICOM images have, requiring "*4-20 GB of storage space*" (Singh et al, 2010) for a comprehensive scan with multiple slices. When this is compounded by multiple scans for multiple patients, the data storage and network bandwidth requirements quickly balloon into terabytes of space, and delivering this information in a timely manner over a network becomes a challenge, regardless of additional traffic present on many medical networks. This shows why simply connecting existing implementations of PACS is unfeasible, as networks would be crippled by the additional load of transporting multiple DICOM files at any given time.

Due to the storage and bandwidth requirements, many PACS systems act independently of one another, with limited or no access to DICOM files created at distant locations save for physical transportation I.E portable hard-disk drives, USB pen-drives and compact disks. This method of transport is slow, requires a third party carrier to transport the data medium, and is also insecure as data has a physical presence and can be lost, stolen or damaged in transit. Faggioni (2011) again addresses this issue *"Unfortunately, many current PACS systems exist like independent islands outside the information stream of healthcare organisations, and in most cases such systems simply enable PACS workstations[sic] to receive, send, and backup DICOM files* *manually*". These implementations of PACS are not sustainable financially or operationally, as the delay in accessing clinical data may have severe repercussions on the quality of health-care for a patient.

The above is confirmed and also conflicted by a statement from Langer (2009) "*Further, the enterprise is frustrated financially because it is paying for many small to mid-size archives, with separate support contracts, rather than enjoying the economies of scale that one archive could provide, with central image management*.", which suggests that by creating an interconnected, centralised archive, the financial costs can be significantly reduced. Here the solution offered differs, whereas Langer opts for a centralised system, which would substantially reduce network overhead costs, it would require a higher implementation cost; and would also require additional fault tolerance systems, formal training for support teams regarding security, as well as disaster recovery in the extreme case the site was compromised. By creating a decentralised PACS network many existing implementations may simply be modified, and the redundancy and fault tolerance of such a network is greatly increased as each PACS archive acts independently, without reliance on any singular archive.

Faggioni et al go on to explain that although vertical integration between localised departments has largely taken place, horizontal integration between sites has not. This lapse in communication between sites can be seen as the largest single contributing factor to a lack of synergy between departments, as well as between sites on both a regional and national scale. By bridging this gap facilities would be able to quickly and effectively give the proper care to patients, reducing the time required for each patient, as well as the associated costs. *Figure 2* and its supporting statement, taken from Faggioni et al's (2011) paper, helps visualise the concept of vertical and horizontal integration.

"*Today, the goal is no longer just to give non-radiological departments access to radiological images ('vertical' integration, i.e. integration at the hospital level across specialties), but to ensure that imaging data is readily available and usable at every possible location within the affinity domain ('horizontal' integration, i.e. integration at the territorial level across healthcare services, including affiliated standalone clinics, referring physician offices, and other structures of the National Health System)*"

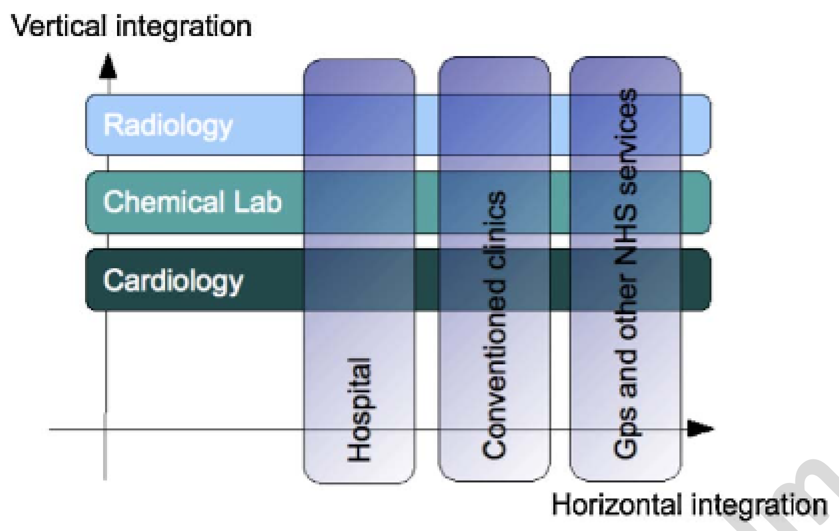


Figure 2 - Vertical & Horizontal Integration of PACS (Faggioni et al. 2011)

Faggioni et al go on to propose the implementation of an EMR, which collates patient data from all departments into a single, consolidated medical file. If implemented this file could be distributed and accessed as required, with the ability to stream only the required data from any given EMR, reducing network resource cost as well as the access time.

"*The ultimate goal behind merging imaging data with other med-ical information from non-radiological specialties is to create an entirely Electronic Medical Record (EMR) for every patient that should be capable of being transmitted anywhere.*" (Faggioni, et al. 2011)

Such a medium, if implemented, would allow for an increase in the quality of patient-care, as patient diagnosis & prognosis can be quickly performed, as well as lowered operating costs for medical financiers due to the increase in care quality and reduced operational time. When considering the reduced network requirements, increased fault tolerance and care quality, as well as the financial benefits, the implementation of EMR becomes crucial to the sustainability of PACS systems. Research conducted by Jorwekar, Dandekar & Baviskar has also shown that PACS systems are a necessary step forward for radiology departments, as productivity of clinicians increased substantially whilst using PACS; the research also shows it as a more robust system than film-based imaging. "*Out of 100 users, 85 % users reported PACS easy to very easy for handle. Ninety-four percent users reported PACS as useful tool for the hospital. Fifty-one percent of users found image quality at image review workstation to be excellent. Out* *of 100 users, 73 % found availability of radiological reports along with images as very useful tool.*" (Jorwekar, Dandekar & Baviskar, 2015)

Ribeiro suggests that implementation of a cluster-based storage is already feasible on most current PACS installations, and that with minor modification to the distribution mechanics of the PACS and DICOM systems in place, it would be plausible to mimic other peer-to-peer distribution technologies. "*A distributed system, besides eliminating single points of failure, also reduces bottlenecks in the work-flow by distributing the workload among the several machines enabling performance improvements.*" Ribeiro, Costa & Oliveira (2012) This further acknowledges the sustainability of a decentralised PACS archival system, and the relative ease of implementation compared to a centralised PACS archive which would require an entirely new data center, along with all required assets to run and maintain one.

When considering a centralised archive against a decentralised one, it becomes apparent that a decentralised archive is a much minor operation, with many PACS systems already being in a position to be adapted for such a network configuration.This, along with the previously identified benefits, makes creation of a decentralised network much more preferable. This also makes the research performed in this report valid for future use if a decentralised network were to be implemented, as a decentralised network would likely cover a wide range of topologies, making regular use of small-world topologies for each individual location.

The paper "*Measuring and Understanding Throughput of Network Topologies*" (Jyothi et al, 2014) investigates the performance of network topologies observed on large scale networks, such as DCell, Fat Tree and Hyperx. These topologies are generally found in data-centers and HPC networks however, so the research is limited in comparability. The methodology for testing these topologies is solid, but the research is limited in scope due to the specific set of topologies tested. In the aforementioned paper, Jyothi et al (2014) attempt to build a robust benchmark for network performance "*Our goal is to build a framework for accurate and consistent measurement of the throughput of network topologies, and use this framework to benchmark proposed data center and HPC topologies.*"; however to do so they require a quantifiable measure. In their paper they measure throughput, meaning traffic can be simulated and accounted for with varying levels of accuracy. This however may be viewed as skewing the result set, as even controlled traffic may affect the packet routing decisions made during the experiment. Whilst this is similar to the research proposed in this paper, this paper differs by taking only RTT measurements, without the need to route around traffic in the proposed topologies.

To measure the worst-case efficiency of a topology, Jyothi et al suggest the most effective method is through observing TMs and their throughput over a timed period. Gibbs & Quillen (2014) give the following definition *"A network's traffic matrix is a description, measure, or estimation of the aggregated traffic flows that enter, traverse, and leave a network.*". This is a common measure of networks, often gained through packet monitoring in real networks. In an experiment however it is difficult to accurately recreate network traffic as it is extremely volatile, with traffic flow varying rapidly during operational hours. This leads to Jyothi et al's (2014) conclusion of using worst-case throughput to measure efficiency "*develop a heuristic to measure worst-case throughput"* as well as to *"provide an expansive benchmarking of a variety of topologies using a variety of TMs*"; again however Jyothi uses throughput to develop an efficiency measure, and whilst with accurate traffic recreation this is an excellent measure, developing accurate traffic simulations for scaling medical networks is unfeasible without excessive data collection and analysis over sustained periods of time.

This identifies the major differences in Jyothi et al's research and that presented in this paper, both the topologies tested and the metric used to measure efficiency, where Jyothi et al's are suited for a centralised archive, this paper chooses topologies present in localised PACS implementations as specified by Stewart (1992), with sight to implementing a decentralised PACS archive. Jyothi et al employ a longest-matching TM heuristic, based on the worst-case performance of a network. This metric is almost identical to the one used in this paper, however the difference between using throughput and RTT should create substantially differing results. The research undertaken in this paper will use a generalised network efficiency metric similar to that created by Latora & Marchiori (2008). It will count the hops required to cross a network topology in a worst-case scenario, I.E where the data must travel the largest distance a network offers. Using the data gathered from each topology we can then test and attempt to prove a theorised formula used to estimate efficiency at any given scale.

Software Development Life Cycle

Introduction

The following section of this report will cover project management, software development methodologies and applied principles, as well as the design, development and evaluation of the artefact itself.

The artefact created for this project takes the form of a set of simulations, each programmed in the C++ language and compiled/run inside the NS3 simulation suite. Whilst the project initially took place in NS2, it became evident that use of the software was not sustainable for prolonged development, and the change to NS3 took place. The official description for NS3 is as follows:

"*ns-3 has been developed to provide an open, extensible network simulation platform, for networking research and education. In brief, ns-3 provides models of how packet data networks work and perform, and provides a simulation engine for users to conduct simulation experiments.*" (About ns-3, 2010)

For each topology a smaller sub-set of simulations is created, with each establishing a different node count based on the requirements analysis. From these subsets a set of ratio data was produced, recording the time taken to traverse a topology at each size using a worst-case efficiency metric of travel. The topologies were sourced from Stewart's (1992) paper, which suggests that localised PACS networks commonly employ P2P, Ring, Star and Tree networks, amongst others.

Methodology

Project Management

The full project was originally planned to run from October 2016 until April 2017, with the original development timeline (*Figure 3*) following an iterative methodology. This later changed to an agile methodology as the development path altered to ensure the requirements were met in the remaining timeframe.

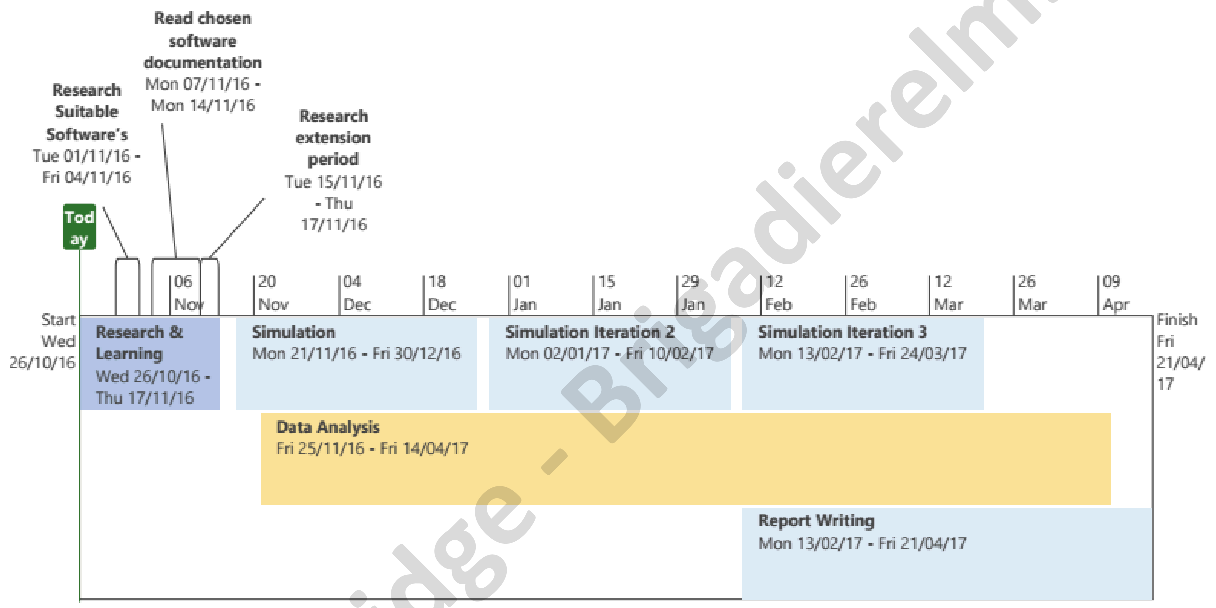


Figure 3 - Project Plan (Liam Berridge, 2016)

The original plan shows three absolute stages, each being defined as an iteration with a secondary goal of data analysis and feedback, this mimics Larman's (2003) definition of iterative software development "*Each iteration is a self-contained mini-project composed of activities such as requirements analysis, design, programming and test.*"

Due to issues with the older NS2 simulation suite, the decision was made in early February 2017 to change to NS3; this change was critical to the project's success as various components of NS2 did not function on newer operating systems. This change set the project back an entire iterative cycle however and resulted in losing valuable time, as although the concepts and requirements were transferable, the work completed was not due to the differences in programming language. This prompted a change in the methodology used to ensure the deadline was still met, and that the artefact produced still fulfilled the requirement outline (*Figure 4*).

- ❏ Create a series of basic simulations, each implementing and demonstrating a different topology
    - ❏ Peer-To-Peer
    - ❏ Ring
    - ❏ Grid
    - ❏ Tree
- ❏ Modify the existing topologies, creating a topology which may be incremented in size with little change to the simulation codes underlying structure
    - ❏ 8 Nodes
    - ❏ 16 Nodes
    - ❏ 32 Nodes
    - ❏ 64 Nodes
    - ❏ 128 Nodes
    - ❏ 256 Nodes
- ❏ Create applications on each topology to enable sending/receiving of TCP/UDP packets.
- ❏ Transfer data across the network using a worst-case efficiency, monitoring the performance time of each.

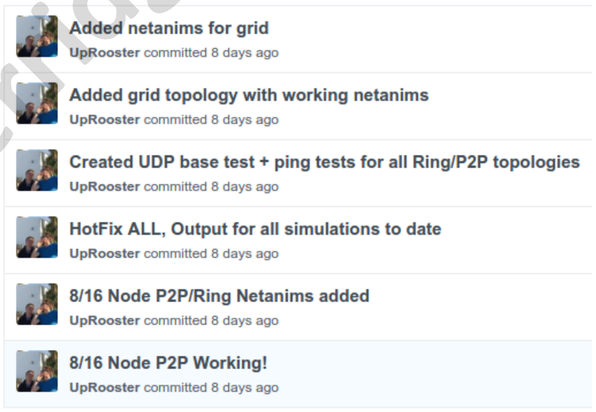
Figure 4 - Artefact Requirements Outline (Liam T. Berridge, 2017)

The decision to choose an agile methodology after this change was made due to the main principles agile methodologies possess. Agile methodologies as defined by Larman (2003) "*In addition, they (Agile methodologies)[Sic] promote practices and principles that reflect an agile sensibility of simplicity, lightness, communication, self-directed teams, programming over documenting, and more.*", and as defined by Lindstrom & Jeffries (2006) "*Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely[sic].*" hold ideal principles in the context of rapid development and deployment, where code development is held priority over procedure and documentation; these principles were also necessary in the context of this project, and so were chosen following the change in supporting tools.

Software Development

During the course of the project, various software development practices were employed; namely those from the agile manifesto such as XP and FDD. These practices were recurring throughout development and assisted greatly in ensuring deadlines were met. XP focuses on "*Collaboration, quick and early software creation, and skillful development practices*" (Larman, 2003), all of which but the first were applied during development sprints. FDD or FOSD differs from XP in the way of focusing on individual features as opposed to working systems; endeavouring to release regular builds once feature-sets are completed. Abrahamsson et al (2003) describes FDD as "*Feature-driven development (FDD) [21, 22] is a process-oriented software development method for developing business critical systems. The FDD approach focuses on the design and building phases.*"

As demonstrated in *Figure 5* the project was rapidly developed, with each programming session attempting to implement topologies as succinctly as possible, and with later sessions then working on individual features and the presentation of the code. This allowed for a combination of both of the aforementioned methodology practices into a single session as well as being used interchangeably across the development life cycle. This kind of practice was consistent throughout the project, creating a rapid and concise workflow.


Added netanims for grid
UpRooster committed 8 days ago

Added grid topology with working netanims
UpRooster committed 8 days ago

Created UDP base test + ping tests for all Ring/P2P topologies
UpRooster committed 8 days ago

HotFix ALL, Output for all simulations to date
UpRooster committed 8 days ago

8/16 Node P2P/Ring Netanims added
UpRooster committed 8 days ago

8/16 Node P2P Working!
UpRooster committed 8 days ago

Figure 5 - GitHub Commit Logs (Liam T. Berridge, 2017)

The requirements outlined also played a large role in determining the methodology used, as the requirements and milestones for development were small, meaning excessive planning and design elements were not required as often as they are in other methodologies. When considering which practice to use, the following quote from Lindstrom & Jeffries (2006) fit in line with those requirements well: "*Whereas many popular methodologies try to answer the question "What are all of the practices I might ever need on a software project?," XP simply asks, "What is the simplest set of practices I could possibly need and what do I need to do to limit my needs to those practices?"*". The requirements outlined are simplistic, with clearly defined goals and a lack of ambiguity; and whilst this was not done specifically to increase development speed, it aided in doing so when combined with XP and FDD practices.

Looking at a sample of code from an older iteration of the artefact (*Figure 7*), it becomes apparent that the focus was placed onto functionality within the program, as well as ensuring that the code implemented all aspects of a topology before being committed into the working repository.

```
std::vector<NodeContainer> subnetList (nNodes);
  NS_LOG_UNCOND ("Creating Subnets");
  for(uint32_t i=0; i<subnetList.size()-1; ++i)
    {
      //NS_LOG_UNCOND ("Creating Subnet " << i);
      subnetList[i] = NodeContainer (Nodes.Get(i), Nodes.Get(i+1));
    }
  uint16_t NSize =  subnetList.size();
  NS_LOG_UNCOND ("Creating Devices");
  std::vector<NetDeviceContainer> deviceList (NSize);
  std::vector<Ipv4InterfaceContainer> subNetInterfaces (NSize);
  for(uint32_t i=0; i<deviceList.size()-1; ++i)
    {
      subnetAddr.str("");
      deviceList[i] = p2p.Install (subnetList[i]);
      subnetAddr <<"10.1."<<i+1<<".0";
      //NS_LOG_UNCOND ("Creating Address " << subnetAddr.str().c_str ());
      address.SetBase(subnetAddr.str().c_str (),"255.255.255.0");
      subNetInterfaces[i] = address.Assign(deviceList[i]);
    }
```

Figure 6 - An Older Version of the Artefact (Liam T. Berridge, 2017)

To further demonstrate how these practices were put to use, *Figure 8* shows the entire project's development cycle, with the graph increasing with each addition made to the artefact. The lapse in the timeline is due to external circumstances, and should not be taken into account when considering the effectiveness of the methodology and practices applied.

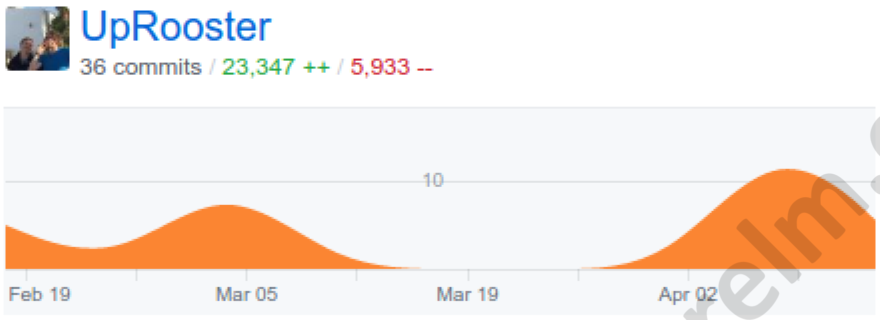


Figure 7 - Artefact Creation Timeline (Liam T. Berridge, 2017)

When only considering the two active stages of development shown in the timeline, the progression for the artefact is almost linear, with development maintaining a constant rate of updates. Constant, sustainable development are also principles in the agile manifesto, as identified by Lindstrom & Jeffries (2008), further increasing the tie between the artefacts development path and the agile methodology. The excerpt identifying the agile manifesto principles is as follows:
"*Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely[sic].*". This is also succeeded by the following statements, classifying further principles of the manifesto "*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to shorter time scales." and "Working software is the primary measure of progress*.".

Research Methods

The data gathered from each simulation is the RTT, or the time taken for a packet of pre-defined size to travel to the furthest accessible node, and to return to the sender using worst-case efficiency. For each simulation only a single, qualitative time measure is required as due to the nature of the artefact any anomalies should be easily discernable; once gathered these times are used to validate the worst-case efficiency formula proposed for the topology.

Whilst Jyothi et al (2016) use throughput to gain a measure of network efficiency, they also attempt to simulate traffic, which contributes to and may determine the throughput of the network. Without the requirement for traffic simulation, this measure is not useful for the research performed in this paper, and thusly is not used for comparison of results. To gain an efficiency measure without traffic, we instead use a similar metric to the work put forth by Langer, French and Segovis (2011), which measures efficiency of transfer time through the following formula: $Transfer\ Time\ =\ Number\ of\ Packets\ x\ RTT$. This formula is originally used for sending bulk data across large physical networks, however it can be adapted for use with measuring network efficiency with little to no modification.

The RTT gathered for each simulation can be analysed and compared against the expected results hypothesised for each topology; also making the data measurement a ratio data measurement type, as time measurements offer an absolute zero and may be multiplied, divided and otherwise modified for further, in-depth analysis.

Design, Development & Evaluation

Requirements & Analysis

The requirements for the artefact were collected by taking the Aim & Objectives outlined previously, and ensuring each was resolved thoroughly by the artefact. The first objective was to create a set of basic simulations based on some of the topologies discussed by Stewart (1992) "*The four basic topologies are the bus, tree, ring, and star;*". Using the NS3 simulation suite, each of these topologies was viable for implementation through physical or wireless networks.

Due to the age of the report wireless connections are only just being investigated for implementation, however each of the topologies investigated in this report are transferable between LAN and wireless connections without alteration of the delivery times to a substantial degree; further to this, BUS topologies were not viable for implementation, as although NS3 accounts for CSMA type connections, it only uses a single delay for the entire channel, meaning that value becomes the worst-case delivery time for a packet. To circumvent this issue, P2P type connections were instead implemented, with each pairing of nodes utilising a separate connection with its own attributes.

The remaining topologies from Stewart's paper are Tree, Ring and Star. Tree and Ring were implemented, whilst Star was substituted for a Grid type topology. The reasoning behind this was due to Grid topologies presenting a consistently measurable efficiency metric, whereas stars topologies vary based on the amount of nodes each star possesses.

Each of these topologies once implemented were then tested for worst-case efficiency, using the RTT of packets on the topology to backup the theorised worst-case efficiency formulas outlined in the design section of this report.

To accurately measure the RTT of packets however, each network must be accurate to a real network, simulating the layers present in all physical networks and as demonstrated in the OSI model; such as link control, medium access control and physical layers. (Stewart, 1992). This created the requirements for all simulations to follow the steps outlined below (*Figure 8*).

- ❏ Command-line input parsing
- ❏ Node Creation
    - ❏ Stack Instantiation
- ❏ MAC Protocol Function
    - ❏ MAC Protocol Assignment
- ❏ IPv4 Address Function
- ❏ Channel Creation
    - ❏ Varies based on topology, iterates through all nodes
- ❏ Net Device & Interface Container Creation
- ❏ Subnet Creation
    - ❏ For each node in each channel
        - ❏ Assign a net device
            - ❏ Assign a MAC protocol
        - ❏ Assign an interface
            - ❏ Assign an IPv4 address
- ❏ Create applications on nodes (As determined by a worst-case efficiency)
    - ❏ Create a client on Node #0
    - ❏ Set a client target on Node #N-1
    - ❏ Create a server or packet sink on Node #N-1

Figure 8 - Simulation Requirements (Liam T. Berridge, 2017)

These requirements ensure that all networks created for the artefact follow the OSI model outlined by Stewart (1992), creating a realistic simulation of PACS networks which encompass all aspects of a real network. These requirements also define the applications, which are used to test the theories proposed in this paper, they do so by sending packets back and forth with timestamps as to their sent/received time; this data can then be collated and used in comparison against the theorised efficiencies, proving or disproving the formulas proposed below.

Design

For the design of the artefact, a basic diagram was created of each topology. Each diagram was then used as a visual aid in theorising the worst-case efficiency of a network based on it's 'hops', or the number of nodes required to cross the furthest possible distance, or worst-case efficiency, on the topology. For the theoretical worst-case efficiency, increments for each topology were based on either 8, 16 or 32 nodes, as these three measures alone were enough to worst-case efficiency, however to prove these formulas, 64, 128 and 256 nodes were also included in the actual implementation; this was to provide additional evidencing for each formula. *Figure 9* shows the diagrams created, along with the theorised worst-case efficiency. Beyond the figure a small breakdown of each topology is shown, explaining how the theoretical worst-case efficiency was reached.

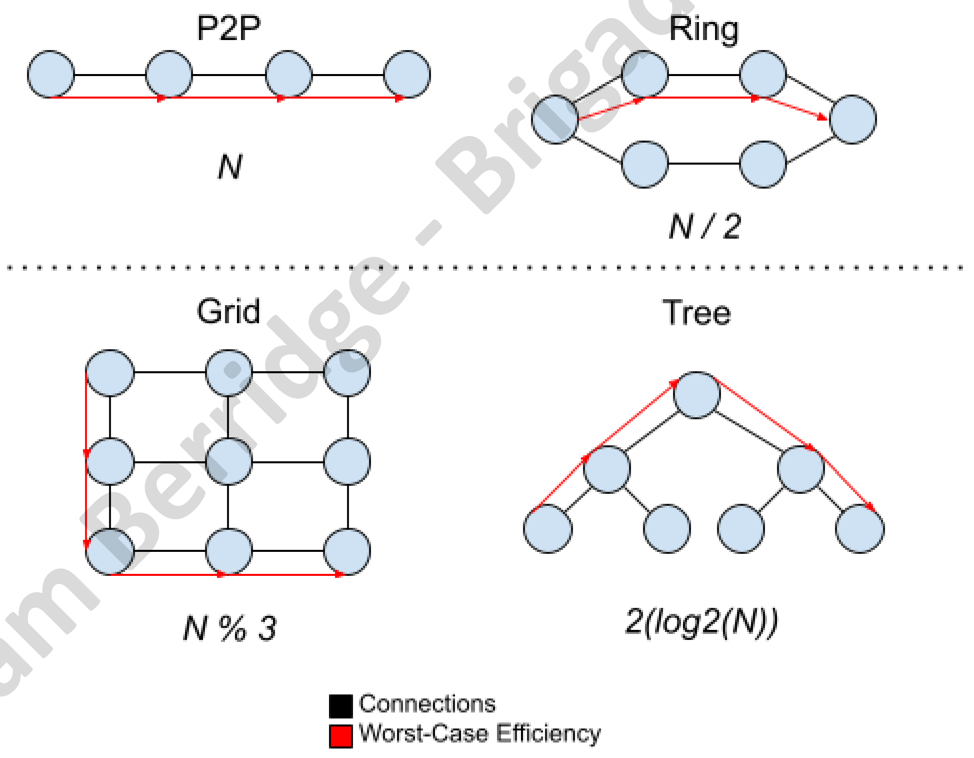


Figure 9 - Theoretical Worst-Case Efficiencies (Liam T. Berridge, 2017)

Theoretical Worst-Case Efficiencies

Peer To Peer

For peer-to-peer topologies, the worst-case efficiency is simply the node count, as in a worst-case scenario the packet would have to travel from the first node to the last. Therefore the formula theorised for P2P topologies is simply:

$$x = N$$

Where N represents the nodes specified in the topology

Ring

For ring topologies, the worst-case efficiency is the node count divided by two, as the topology is the same as P2P except the first and last node are connected, meaning the furthest possible distance is half of the full node count. The formula for such is:

$$x = N / 2$$

Where N represents the nodes specified in the topology

Grid

For grid topologies, the worst-case efficiency is the node count modulus three, as the nodes are distributed evenly across a two-dimensional grid. The formula is as follows:

$$x = N \% 3$$

Where N represents the nodes specified in the topology

Tree

For tree topologies, the worst-case efficiency is the node count multiplied by a logarithm of base two, multiplied by a further two. This is due to how tree structures work, the logarithm calculates how many steps to traverse up to the root, and this is multiplied by two as we postulate the tree to be evenly distributed, and the worst-case efficiency then travels down another path of the same length. The formula for this is as follows:

$$x = 2(log_2(N))$$

Where N represents the nodes specified in the topology

Once the topologies worst-case efficiency formulas had been theorised, the next step was to ensure the requirements for each simulation were addressed. To do so the underlying simulation elements are created in a unified sense, meaning across simulations the definitions for elements stay the same. This allows for a more universal understanding of the simulations created for the artefact, and further assists when resolving bugs encountered in the simulation, as elements may be checked against working counterparts in other topologies.

Simulation Pseudo-Code

The following pseudo-code (*Figure 10*) was designed to illustrate the basic concept for each simulation, basing the pseudo-code on existing NS3 functions.

```
$N = InputVariable$
$Nodes.Create(N)$
$Stack.Install(N)$
$P2PHelper\ P2P$
$for(N/2)\ \{Channel[N].Create(TOPOLOGY\ DEPENDANT)\}$
$for(Channel[N] - 1)\{NetDevice.Create(Channel[N]),$
        $Interface.Assign("10.0.[N].0")\}$
$ApplicationClient.Create(Nodes.Get(0))$
$ClientTarget.Set(Nodes.Get(0))$
$ApplicationServer.Create(Nodes.Get(N - 1))$
$Simulation.Run()$
$Animation\ Anim = Anim.xml$
```

Figure 10 - Simulation PseudoCode (Liam T. Berridge, 2017)

This pseudo-code was used as the basis for the underlying simulation, with little change occurring over the course of the development cycle; this contrasts with the noticeable changes visible in the implementation for each topologies channel structuring. These changes are discussed below, with the pseudo code for each being shown to visualise the changes.

Peer To Peer

For peer to peer topologies, the connection type is simple, with each node connecting to the next in the node container until the last node has been added to a channel. This was easily implemented using a single for loop which iterates through the channel node container object.

```
$for(i < Channel.Size - 1)\{$
    $Channel[i] = Nodes.Get(i), Nodes.Get(i + 1)\}$
```

Figure 11 - Peer-to-Peer Pseudocode (Liam T. Berridge, 2017)

Ring

The ring topology re-implements the approach taken by the Peer-to-Peer topology, connecting the first and last node in the node container to loop the connection around fully. The pseudo-code for this is below, and is noticeably similar to the above pseudo-code.

```
$for(i < Channel.Size - 1)\{$
    $if(Channel[i] == Channel.Size - 1)\{$
        $Channel[i] = Nodes.Get(i), Nodes.Get(0)\}$
    $else\{$
        $Channel[i] = Nodes.Get(i), Nodes.Get(i + 1)\}$
```

Figure 12 - Ring Pseudocode (Liam T. Berridge, 2017)

Grid

For the grid topology, we take the same approach as the peer-to-peer topology again, but implement this over a matrix of horizontal and vertical arrays. We calculate the columns by getting the square root of the input, and the rows by calculating the ceiling of the input divided by the columns. The pseudo-code for this is below:

```
$cols = sqrt(N)$
$rows = ceil(N/cols)$
$for(x < rows.size)$
    $for(y < cols.size)$
        $rowN.Create(1)$
If the node is not the first in the column or row, create channels to previous nodes on both axis
        $if(x > 0)$
            $ChannelRows[x] = rowN.get(x - 1), rowN.get(x))$
        $if(y > 0)$
            $ChannelCols[y] = colN.get(y - 1), rowN.get(x))$
```

Figure 13 - Grid Pseudocode (Liam T. Berridge, 2017)

Tree

The tree topology is constructed by creating channels from the root down to the end-nodes (determined by input nodes), however the tree structure must be created upwards from the end-nodes first. Although this means the tree topology has more nodes overall, the only nodes used are those connecting the furthest distanced nodes. The Pseudocode for determining the support tree, and then the channel creation, is shown below:

Determine the number of levels required, as well as the size of each level.

```
$i = N/2$
$Levels = N/2; LevelSet = 1$
```

Logarithmically determine the tree from the base up to the penultimate level, the root of the tree must be manually created outside of the loop.

```
$while(i >= 2)\{$
    $Levels[LevelSet].Create(i)$
    $i = i/2$
    $LevelSet = LevelSet + 1$
$Levels[LevelSet].Create(1)$
```

Create the channels from the root to the base of the tree

```
$for(x = LevelSet, x > 0, x --)\{$
    $k = 0$
    $for(y < Levels[x])\{$
        $for(z < 2)\{$
            $Channel[j] = Levels[j].Get(x), Levels[j - 1].Get(k)$
```

Figure 14 - Tree Pseudocode (Liam T. Berridge, 2017)

Building & Coding

Once requirements had been outlined, and the design for each topology and the underlying simulation had been created, the next step was to create a basic template for each topology. This was to ensure the topologies were implementable as proposed in the design stage, as well as to ensure the NS3 suite was fully functional before exhausting further time and effort, avoiding a recurrence of the issues NS2 presented.

The topologies tested as templates were Point-To-Point, Mesh and Star. Point-To-Point topologies were the first to be tested, as the template only hosted two nodes, acting as a single channel or subnet between the two. This was then used as the basis for all channels present in other simulations. Mesh was the secondary simulation to be created and tested, ensuring that all nodes were capable of being interconnected without programming errors. Star topologies were the last to be tested, simply ensuring nodes were able to interconnect and create a larger network. All of these templates were also used to gain familiarity with the NS3 suites functions, classes and methods. *Figure 15* shows a segment of the first script created as a part of the artefact, testing the ability to create and connect two nodes with appropriate network characteristics such as those found in layers one and two of the OSI model. (Stewart, 1992)

```
NodeContainer NODES;
  NODES.Create(2);

  InternetStackHelper STACK;
  STACK.Install(NODES);

  PointToPointHelper P2P;

  Ipv4AddressHelper ADDRESS;
  ADDRESS.SetBase("10.1.1.0","255.255.255.0");

  NodeContainer SUBNET1;
  SUBNET1.Add(NODES.Get(0));
  SUBNET1.Add(NODES.Get(1));

  NetDeviceContainer SUBNET1DEVICES = P2P.Install(SUBNET1);

  Ipv4InterfaceContainer SUBNET1INTERFACES = ADDRESS.Assign(SUBNET1DEVICES);
```

Figure 15 - The First Script (Liam T. Berridge, 2017)

Once these basic topologies had been created, the next stage was creating each of the four chosen topologies, and ensuring that each was created in a manner which allowed minimal change to code and structure, but allowed variable input sizes for networks. Each of the four major topologies studied were created in the order of Point-To-Point, Ring, Grid and Tree, much the same order as they had been designed in.

The Point-To-Point simulation underwent several versions, the first of which contained the majority of the topologies functionality with following iterations focusing on the code structure and animation output. The first version of the topology is shown in *Figure 16*, and implements the core functionality for creating nodes, instantiating stacks on each node, as well as creating channels, devices and network interfaces; the code then ends by creating an application client/server, sending a UDP Echo packet and then outputting an animation XML file, with the nodes being placed at intervals defined by a for loop.

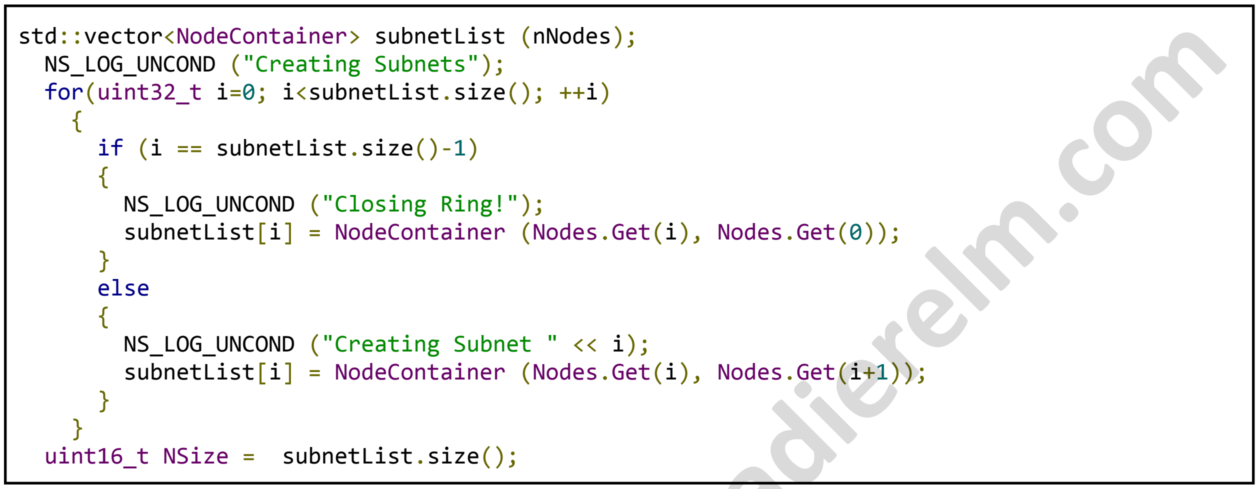
```
/*----------------SUBNET CREATION----------------*/
 std::vector<NodeContainer> subnetList (nNodes);
 NS_LOG_UNCOND ("Creating Subnets");
 for(uint32_t i=0; i<subnetList.size()-1; ++i)
   {
     //NS_LOG_UNCOND ("Creating Subnet " << i);
     subnetList[i] = NodeContainer (Nodes.Get(i), Nodes.Get(i+1));
   }
 uint16_t NSize =  subnetList.size();
 NS_LOG_UNCOND ("Creating Devices");
 /*----------------DEVICE CREATION----------------*/
 std::vector<NetDeviceContainer> deviceList (NSize);
 std::vector<Ipv4InterfaceContainer> subNetInterfaces (NSize);
 for(uint32_t i=0; i<deviceList.size()-1; ++i)
   {
     subnetAddr.str("");
     deviceList[i] = p2p.Install (subnetList[i]);
     subnetAddr <<"10.1."<<i+1<<".0";
     //NS_LOG_UNCOND ("Creating Address " << subnetAddr.str().c_str ());
     address.SetBase(subnetAddr.str().c_str (),"255.255.255.0");
     subNetInterfaces[i] = address.Assign(deviceList[i]);
   }
```
Figure 16 - Peer-To-Peer Channel Creation (Liam T. Berridge, 2017)

Figure 17 demonstrates the content of later changes, where functions were added to improve the codes structure and presentation, with no real change being made to the underlying functionality or content of the code. This is again in line with the principles of the agile manifesto, where working code is placed ahead of the documentation or presentation of the code.

```
std::vector<Ipv4InterfaceContainer> subnetCreation (std::vector<NetDeviceContainer>
deviceList, std::vector<NodeContainer> subnetList, std::vector<Ipv4InterfaceContainer>
subNetInterfaces, PointToPointHelper p2p)
{
  // CREATE IPV4 HELPER & IP STRING
  Ipv4AddressHelper address;
  std::ostringstream subnetAddr;
  for(uint32_t i=0; i<deviceList.size()-1; ++i)
  {
    subnetAddr.str("");
    deviceList[i] = p2p.Install (subnetList[i]);
    subnetAddr <<"10.1."<<i+1<<".0";
    //NS_LOG_UNCOND ("Creating Address " << subnetAddr.str().c_str ());
    address.SetBase(subnetAddr.str().c_str (),"255.255.255.0");
    subNetInterfaces[i] = address.Assign(deviceList[i]);
  }
  return subNetInterfaces;
}
```

Figure 17 - Peer-To-Peer Function Creation (Liam T. Berridge, 2017)

The Ring topology only required a minimal amount of effort, as its code is largely based on that of the Point-To-Point topology. *Figure 18* shows the small change required to change the Point-To-Point topology code to a Ring topology.



```
std::vector<NodeContainer> subnetList (nNodes);
  NS_LOG_UNCOND ("Creating Subnets");
  for(uint32_t i=0; i<subnetList.size(); ++i)
    {
      if (i == subnetList.size()-1)
      {
        NS_LOG_UNCOND ("Closing Ring!");
        subnetList[i] = NodeContainer (Nodes.Get(i), Nodes.Get(0));
      }
      else
      {
        NS_LOG_UNCOND ("Creating Subnet " << i);
        subnetList[i] = NodeContainer (Nodes.Get(i), Nodes.Get(i+1));
      }
    }
  uint16_t NSize =  subnetList.size();
```

Figure 18 - Peer-To-Peer Altered To Ring (Liam T. Berridge, 2017)

Similar to the Point-To-Point implementation, the only changes to take place after the initial creation of the simulation were concerned with the codes readability and structure, with improvements to programming principles applied being the only major change. Examples of this are present in *Figure 17*, where functions are created. Although the figure relates to another topology, the same functions were replicated and altered to fit the ring topology.

The grid topology differed from the original design, as NS3 already possessed a function to recursively create entire grid topologies given the grid dimensions and the MAC connection type. Again this script underwent minimal changes as the first iteration of the script implemented all required functionality, with changes made to improve code structure and readability, as well as to output the simulation animation file. With concern to the animation file, the grid topology helper class possessed pre-made functions which assisted in placing nodes for the animation file. *Figure 19* shows the application of the grid topology helper object as well as the assignment of IPv4 addresses, and *Figure 20* shows the use of the helper again when creating the animation file.

```
// Create grid dimensions
  uint32_t cols = (uint32_t)sqrt(nNodes);
  uint32_t rows = (uint32_t)ceil(nNodes/(double)cols);

  // Init p2p (p2p)
  PointToPointHelper p2p;

  // Create grid
  PointToPointGridHelper grid (rows,cols,p2p);

  // Init Stacks on grid nodes
  InternetStackHelper Stack;
  grid.InstallStack(Stack);

  // Init Ipv4 + Address String
  Ipv4AddressHelper address;

  NS_LOG_UNCOND ("Creating Subnet List");
  // Assign Addresses to Grid
  grid.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"),
                            Ipv4AddressHelper ("10.2.1.0", "255.255.255.0"));
```

Figure 19 - Grid Topology Helper (Liam T. Berridge, 2017)

```
 NS_LOG_UNCOND ("Creating Animation");
  /*----------------ANIMATION CREATION----------------*/
  grid.BoundingBox (1, 1, 100, 100);  // Set animation view size
  AnimationInterface anim (animFile);

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Figure 20 - Grid Topology Animation Creation (Liam T. Berridge, 2017)

The tree topology was the final implementation for the artefact, and required some alteration from the design shown in *Figure 14*. This was largely due to the topology scaling mostly in sets of two, whereas the root is singular; to circumvent the program crashing an if statement was required which was used to determine the current level which channels were being created from. When the statement reached the highest layer, or the root, it would instead only link to a singular node, completing the tree. This change is evident in *Figure 21*, with changes in later iterations of the file only being made to improve the code structure and readability, with additional comments and a diagram used to explain the concept behind certain functions (*Figure 22*).

```
std::vector<NodeContainer> channelCreation (uint32_t levelSet, std::vector<NodeContainer>
tLevel, std::vector<NodeContainer> subnetList, NodeContainer Nodes)
{
  uint32_t z = 0, j = 0, x = 0, y = 0; // Create variables for loops
  for(j = levelSet; j > 0; j--){  // Create channels from the top of the tree to the bottom
    NS_LOG_UNCOND ("Creating Level:" << j);
    uint32_t levelB = 0;
    if(j <= 1){
      for(x = 0; x < tLevel[j].GetN(); x++){  // For each node in the current level
(Penultimate level)
        for(y = 0; y < 2; y++){ // For each node in nNodes (Bottom Level of Tree)
          // NS_LOG_UNCOND ("Channel#:" << z << "  From:" << j << ":" << x << " To:" << j-1
<< ":" << levelB );
          subnetList[z] = NodeContainer (tLevel[j].Get(x), Nodes.Get(levelB));
          z++;
          levelB++;
        }
      }
    }
    else{
      for(x = 0; x < tLevel[j].GetN(); x++){  // For each node in the current level
        for(y = 0; y < 2; y++){ // For each node in the lower level
          // NS_LOG_UNCOND ("Channel#:" << z << "  From:" << j << ":" << x << " To:" << j-1
<< ":" << levelB );
          subnetList[z] = NodeContainer (tLevel[j].Get(x), tLevel[j-1].Get(levelB));
          z++;
          levelB++;
        }
      }
    }
  }
  return subnetList;
}
```

Figure 21 - Tree Topology Structure Creation (Liam T. Berridge, 2017)

```
std::vector<NodeContainer> levelCreation (uint32_t nNodes, uint32_t* levelSet, uint32_t*
channelCount)
{
  uint32_t i = nNodes/2; // Set variables
  std::vector<NodeContainer> tLevel (nNodes/2);                   // Create a NodeContainer
array, with a container for each separate level

  // Create nodes for tree structure, use nNodes as base of tree
  while(i >= 2){ // Divided by two so the first level is not the node level
    tLevel[*levelSet].Create(i); // Create a new level, populate with nNodes /= 2
    *channelCount = *channelCount + (i*2); // Add up nodes in the tree, multiply by 2 for
the leaf count
    NS_LOG_UNCOND ("Level Size:" << i << " Level No#:" << *levelSet << " Channel Count:" <<
*channelCount);
    i = i/2; // Divide by two to go up the tree one level
    *levelSet = *levelSet + 1; // Increase the level by 1 count
  }
  /* This code creates the tree upwards, using the nNodes variable as a measure of how large
the tree should be
    E.G
    Level 3:    0      Size: 1
              /  \
    Level 2:  0   0    Size: 2
              / \ / \
     Level 1:0  00  0   Size: 4
            /\ /||\/\
     NODES:0000 0000    Size: nNodes (8)
     However no connections exist at this stage, just level containers + internal nodes
     This also misses out the root of the tree (The single node at the top), which must be
created manually */
  return tLevel;
}
```

Figure 22 - Tree Topology Comments & Annotations (Liam T. Berridge, 2017)

Testing

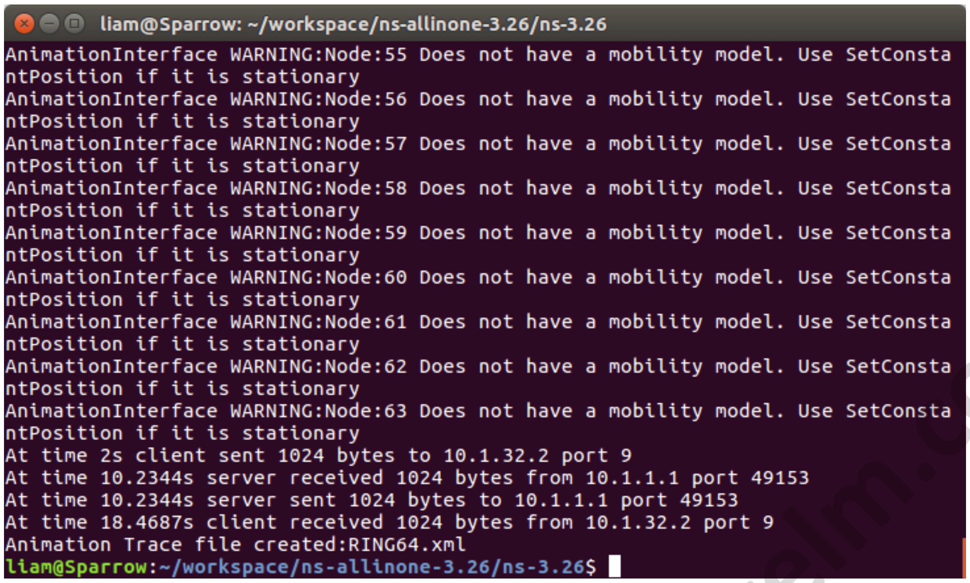
To test the various simulations created for the artefact, each was run using the NS3 simulation suite and associated tools; each simulation then produced a console output when run, providing timestamped packet logs. Each timestamp was measured in nanoseconds as specified at the start of each of the simulation scripts. These timestamps allowed collection of the RTT for each topology at different scales, and then using this information the results section graphs were created and allowed for comparison of the different topologies as they increased in scale. Additionally, animations were created for each topology to ensure the correct worst-case route was taken, as well as to produce a visual for the network simulation timeline. *Figure 23* demonstrates the console output created when any of the simulations created was run, and *Figure 24* demonstrates the animation output file, with the time, topology and traffic being visible. The following links also demonstrate the animation output for each topology. Worthy of note however is that the recording only shows scales 8 through 64, and does not include 128 and 256 node topologies. This was done as the animations become increasingly difficult to read as the topology scales, thus further inclusion of additional topology scales would not provide any valuable insight, although the animations were produced as part of the artefact.

P2P

Ring

Grid

Tree



```
liam@Sparrow: ~/workspace/ns-allinone-3.26/ns-3.26
AnimationInterface WARNING:Node:55 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:56 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:57 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:58 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:59 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:60 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:61 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:62 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
AnimationInterface WARNING:Node:63 Does not have a mobility model. Use SetConsta
ntPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.32.2 port 9
At time 10.2344s server received 1024 bytes from 10.1.1.1 port 49153
At time 10.2344s server sent 1024 bytes to 10.1.1.1 port 49153
At time 18.4687s client received 1024 bytes from 10.1.32.2 port 9
Animation Trace file created:RING64.xml
liam@Sparrow:~/workspace/ns-allinone-3.26/ns-3.26$ 
```

Figure 23 - Simulation Console Output (Liam T. Berridge, 2017)

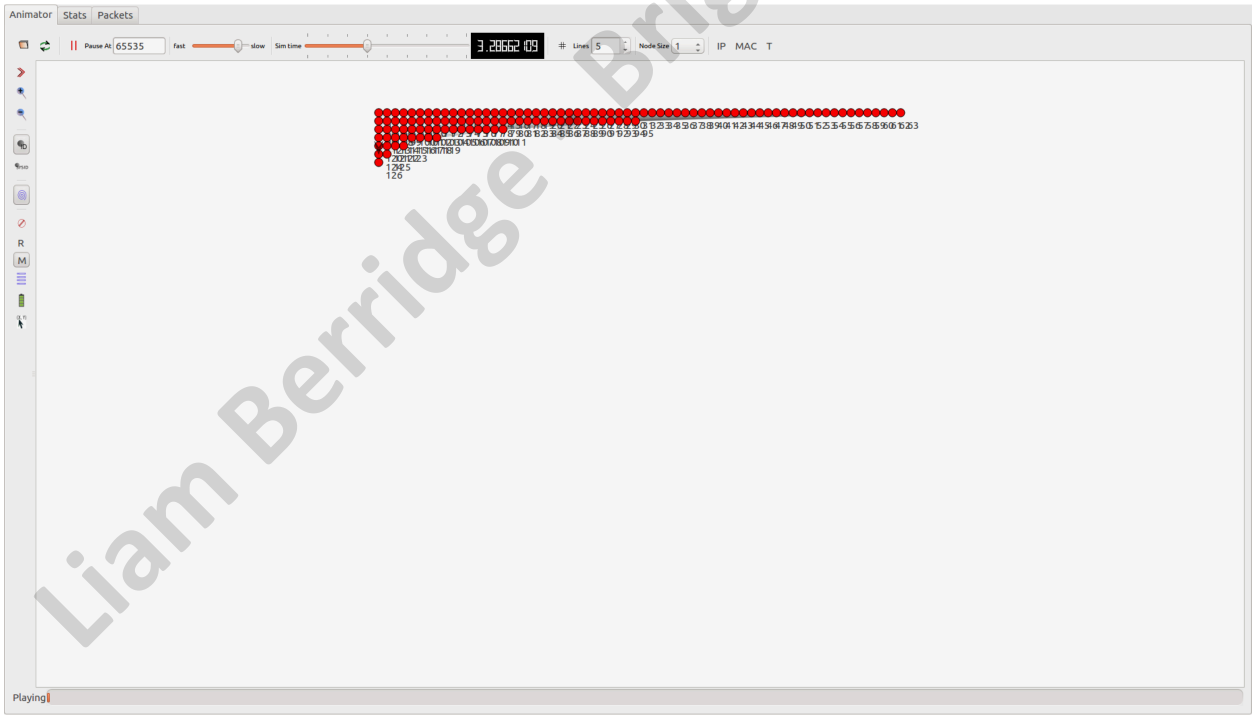


Figure 24 - Simulation Animation Output (Liam T. Berridge, 2017)

Evaluation

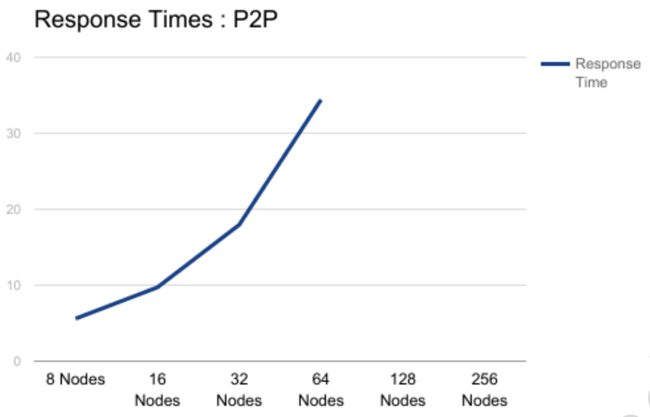
In evaluation of the artefact, it fulfills the aim and objectives outlined as it is able to accurately create a network simulation, featuring the lower layers present in the OSI model. Each of the topologies specified in the requirements stage was implemented, and the topologies are easily scaled using only a single variable, which may be determined in the code, or passed in through the console as required. The development of the artefact however was not as planned, and suffered a severe delay in its SDLC due to a change in the underlying system used for the project. This may have been avoided given sufficient research and forethought, however it was not until late into the first iterative cycle that the major issues with the software were brought to light.

These issues played a major role in the SDLC of the project, even altering the methodology used for its delivery, however this may have benefited the project in the long term as development was refined into a streamlined process. Further to this, the methodology used was only suited due to the nature of the SDLC issue, and without this issue an entirely different methodology and associated development practices may have been used instead.

Also worthy of note is the Grid topology producing more nodes than desired in some simulations, making results from these specific simulations unusable for scientific evidencing of a proof for the formulas theorised. This may have been fixed with additional time to work on the artefact or project.

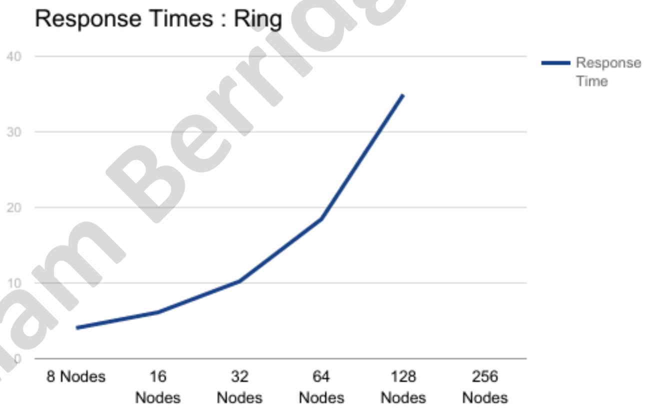
Results

P2P:



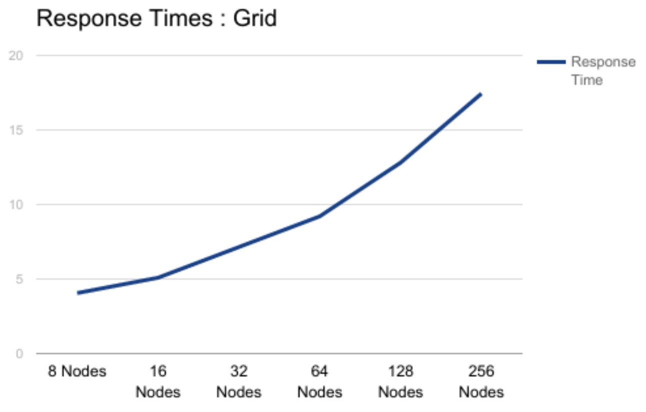
	Response Time
8 Nodes	5.602
16 Nodes	9.719
32 Nodes	17.954
64 Nodes	34.4229
128 Nodes	N/A
256 Nodes	N/A

Ring:



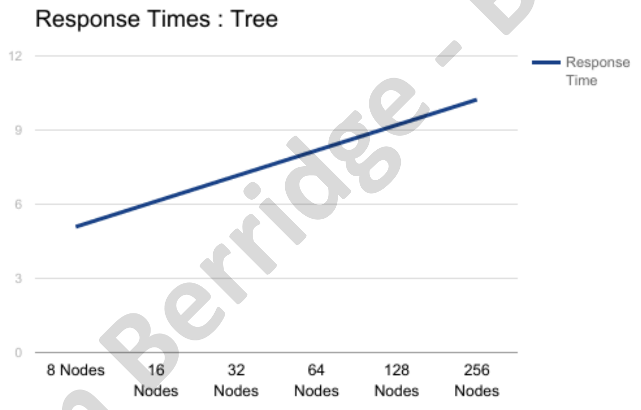
	Response Time
8 Nodes	4.058
16 Nodes	6.117
32 Nodes	10.234
64 Nodes	18.468
128 Nodes	34.937
256 Nodes	N/A

Grid:



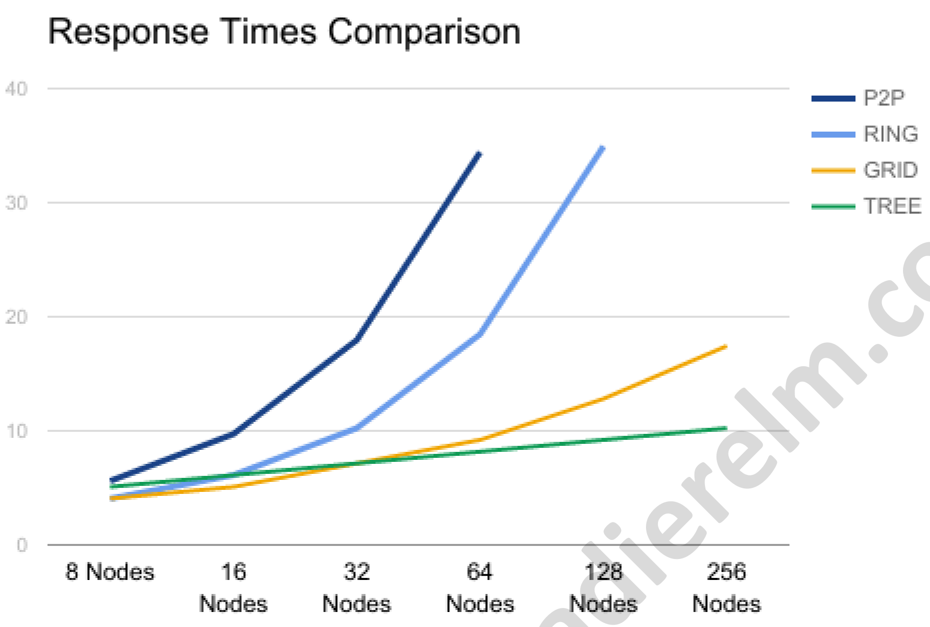
	Response Time
8 Nodes	4.058
16 Nodes	5.087
32 Nodes	7.146
64 Nodes	9.205
128 Nodes	12.807
256 Nodes	17.439

Tree:



	Response Time
8 Nodes	5.087
16 Nodes	6.117
32 Nodes	7.146
64 Nodes	8.175
128 Nodes	9.205
256 Nodes	10.234

All Topologies:



	P2P	RING	GRID	TREE
8 Nodes	5.602	4.058	4.058	5.087
16 Nodes	9.719	6.117	5.087	6.117
32 Nodes	17.954	10.234	7.146	7.146
64 Nodes	34.4229	18.468	9.205	8.175
128 Nodes	N/A	34.937	12.807	9.205
256 Nodes	N/A	N/A	17.439	10.234

Analysis

The above graphs show the RTT for each topology across different scales, with the last graph including all topologies for contrast and comparison. Both the Point-To-Point and Ring topologies presented in the graphs are shown to reach the ceiling of the graph, which was due to the simulation having a hard limit for timing Echo UDP type packets. A workaround for this was not attempted as the data gathered was sufficient to prove the theorised formulas, and would have taken an excessive amount of time to circumvent; had the data been inconclusive, then this would have instead been made a priority, however this was not the case. Also worthy of note is that each of the RTTs includes 2 nanoseconds of delay as the simulation does not send packets until 2 nanoseconds have passed.

The Point-To-Point topology shows an exponential increase in the RTT as the topology scales upwards, as is visible in the graphs shown in the above section.The results set from this topology however instead matches the efficiency expected from the ring topology, with the correct formula for producing the RTT being $N / 2$. There are still some minor anomalies in the results set, however these may be attributed to the simulation suite or the artefact producing inaccuracies in the simulations timing. On top of this the 2 nanoseconds of delay attribute for some of the anomalous time present in the results set. The results are still conclusive enough to allow deduction of the correct efficiency formula however.

The Ring topology acts in a very similar manner to the Point-To-Point topology, with the results being a further division of the results produced by the Point-To-Point topology. This would alter the formula for producing the expected efficiency to $N / 4$. Again minor anomalies are presented, but the results still are still viable for producing a worst-case efficiency formula for the topology. This makes the Ring topology a more desirable topology for implementation over Point-To-Point, as the RTT is halved and fault tolerance is added by the addition of a single channel. This however may not be feasible on networks covering a larger physical distance, or in networks where a circular topology may compromise the security of the network.

The Grid topology again shows an exponential increase in the RTT as the topology scales upwards, however it is significantly lower than both the Point-To-Point and Ring topologies. The results set for the grid topology however are not all viable, as the artefact will generate grids with additional nodes in certain cases. This is due to the underlying mathematical functions used to generate the grids, and whilst it does invalidate some results, the majority of the results are still valid for use in producing a grid efficiency formula. With some alterations to the code this may be fixed, but this would take additional time beyond the projects allotted time-frame. The theorised efficiency is disproved by the valid results from the grid topology, and the correct formula for the worst-case efficiency of graphs is difficult to discern from the valid results. The grid topology is still preferable for implementation over the previous topologies however, except for the cost of implementation where additional connections are required.

The Tree topology shows a linear increase in the RTT, unlike any of the other topologies previously designed and created for the artefact. The results set from this topology also matches the theorised formula, meaning the formula does not require any alteration or change to hold true. There are again minor discrepancies in the results set, but again the results are close enough to the expected values to prove the formula true. The linear efficiency of this topology also makes it less desirable for smaller networks, but extremely desirable for larger networks, as at 32 nodes it becomes the most efficiency topology of those tested and remains so whilst scaling the different topologies upwards.

Whilst not all the formulas theorised were correct, two of the four proposed only required minor alteration, and a third was proven to be true. Despite this however the artefact was successful in achieving its aim of proving or disproving the theorised worst-case efficiencies.

Reflective Analysis

In conclusion to this project and the research contained in this report, there are several factors which affected the development process of the artefact. Some of the affecting factors drastically altered the project from it's original proposal, with changes in the software and methodologies taking place over the course of development. Given that NS3 had been used from the outset of the project instead of NS2, the additional time may have been used to gain a more founded understanding of the software, and may have allowed for development of a much larger variety of topologies and with more accurately theorised worst-case efficiencies, though this is left to speculation. Alternatively this may have lead to more complications further along the development timeline, leaving less time for a solution to be found, and resulting in an unfinished or non-functional artefact; again however this is left to speculation. Additionally, MatLab features integration with NS3, and may have been useful to create visual representations of networks instead of NetAnim, however this presented various licensing issues, and instead the research conducted in this report uses entirely free softwares, making the results easily reproducible without any cost to other researchers.

The artefact itself could have been improved, with the Tree topology allowing an uneven number of nodes to be entered, or to allow a different number of connecting nodes to be specified, these changes would not drastically alter the results, but would instead allow for the reuse of the artefact for extensively testing other abstractions of the basic binary tree topology. This also applied to the Grid topology, where with additional time the topology could have been further tested and altered to ensure that grids confirm to the node count entered, and that additional nodes were not created and added to the topology at various, intermittent scales. Further to this, additional time may have been used in creating a more thorough documentation of the artefact, as well as being used to produce the artefact to a higher professional standard.

Despite these changes, the artefact was still successful in its aim, and provided flexibility in its ability to implement any given number of nodes, as long as the number specified fell within the ranges allowed by the IPv4 addressing protocol. With minor alterations this could be changed to the IPv6 protocol, allowing for even greater ranges of nodes to be implemented, though this may be excessive when testing small-world topologies.

Finally, given an earlier start to development in NS3, or an extended timeline for the project, it may have been possible to implement accurate traffic simulation, based on the work performed by Jyothi et al (2016), allowing for comparison of the research presented in this paper, with that published in another, giving an interesting comparison as to the use of RTT against throughput when measuring a given network topologies efficiency.

References

Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J. “New Directions on Agile Methods: A Comparative Analysis” (2003) Paper.

Faggioni, L., Neri, E., Castellana, C., Caramella, D. and Bartolozzi, C. “The future of PACS in healthcare enterprises”. European Journal of Radiology, 78(2) (2011) p.253-258. Article.

Gibbs, M., Quillen, H. “The Medical-Grade Network: Helping Transform Healthcare”. (2007) Paper.

Hirad, Homan Mike. "Hospital Network Infrastructure: a Modern Look Into the Network Backbone with Real Time Visibility" (2010). All Regis University Theses. Paper.

Jorwekar, G. J., Dandekar, K. N., Baviskar, P. K. "Picture Archiving and Communication System (PACS): Clinician’s Perspective About Filmless Imaging". Indian Journal of Surgery 77.S3 (2013) p.774-777. Article.

Jyothi, Sangeetha Abdu., Singla, Ankit., Godfrey, P. Brighten., Kolla, Alexandra. (2016) “Measuring and Understanding Throughput of Network Topologies”. University of Illinois. Paper.

Katkar, P. S., Ghorpade, V. R. “Comparative Study of Network Simulator: NS2 and NS3”. International Journal of Advanced Research in Computer Science and Software Engineering, 6.3 (2016) p.608-612. Article.

Langer, S. “Issues Surrounding PACS Archiving to External, Third-Party DICOM Archives”. Journal of Digital Imaging, 22(1) (2008) p.48-52. Article.

Langer, S., French, T., Segovis, C. “TCP/IP Optimization over Wide Area Networks: Implications for Teleradiology”. Journal of Digital Imaging, 24 (2011) p.314-321. Article.

Latora, V. Marchiori, M. “Efficient Behaviour of Small-World Networks”. (2008) Paper.

Lowell Lindstrom & Ron Jeffries “Extreme Programming and Agile Software Development Methodologies, Information Systems Management” (2004) 21:3, p.41-52. Article.

Nsnam.org. (2010). About NS-3. Web.

Singh, R., Chubb, L., Pantanowitz, L. and Parwani, A. “Standardization in digital pathology: Supplement 145 of the DICOM standards”. Journal of Pathology Informatics, 2(1) (2011) p.23. Article.

Stewart, B. K. “PACS Mini Refresher Course”. RadioGraphics 1992, 12 (1992) p.549-566. Article.

Ribeiro, L. S., Costa, C., Oliveira, J. L. “Clustering of distinct PACS archives using a cooperative peer-to-peer network”. Computer Methods And Programs In Biomedicine, 108 (2012) p. 1002-1011. Article.