Path protection switching in information centric networking

Edita Gashi

Supervised by: Dr. Martin Reed

School of Computer Science and Electronic Engineering

University of Essex

A thesis submitted for the degree of PhD

Date of Submission: 2nd of November 2022

In honor of my Father Selami Gashi I dedicate my Phd to him. I hold a special place in my heart for my late Dad. Education was important to him, and he encouraged me during my childhood. I am honored and privileged to honor Nanushi (Gjylie Maloki)'s wish, who passed away during the course of my doctoral studies.

Abstract

Since its formation, the Internet has experienced tremendous growth, constantly increasing traffic and new applications, including voice and video. However, it still keeps its original architecture drafted almost 40 years ago built on the end-to-end principle; this has proven to be problematic when there are failures as routing convergence is slow for unicast networks and even slower for multicast which has to rely upon slow multicast routing as no protection switching exists for multicast. This thesis investigates protection in an alternative approach for network communication, namely information centric networking (ICN) using the architecture proposed by the PSIRP/PURSUIT projects. This uses Bloom Filters to allow both unicast and multicast forwarding. However, the PSIRP/PURSUIT ICN approach did not investigate protection switching and this problem forms the main aim of this thesis.

The work builds on the research by Grover and Stamatelakis who introduced the concept of pre-configured protection p-cycles in 2000 for optical networks and, with modification, applicable to unicast IP or packet networks. This thesis shows how the p-cycle concept can be directly applied to packet networks that use PSIRP/PURSUIT ICN and extends the approach to encompass both unicast and multicast protection switching. Furthermore, it shows how the chosen p-cycles can be optimised to reduce the redundancy overhead introduced by the protection mechanism.

The work evaluates the approach from two aspects, the first is how the proposed approach compares to existing switching state and traffic in an MPLS multicast architecture. The second considers the redundancy overhead in three known network topologies for synthetic traffic matrices. The thesis is the first work to demonstrate the efficiency of Bloom filter based switching for multicast (and unicast) protection switching.

Contents

List of Figures

List of Tables

Acknowledgements

This thesis has been a long project of learning and maturing as researcher as an employee as a person and as a mother. It is with a great deal of gratitude and appreciation that I acknowledge that Professor Martin Reed is my hero. He will always hold a special place in my heart as well as in my family. It has been a great experience working with Professor Reed. The most amazing human being I have ever met or will ever meet is him .Thanks to his constant encouragement and guidance, I have completed my Ph.D. thesis. In appreciation, I would like to express my sincere gratitude and thanks to my husband, Ylli Nogu Gashi, who has supported me throughout my PhD research, encouraged me throughout my journey, listened to my frustrations and complaints, and believed in me regardless of the circumstances. I am very grateful that you have been an extremely generous and supportive friend to me during my last few months caring for our son Noah. I am highly obliged to your generosity and encouragements that enabled me to complete my doctorate. This is my greatest pleasure and deepest joy, as I acknowledge that I am the mother of a 15-month-old boy. I have had a very rewarding and amazing experience. I am thankful to Noah for his patience during the last few months. This was due to my absence from the house most of the time. I found this difficult to deal with.

I would like to take this opportunity to express my sincere gratitude to my mother Serbez Maloki who brought me to England for my education.Thank you so very much for giving me the chance to attain such a great honor and privilege to achieve the highest level of education and for fulfilling my Father's dream and yours.

I would like to thank my brother Selami Gashi and his wife Erza and my niece Moena. As well as my sister Anita Gashi and her husband Granit and my niece Maja for being in my life and supporting me. I would like to extend my appreciation to my uncle Bejtush Gashi encouraged me to pursue my Ph.D. I am thankful for his support. Furthermore, I would like to take this opportunity to express my gratitude to all the members of my Gashi and Nogu families who have been so supportive of me during my journey towards getting my doctorate, and I am grateful for that.

In Essex University, I had the pleasure of meeting Mays Al-Naday, Maryam Alhalboni Chathura Sarathchandra, and Louis Clift. Thank you for your encouragement and support. I have especially appreciated May's support and encouragement and consider her to be a close friend.

I am especially grateful to my close friend Eniana Kobuzi who has always motivated and encouraged me through my journey. She has consistently motivated me to achieve the success I've needed throughout the process of finishing my thesis and I am grateful for her encouragement.

Thank you God for your Blessings for making this possible for me.

Chapter 1

Introduction

1.1 Overview

The Internet has been exceeding all expectations with respect to its capability to cope with the modern demands for information dissemination. In recent years, YouTube has seen a dramatic increase in the amount of newly created content available as consumers' appetite for online video has increased. It is estimated that approximately 30,000 hours of new video content are uploaded every hour and from 2014 to 2020, video content hours uploaded every 60 seconds increased by about 40% percent. The amount of web traffic generated by mobile devices accounts for half of all web traffic worldwide in 2017 [1]. From the beginning of 2017, mobile devices (excluding tablets) accounted for 58.99% of the global website traffic, which had reached around 50% before finally surpassing that in 2020 [1]. Considering that multimedia content can now be served over the Internet in a variety of ways, we are witnessing staggering statistics. Video streaming traffic today is mostly uni-cast using

HTTPS based streaming [2], whereas the tremendous growth in video traffic suggests that multicast would be a better option [3]. However, multicast delivery to mobile and desktop devices (including smart TVs) is not supported by the current Internet architecture due to limitations in streaming architectures. However, future Internet architectures such as Information-Centric Networking (ICN) suggest that such multicast delivery may be possible [4] and this is discussed further in Chapter 2 and used as the groundwork of this thesis.

Failures appear frequently in networks, due to a variety of reasons e.g. cable cuts, power outages, compromised network elements, etc. Extensive research on path protection and recovery techniques in IP and MPLS current networks, has been carried out in the last few decades but all of these efforts have focused on either unicast routing or the physical layer. Deployment of native IP-based multicast has failed in the global Internet, and overlay-based multicast systems are inherently inefficient. The importance of network protection and recovery in current and future networks is essential part of networking for uni-cast and multicast.

In Information-Centric Networking (ICN), multicast forwarding is naturally handled with the help of a Bloom Filter when using the PURSUIT architecture [5]. Providing resilience against failures is an important requirement for many networks today, and the amount of disruption caused by a network-related outage is becoming increasingly significant, because even a single outage can cause millions of users to lose access to the network, resulting in significant losses for both users and network operators. Protection switching is the key technique used to ensure survivability. These protection techniques involve providing rerouting the traffic around the failure using multiple techniques. While, such protection switching has been addressed many times in traditional networks, there is very little discussion of this applied to ICN and this is focus of this thesis.

Physical layer impairment aware routing and physical layer routing with protection have been well investigated for current networks. This work asks the question: can existing techniques for these problem areas be applied to Information Centric Networks? In particular can traditional algorithms for protection and impairment aware routing be implemented in an ICN utilizing a switched optical system?

Information Centric Networking (ICN) is one of the significant directions of current networking research. ICN architecture proposes the use of the Publish/Subscribe paradigm to achieve data oriented approach unlike one that is found in current networks based on a destination approach [4,5]. The Internet allows rapid dissemination of information to users of internet/network, where the users only care about the content and is oblivious to the location of the content. Whether the consumer requires headlines from BBC news or videos from YouTube does not care nor is aware of the desired data or service location or for that matter in which ever machine the content resides and is retrieved from. ICN has a number of approaches and architectures to solve the problem of content delivery and these will be described in Chapter 2.

In this investigation we propose an overview on this increasing ICN paradigm, taking the PURSUIT network architecture approach. This will include the most relevant literature review on former research papers in areas of ICN focusing in PURSUIT approach. Furthermore the work explores how existing path protection mechanisms can be reused in ICN networks and possibly improved in this new context. In particular the work will look at extending the use of p-cycles [6] for the ICN path protection; p-cycles will be introduced in Chapter 2.

1.2 Aims and Objectives

The aim of this thesis is to propose protection switching in ICN based on the PURSUIT architecture that supports multicast and combine this with p-cycles which support fast recovery.

Following the primary aim the secondary aim is to improve the efficiency of the p-cycle based protection mechanism.

1.2.1 Objectives

- 6 ICN context objective***Explain the context of ICN -*** introduce the concept of ICN and in particular in the context of the PURSUIT network architecture.

- ***Explore ICN networks and its research approaches -*** Focusing on PURSUIT network architecture approach and along with its LIPSIN forwarding mechanism [7] (using Bloom Filters) and p-cycles for protection.

- ***Explore p-cycles -*** Exploring the use of p-cycle based on the LIPSIN forwarding mechansim [7] for protection in the use of ICN (PURSUIT architecture) and compare it with the current MPLS switching for multicast in terms of switching state and traffic capacity.

- ***Demonstrate through simulation the novel idea of using Multicast Protection-cycles Through Path-Based Switching.*** Optimising the choice of cycles to reduce the redundancy overhead.

1.3 Thesis Structure

This chapter reviews several propositions for the potential structures of ICN, briefly describing the current researched projects and defining the concepts along with roles of PURSUIT and POINT network architecture approach.

Chapter 2 carries out a literature review and explains the necessary background. Chapter 3 proposes the central novel idea on multicast protection-cycles through path-based switching. Chapter 4 proposes how the cycles used for the multicast protection can be optimised to reduce the redundancy overhead introduced by the p-cycles. Finally, Chapter 5 presents conclusions of the commenced research.

Chapter 2

Literature Review

In this chapter, we provide an overview of the background and foundations of traditional path protection in future networks such as content-based pub-sub systems that will be required to understand this dissertation. We leave the discussion on ICN for Chapter 3.

In this chapter we begin by describing related work on path protection approaches such as MPLS, including p-cycle concept [6]. We present the ICN internet architectures that focus on the content-based pub-sub system as central entity as opposed to the current host-centric IP networking architecture. Firstly, we describe related work and approaches, presenting the state-of-the-art in the area of content-based pub-sub systems. Providing this background allows for the classification of our own proposals that are presented later.

Secondly, we start to review recent path protection approaches to identify their inherent assumptions and the implications that can be drawn from them. This review substantiates the hypothesis we have proposed in Chapter 1 and intensifies the need to solve the associated research questions. These two contributions are reflected within the structure of this chapter. We begin by briefly explain the mechanism of a typical packet router which involves two closely couple concepts: (i) “routing” and (ii) packet forwarding, see Section 2.1. We next introduce generally the notions and concepts of traditional path protection in Section 2.4. Background on the p-cycle mechanism is provided in Section 2.5; this is important as in this thesis it is applied to a specific content-based pub-sub ICN architecture.

Section 2.4 explains the traditional path protection then elaborates on MPLS path protection and Traffic Engineering for the parameters influencing these measures that are current solutions to the methods of path protections. One of the main advances in protection are p-cycles that have been used in pre-configuring protection for current networks in multiple ways that we will explore in Section 2.5. We focus on exploring and reviewing the other main concept, the routing process, that includes multicast in Section 2.2.

2.1 Traditional IP routing

Section 2.1 is now changed 1 start of the art IP, not source routing

1 moved textIn a traditional IP network, the router analyses the destination address in the packet header at each hop and makes an independent forwarding decision as the packet travels from the source to the destination. IP forwarding is based on routing protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP). These protocols are designed to find the shortest path from the source to the destination and do not consider factors such as latency and traffic congestion. However, the shortest

path is not known to the end station, but is rather left to the routers to construct this in a connection-less manner. Clark gives an excellent introduction to the design choices chosen for the Internet [8], which are driven often by historic economic and technical conditions. Clark notes that source routing was not chosen for two reasons: firstly, it is an economic issue as there is no means to charge users for their choice of path; secondly, it is a security issues as malicious users could then use the knowledge of paths to introduce attack vectors to specific points [8]. The connection-less nature of IP has led to some issues for example the lack of scalability of high-speed routing at Internet scale. In order to overcome some of these restrictions Multi-protocol Label Switching (MPLS) was introduced [9]. While this is not strictly source routing, it does introduce the notion of a *label switched path* that achieves a similar intention to source routing, albeit from a different mechanism. While MPLS solves this scalability for unicast routing, it does not solve it for multicast; for multicast switching the PURSUIT/PSIRP switching described later in Chapter 3 introduces source routing using an efficient header data structure, Bloom filters. It should be noted that while MPLS and PURSUIT/PSIRP have source routing like capabilities they get around the problems discussed by Clark [8] as they do not allow the end station to directly know or control this source route.

The mechanism of a typical packet router are often separated into (i) the control computations "routing" and (ii) packet forwarding. The general routing problem in a network consists of finding a routing protocol, or routing function, or distributed routing algorithms, practically any pair of source and destination nodes, any message from the source are often routed to the destination.

When routing a message from a source to a destination in an IP network, to determine where to forward the message to, a node relies on the local routing table, the destination address, and the message headers. As a result of the routing algorithms, network state in form of forwarding information base (FIB) encoded in forwarding tables is created by the routing and resource control computations. As a result in-network memory information enables hardware to perform fast packet processing operations [10].

While the traditional model works well for host-centric unicast routing and forwarding systems, we find necessary to introduce subtle refinements in order to (i) match our focus on packet forwarding for content-oriented networks with multicast being the natural way of communication mode, and (ii) account for our probabilistic approach, where we explore solutions that deliver packets over protected path. Bloom filters [11] are data structures that are widely used to query group membership for use in network applications that require the inspection of packet headers and content in order to achieve space efficiency. Bloom filters are highly randomized data structures that are highly effective in space efficiency [11, 12].

Within this chapter, we will introduce the concept of forwarding efficiency in order to measure the bandwidth efficiency of multicast-capable forwarding methods, which will facilitate the comparison of alternative (probabilistic) approaches to the solution space that will be discussed in more detail in the following Chapter 3.

2.2 Multicast delivery

IP multicast, proposed in the early 90s by Deering et al. [13] was designed as a new general Internet layer service. In IP multicast, a new named entity, group address, was introduced to the network layer. Senders could choose a multicast group address they would send to and receivers could choose the group address they would listen to. The network would, match the senders and the receivers in a general many-to- many architecture. For the many-to-many model, the group address was topologically independent. Later on, source-specific multicast was designed, making a group name dependent on the source IP address in addition to the group address [14].

However, such multicast architectures placed the group maintenance at the routers. This requires routers to maintain per group state, which leads to additional complexity, and scalability and security problems [15]. Bloom filter based multicast has been proposed as a way to create scalable multicast by pushing the group management to the source and thus removing the per group state at the transit routers [7]. In Bloom filter based multicast, the links belonging to a multicast tree are encoded as a Bloom filter [11], which is then placed in the packet header. Each router checks for the presence of outgoing links and forwards the packet accordingly. Our work builds on using Bloom Filters in-coding multicast trees and then providing protection for this mechanism of forwarding.

2.2.1 IP Multicast

Multicast management for traffic flow through a L2 switch, static configuration of multicast addresses on ports is required to send multicast traffic it has a lot of administrative overhead and it is not very flexible.

IGMP protocol [16] is is used by hosts on a network segment to identify group membership sending IGMP messages to multicast router. For example using multicast for client to join a video stream, it sends a message directed at the multicast router. The multicast router is a router or L3 Switch with multicast enabled with IGMP Proxy or protocol independent multicast (PIM) routing protocol [17].

IGMP Snooping is used at a L2 switch to control multicast streams. IGMP Snooping prevents multicast flooding to all ports in the switch. A switch keeps a table of group to port members and streams are only sent where they are required instead of all ports. A LAN segment must have a Querier, which is usually a L3 switch or router, again, running IGMP Proxy or PIM routing.

IGMP Querier is a feature allowing L2 Switch to pretend to be a multicast router for the purpose of keeping IGMP snooping working. L3 equipment needs to support: IGMP Proxy this is to route between VLANs and Protocol independent Multicast (PIM-SM) (PIM -DM).

Unicast forwarding sends a copy of each packet to every client in case of a video stream, unicast typically is used when only few clients needs to access the application, if a message has to be sent to a larger group, the same information has to be carried multiple times even on the same links which results in consuming a lot bandwidth. Worst case scenario would be if the unicast addresss is unknown it would then broadcast the traffic to the whole network which would be huge waste of bandwidth and extra load on the server and reduce the network performance.

2.3 MPLS

MPLS and how Label Switch Paths are set up MPLS establishes a connection-oriented network overlaid onto the connection less framework of IP networks. Due to this connection-oriented architecture, several new techniques for traffic management are available [18]. In MPLS packets are forwarded by label swapping or label switching. The labels are contained in an MPLS header inserted into the data packet.

The label is a short fixed length physically contiguous identifier that instructs the routers how to forward the packets from the source to the destination. The packets follow a predetermined path called the Label Switched Path (LSP) illustrated in Fig. 2.1, a LSP is also referred to as an MPLS tunnel. A LSP is simply a concatenation of one or more hops in the network. Signalling protocols such as Resource Reservation Protocol with Tunneling Extensions (RSVP-TE) and Label Distribution Protocol (LDP) can be used to establish connections and distribute labels [18].

MPLS operates at the network layer of the OSI model and provides a mechanism for forwarding packets based on labels rather than traditional IP routing based on destination IP addresses.

In MPLS, Label Switch Paths (LSPs) are established to create virtual paths across the network. These LSPs are used to forward packets from one network node to another based on the labels attached to the packets. The following section explains how LSPs are configured in MPLS networks:

Label Distribution Protocol (LDP) or RSVP-TE: MPLS routers use protocols like Label Distribution Protocol (LDP) or RSVP-TE (Resource Reservation Protocol with Traffic Engineering extensions) to exchange label information with their neighboring routers. These protocols allow routers to establish label bindings, which associate specific labels with network destinations.

Label assignment: Once the label distribution protocols have exchanged information, each MPLS router assigns a label to a particular route or network prefix. These labels are typically locally significant within a router's domain and are used to forward packets toward the destination.

Label Forwarding: Once the labels are assigned, the MPLS routers construct a Label Forwarding Information Base (LFIB) that maps incoming labels to outgoing interfaces or next-hop routers. This mapping determines how the incoming labeled packet should be forwarded to reach its destination.

Label switching: When a labeled packet arrives at an MPLS router, the router examines the incoming label and performs a lookup in its LFIB to determine the appropriate outgoing interface or next-hop router for that label. The router then swaps the incoming label with the outgoing label and forwards the packet accordingly.

Label stack: In cases where a packet needs to traverse multiple MPLS networks or hops, a label stack is used. The label stack allows routers to apply multiple labels to a packet to guide it through various LSPs until it reaches its final destination.

It is important to note that the exact process of setting up LSPs can vary depending on the specific implementation of MPLS and the protocols used. MPLS can be configured using different protocols such as LDP, RSVP-TE, or even BGP (Border Gateway Protocol) with MPLS extensions. Each protocol has its own mechanisms for label distribution and path establishment.

These protocols establish paths through the MPLS network and reserve network resources along the path according to the requirement. The routers in the MPLS network are called the Label Switch Routers (LSR's) and the routers in the edge of the network are called Label Edge Routers (LER's). The router at the entry point of the tunnel is referred to as the ingress router and the router at the endpoint is called the egress router.

Fig. 2.1 When packets enter the network they are classified into Forwarding Equivalence Classes (FEC). All the packets belonging to the same FEC receive the same forwarding treatment. A FEC is a logical entity created by the router to represent a class of packets. After the initial classification, the packets are then assigned a label and the path corresponding to that FEC by the LER.

The label is used to identify an FEC. The packets are then forwarded along the LSP. At each hop, the LSR removes the incoming label and attaches an outgoing label. This outgoing label instructs the next router how the packet should be forwarded. In Fig. 2.1 exhibits a typical MPLS network and the associated elements. The central cloud represents the MPLS network. The customer edge routers (CE routers) interface with the provider edge routers (PE routers). The PE router at the ingress point attaches the MPLS label to the packet and the PE router at the egress point removes the MPLS

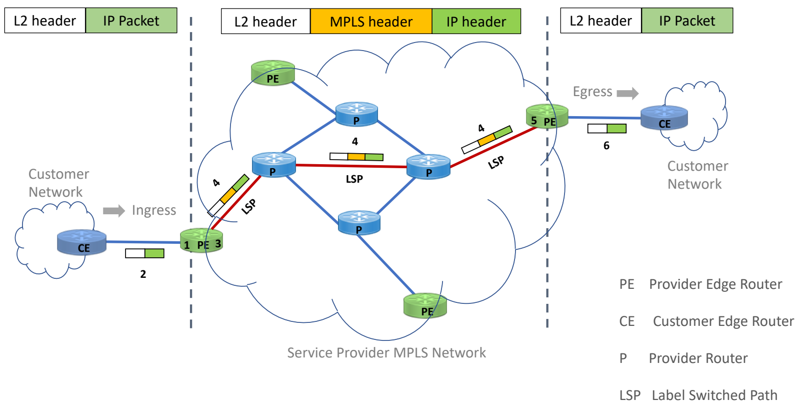


Figure 2.1: MPLS Network

label from the packet. The CE and PE routers are the LER's. Within the MPLS domain, the Provider (P) routers forward the traffic hop by hop based on the labels. The P routers are the LSR's.

MPLS is a framework of functions it combines the benefits of packet forwarding based on Layer2 Switching with that of Layer3 Routing and also provides the benefit of Traffic Engineering (TE). It can also be used to set up Virtual Private Networks.

MPLS allows label stacking [19], it stores a number of labels that can speed up the flow of the network traffic and it makes it easier to manage, rather than looking up routing tables based on the IP addresses. In the case of ICN, there are no IP addresses so MPLS techniques can be compared with ICN networks as we move traffic around based on labels known as Label Link Identifiers (LIDs) explained fully in Section 2.3.2.

2.3.1 Multicast with MPLS multi-point LSPs

The predicaments with using multicast at operator scale have triggered stateless solutions, such as multi-protocol stateless switching (MPSS) [20] and line speed publish/subscribe inter-networking (LIPSIN) [7], that use the Bloom filter (BF) as the forwarding identifier (FID) in the packet-header. For conciseness, this form of switching will be described as BF switching. Certainly, BF switching has been utilised by architectures such as PSIRP and PURSUIT [5] that determine the basis of a clean- slate information-centric network (ICN). These trials demonstrated that BF switching can deliver very efficient unicast/multicast forwarding with the minimal state either in software switches or utilising SDN switches [21].

MPLS supports Traffic Engineering as it allows explicit routing of packets. Traffic Engineering (TE) is the process of selecting suitable paths for the flow of data in a network so as to efficiently utilize the network resources and enhance the network performance while maximizing the network revenue. The main goal of Traffic Engineering is to deliver efficient and reliable network operations. Traffic Engineering attempts to compute a path from the source node to the destination while adhering to constraints such as bandwidth, delay and other administrative requirements. After the path is computed, Traffic Engineering enables the establishment of the path and also maintains the forwarding capability of the path.

This allows MPLS networks to pre-establish backup paths for the primary paths [22]. Each of these backup LSP's has the same amount of bandwidth as the primary LSP's. Path Protection provides an end to end failure recovery mechanism for MPLS Traffic Engineering tunnels. The backup paths have the same source-destination pair as the corresponding primary paths. One or more LSP's are established in advance to provide failure protection for the protected LSP.

If the primary LSP fails, the traffic is instantly switched onto the backup LSP's temporarily [22]. It is important that the backup LSP's are either link disjoint or node disjoint with the primary LSP's i.e. the backup LSP's do not share a link or a node with the primary LSP's else the failure of a shared link or node would affect both the primary and backup paths.

Rerouting is another technique of providing path protection. In this method, upon failure of the primary path the traffic is simply rerouted along a new path or path segment. No resources are reserved in advance until the fault occurs. This mechanism provides higher resource utilization but is slow compared to protection switching as the secondary path is established upon detection of a failure.

In MPLS, the routing decisions, particularly the choice of the path, are addressed in connection with the underlying IP network. In MPLS, the path is determined at the source and the forwarding is performed by use of a label switched paths which are established beforehand on the base of an underlying IP network. However, once a packet passes through the MPLS network only the local forwarding decisions are made. Consequently, multicast turns out to be very complex as an aggregation of switched labels is not easy and the aggregation translates in multicast state and consequently, the efficiency is reduced. Furthermore, considering the whole path is exposed in the network, security may become an issue [8].

A similar perspective to MPLS has been proposed for the novel clean-slate information/content-centric network developed initially within the framework of the European project, PSIRP/PURSUIT [5] and POINT [4], where the concept of an IP address is not part of the architecture anymore as explained in Chapter 3.

2.4 Traditional Path Protection

The aim of this section is to provide an overview of the traditional path protection mechanism. To date, IP traffic is the most common type of traffic carried across networks, and it accounts for the majority of traffic. The native IP system is limited in several respects, including the lack of traffic engineering capabilities, QoS features, Path Protection mechanisms, and mechanisms for ensuring reliability. It is due to the limitations of IP alone that we explore a variety of approaches that are currently being used in MPLS and Traffic Engineering, as well as the p-cycle approach for path protection, to overcome these limitations [22–26].

2.4.1 Protection methods

Protection also known as protection switching refers to a primary path pre-established backup path that is switched when and if the primary path has failed. Path protection requires the protection path of a request to be completely link-disjoint from the corresponding primary path, while the link protection scheme reroutes all affected requests over a set of replacement paths between the two nodes terminating the failed link. Depending on the span of the backup path a few protection techniques can be described as follows:

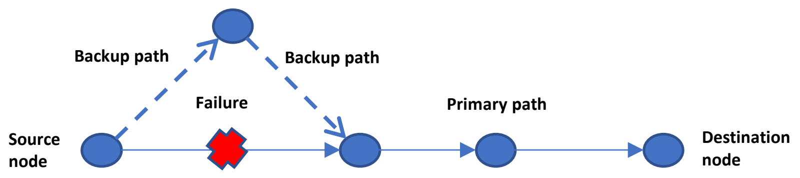


Figure 2.2: Link Protection with alternative protection path

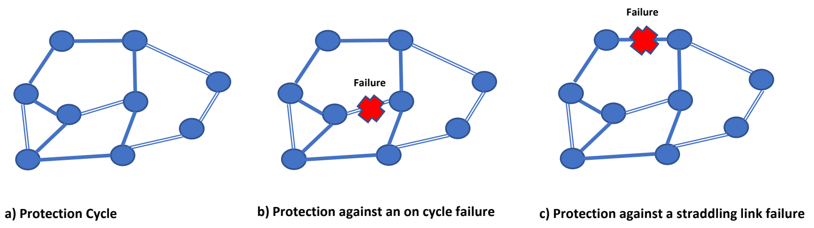

Failure
Failure
a) Protection Cycle
b) Protection against an on cycle failure
c) Protection against a straddling link failure

Figure 2.3: Protection Cycles : a) protection cycle without failure b) Protection against an on cycle failure and c)Protection against a straddling link failure

(i) Local protection : when the backup path protects a single link also known as link protection refer to Fig. 2.2 that depicts a graphical representation of a link protection with the alternative path protection as alternative path.

(ii) Pre-configure protection cycle known as a p-cycle is a form of pre-configured cyclic closed paths from a spare capacity in a given mesh networks

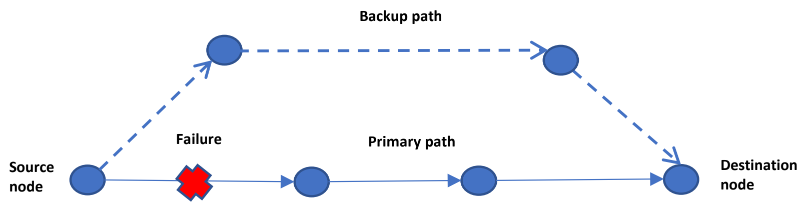


Figure 2.4: Path Protection

used to protect against on-cycle link and straddling link not on the cycle but the end nodes of the link are on the cycle refer to Fig. 2.3 that depicts the graphical representation of those protections.

(iii) Path protection when the path span from source (ingress) node to the destination (egress) node refer to Fig. 2.4 that represents the path protection in graphical form.

Local protection mechanisms have the disadvantage of requiring the establishment of a backup path for each link/node of the primary path in order to protect against loss of data. Network resources on a backup path are dedicated for protecting only one primary path, local protection mechanisms imply inefficient usage of resources. Switching time associated with protection from primary to backup path for the local protection mechanisms is shorter than for a path protection mechanisms, cause it is performed locally at the detecting node. Path protection switching is performed at the source node, hence more signaling is required implying longer recovery time, which also means larger amounts of lost traffic as compared to local protection mechanisms. There are many advantages to path protection, including the ability to more efficiently utilize backup resources and to minimize propagation delays [27].

2.4.2 Packet network resilience

Network resilience in packet switched networks is enabled through dynamic routing protocols such as OSPF. OSPF [28] is based on the relative costs of transferring information between hops (mainly routers and networks). The protocol is classified as an Interior-Gateway Protocol (IGP), and is intended to be run internally in an AS. It is distributed amongst the routers in the AS, and allows them to build the same representation of the AS's network topology. This is achieved through publishing Link-State Advertisements (LSAs) by the routers. Each router then constructs a shortest-path tree to different destinations, with itself as a root. Then, it routes IP packets through the net, based solely on their IP addresses.

In case of topological changes, the routes will be recalculated, using updated LSAs (or their absence). Yet, the protocol generates relatively small amounts of traffic used for the configuration. However, routing resilience using such protocols is much slower than protection mechanisms [27].

2.4.3 Optical resilience

This thesis is bringing fast protection to packet networks. Currently, this does not exist within the packet networks which instead rely upon routing to provide routing table restoration. This type of resilience happens on a much longer time-frame. However, fast protection switching has been used over a long period in optical networks. Consequently, as this is closer to the mechanisms used in this thesis optical network protection will be reviewed here. Later we will show how p-cycles that were originally designed for optical networks can be utilised for the packet network switching used in this thesis.

SONET Protection Networks Protection is defined here as a resilience mechanism where the path for the protected traffic is both determined and established prior to a failure. Protection examples include SONET APS, Bundled Interfaces Protection (BIP), RPR protection, and MPLS TE and FRR. This is in contrast to restoration mechanisms, where the path for the protected traffic is established (and often discovered) after the failure (e.g., through Open Shortest Path First, OSPF, or Intermediate System to Intermediate System, IS-IS, convergence). By definition, protection is faster than restoration since the protection path is assigned and pre-programmed prior to a failure [29].

1+1 Path Protection Mechanism

With 1+1 path protection [30] the backup path transmits a copy of the traffic on the primary path. Once detection of the failure is acknowledged, the traffic is routed across the backup path. However the backup path resources are not available to low priority traffic. Hence there is no sharing of the resources involved in the backup path, resulting in fastest backup path. 1+1 path protection mechanism can be link disjoint or node disjoint.

1:1 Path Protection Mechanism

With 1:1 path protection [30] the resources of the backup path are available to low priority traffic. Therefore it allows sharing of the backup path resources. Once a detection of a failure is acknowledged the low priority traffic is taken into account and is switched from the primary to the backup LSP. 1:1 path protection mechanism can be link disjoint or node disjoint.

2.5 Path Protection with P-cycles

P-cycles provide an attractive way for network protection, since the benefits of both ring-based protection and mesh-based restoration can be utilized [31]. In this section the concept of p-cycles for link protection is explained.

Span-protecting p-Cycles are a type of span protection technique “p” stands for "protection" and "preconfigured." The traditional span restorable network performs restoration at the two end nodes of a link, and shows much better spare capacity efficiency but requires a relatively longer restoration time than the ring techniques [30]. Better than the span-restorable technique, span-protecting p-cycles can achieve not only good spare capacity efficiency, but also a very fast restoration speed as they are ring like [31]. Thus, span-protecting p-cycles are often featured as mesh-restorable spare capacity efficiency and ring-like restoration speed.

A p-cycle has the same layout as the traditional ring, but it is more advanced than a ring in its protection capability. In addition to protecting on-cycle spans, which is also protected under the ring technique, a p-cycle can also protect straddling spans as shown in Fig. 2.6. An on-cycle span is defined as a span or link that is traversed by the cycle, and straddling span is referred to as a span who is not on the cycle, but whose two end nodes are on the cycle. For protection, a p-cycle can provide one protection route for an on-cycle span failure and two protection routes for a straddling span failure. Fig. 2.5 shows a pre-established p-cycle and it protects all the on-cycle spans including spans (1-3), (1-5), (3-7), (7-4), (4-6) and (6-5) and straddling spans including spans (1-4), (1-3) and (4-5). In Fig. 2.5, when on-cycle span (3-7) fails, the p-cycle provides a protection path (3-1-5-6-4-7) to recover the failure, which operates very similarly to the traditional BLSR technique. One unit of p-cycle protection capacity can protect one unit of working capacity on an on-cycle span. More efficient than the ring, the p-cycle also offers protection for straddling spans as shown in Fig. 2.6, where when straddling link (3-4) fails the p-cycle provides two protection routes

(3-7-4) and (3-1-5-6-4) for the failure recovery. Each of them carries a half of traffic from the working straddling link. Thus, for straddling span protection, one unit of p-cycle capacity can protect two units of working capacity on a straddling span.

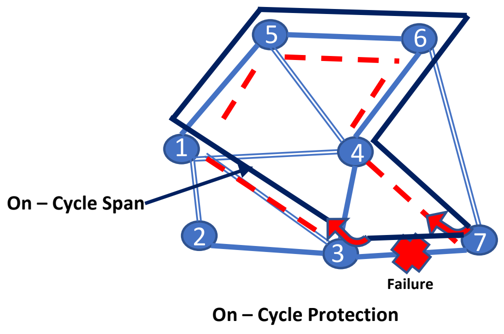


Figure 2.5: on-cycle span protection of p-cycle

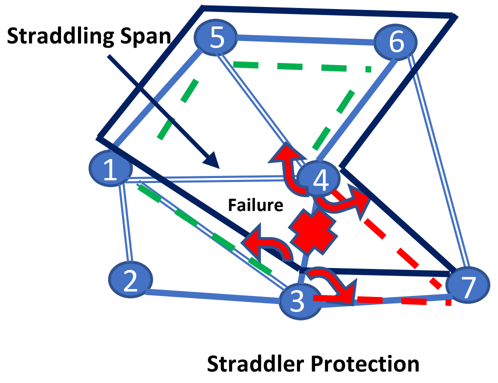


Figure 2.6: Straddling span protection of p-cycle

P-Cycles, as described in [32,33] provide a method for protecting MPLS networks by establishing virtual protection cycles. P-Cycles are a form of pre-planned protection scheme that offer fast and efficient restoration in case of network failures. 1 p-cycles are best used for relatively static links, and what assumptions you are making about transient links that appear at the edge.It should be noted that P-cycles are designed to provide protection in a mesh like network where cycle structures can be created in the mesh. As such they are suitable for core networks with relatively static connections. They are less suited to leaf connections at the access. In the same way, this thesis addresses protection in the core network. To protect the access network alternative techniques such as multi-homing (a form of 1+1) are generally used [34], the thesis assumes that such mechanisms are in place for access protection. In this section, we will discuss several different ways in which P-Cycles can be employed to protect MPLS networks. These are as follows:

Protection Cycle Creation: P-Cycles are established by selecting a set of network nodes and links to form a cycle or ring topology. This cycle is designed to encompass the protected network elements and provide alternative paths for traffic in the event of a failure. The P-Cycle should be able to handle the capacity of the working network elements it protects.

Label Switch Path (LSP) Establishment: Once the P-Cycles are defined, the MPLS routers establish Label Switch Paths (LSPs) that traverse the primary working paths as well as the P-Cycles. LSPs are set up to direct traffic along the working paths in normal conditions.

P-Cycle Activation: When a failure occurs on the working path, the MPLS routers detect the failure and activate the corresponding P-Cycle. The routers switch the affected LSPs from the failed working path to the corresponding protection path within the P-Cycle.

Traffic Rerouting: By switching the LSPs to the protection paths within the P-Cycle, MPLS routers are able to reroute traffic around the failed network elements. The P-Cycles provide alternate paths that can carry the traffic while the failed elements are restored or repaired.

Fast Restoration: P-Cycles offer fast restoration times because the protection paths are pre-planned and readily available. There is no need for complex computation or signaling protocols to establish new paths. The MPLS routers simply switch the LSPs to the protection paths, minimizing downtime and ensuring continuity of service.

By incorporating P-Cycles into MPLS networks, network operators can enhance the resiliency and fault tolerance of their infrastructure. P-Cycles provide a robust protection mechanism that can do fast reroute traffic in the event of network failures, ensuring uninterrupted service for critical applications.

Chapter 3

Information Centric Networking

Information-Centric Networking, is a promising future internet architecture that aims to revolutionize the way information is accessed, distributed, and consumed on the internet. Unlike the traditional host-centric approach of the current internet, ICN focuses on content as the primary entity, shifting the paradigm from ”where” information resides to ”what” information is being requested. [35]

In ICN, the fundamental concept is to assign unique names or identifiers to content, rather than relying solely on IP addresses. This enables a more efficient and scalable content delivery model. When a user requests content, routers in the network use these content names to retrieve and deliver the requested data, regardless of its physical location. [35, 36]

Several key aspects and benefits of ICN should be noted, including:

- ICN operates on the principle of named data: content is assigned a unique name. This allows content to be cached and distributed across the network, bringing it closer to the end-users and reducing the reliance on centralized servers. [37]

- Data-centric Security: ICN inherently provides better security mechanisms compared to the host-centric model. Since data is named and authenticated, it becomes easier to apply access controls, verify integrity, and ensure confidentiality. [38]

- Efficient Content Delivery: ICN utilizes in-network caching, enabling content to be stored closer to users. This reduces the strain on content servers and improves overall network efficiency by minimizing redundant data transfers. [36, 38, 39]

- Mobility Support: ICN seamlessly handles mobility by associating content names with the actual data, rather than relying on fixed IP addresses. This simplifies the process of content retrieval and delivery, even when devices or users change their network attachment points. [40]

- Multicast and Content Routing: ICN inherently supports content-based multicast, where multiple users interested in the same content can efficiently receive it from a single transmission. Additionally, routing in ICN is based on content names, enabling more flexible and scalable routing schemes. [36]

- Internet of Things (IoT) Integration: ICN is well-suited for IoT applications, as it can handle large-scale data dissemination and facilitate communication between IoT devices by leveraging content names. [41, 42]

While ICN holds significant potential, it is still an evolving area of research and faces challenges in terms of standardization, deployment, and transitioning from the current internet infrastructure. However, ICN offers a compelling vision for a future internet architecture that addresses many of the limitations and inefficiencies of the present host-centric model, making it an exciting area of exploration for researchers and industry experts alike.

Internet pioneer Van Jacobson provides a vision [35] to understand the motivation for a networking revolution; while the first networking generation was about wiring (telephony) and the second generation was about interconnecting wires (TCP/IP), the next generation should be about interconnecting information at large. This shift in the orientation of network architecture design implies rethinking many fundamentals by handling information as a first class object. A key question is to what extent a new paradigm thinking 'out-of-the-TCP/IP-box' for the future network is really necessary, e.g., as packet switching was to circuit switching in the 70's. The reasoning is based on the large scale use of the Internet for dissemination of data. A myriad of devices, including user-attended terminals and long-running automated services, generate and consume content, without caring about the actual data source location as long as integrity, authenticity and timeliness are assured. This shift toward information-oriented networking is also noticeable in the momentum of service oriented architectures (SOA) [43], deep packet inspection (DPI) [44], content delivery networks (CDN) [45] and peer-to-peer (P2P) overlay technologies such as POINT [4].

3.1 Overview of ICN

In the following, we first briefly describe our overall pub/sub based internetworking architecture, and then present a forwarding solution that resembles strict source routing but uses a fixed-sized, compact header, using Bloom filters (explained later in Section 3.2) which are suitable for fast hardware implementation.

The Internet architecture has been a tremendous success, during past decades it has grown from a small research network into a global critical communications infrastructure with billions of users. Its success has come with rapid changes in the user base and use patterns. The Internet was designed to enable global communications by providing a simple "best effort" service of datagram delivery among network attached devices an end-to-end communication and accessing computing facilities remotely, the patterns of communications has changed. Van Jacobson predicted in his first talk in 2006 that networks was changing from the idea of interconnecting host to interconnecting information and quite rightly the network architecture should change with it and in-visioned a New Way to Look at Networking [35].

The recent rapid developments of Information Centric Networking (ICN) research activities have significantly increased in results. The vision is to move to so called 'Future Internet', based on Pub/Sub paradigm as an alternative to the commonly used Send-Receive paradigm. The principle of ICN communication is focused on directly retrieving information objects: securely, reliably, scalable, and most of all efficiently. The architectural design is driven to directly address the current network challenges that are fast growing and demanding effective and efficient data distribution in the

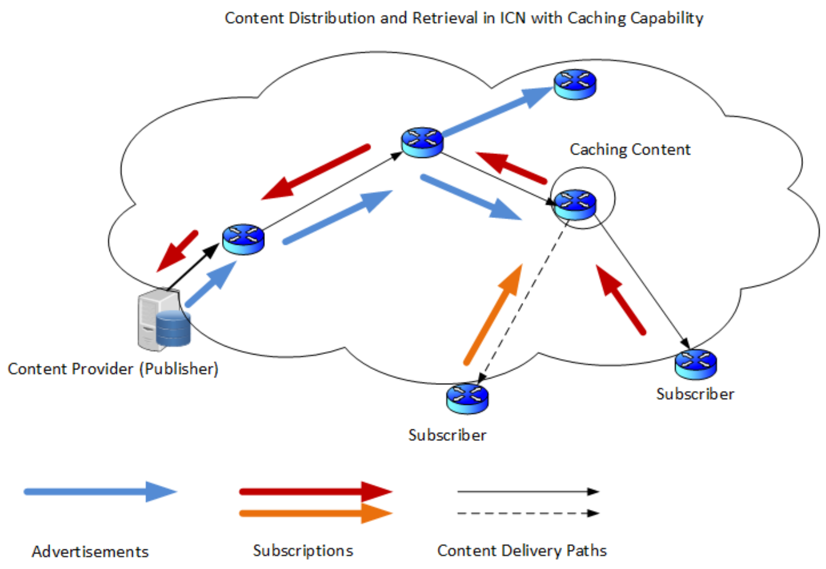

Content Distribution and Retrieval in ICN with Caching Capability
Caching Content
Content Provider (Publisher)
Subscriber
Subscriber
Advertisements
Subscriptions
Content Delivery Paths

Figure 3.1: Content Distribution and Retrieval

communication services in networks/internet.

Furthermost ICN architectures use data caching, and two most commonly caching approaches are: caching at the network edge and in network caching [46]. When caching is used information gets stored along a path due to previous requests made earlier in addition to this means subsequent requests for the same information will be returned from a closer location/server to the requester/client. Therefore this will result in efficient use of bandwidth and improved reduced latency.

Routing in ICN involves finding and delivering copies of data objects to provide receivers from the most efficient location in the network, as it shows below on the diagram Fig. 3.1

When using caching for content retrieval, it is important to assure users that cached content/information has truly come from the original source.

The focus is only on content, not on the hosts storing content. However any data object delivery is controlled by receivers/subscribers as shown on Fig. 3.1 above only when they requested it than it is distributed to be received. In ICN, the content provider/senders who provide the object to distribute, do not send content directly to receivers/subscribers neither it transmits data into the network. This is completed via advertisement message as presented on the Fig. 3.1.

First the provider/publisher sends an advertisement message to notify the network that it has content to distribute, however this is done without the receivers/subscribers awareness that may be interested in the particular data. Only when receivers/subscribers requirement matches a published information object, then the network initiates a delivery path from the sender to the receiver/subscriber so that content/object retrieval can be initiated for the receiver/subscriber. This is very much similar to rendezvous system as is described in the following PURSUIT network architecture.

3.1.1 PSIRP/ PURSUIT Network Architecture

We can view the global network of information as an acyclic graph of related pieces of data, each identified and scoped by some identifiers. In the PSIRP architecture, identifiers define the relationships between the pieces of information in our system on the different levels, such as the application or networking level. The following are classes of identifiers:

- Application identifiers (AId), used directly by publishers and subscribers.

- Rendezvous identifiers (RId), used to bridge higher level identifiers with lower layer identifiers.
- Scope identifiers (SId), used to delimit the reachability of given information.
- Forwarding identifiers (FId), used to define network transit paths and transport publications across networks using Bloom based forwarding (see Section 3.2).

A rendezvous identifier is implicitly associated with a well-defined (but not necessarily fixed) data set, consisting of one or more publications. The data sets may also have associated metadata, which may include, e.g., scoping information and other useful information either for ultimate receivers or for network elements. We also consider metadata that is understood within the network itself. Such network-level metadata might be concerned with how the communication for a particular data item may be conducted. This network metadata may be found as soft state within the network, carried as part of the communication header, or referred to by separate identifiers. Such functions may include access control, flow control, error notification, congestion notification, etc.

PURSUIT architecture is formed from three parts: Protocol Suit Architecture also known as Component Wheel, the Network Architecture, and the Service Model. Component wheel considered as the core of PURSUIT conceptual architecture.

The PURSUIT architecture Component Wheel including Rendezvous, Routing, Forwarding and Caching is displayed in figure below:

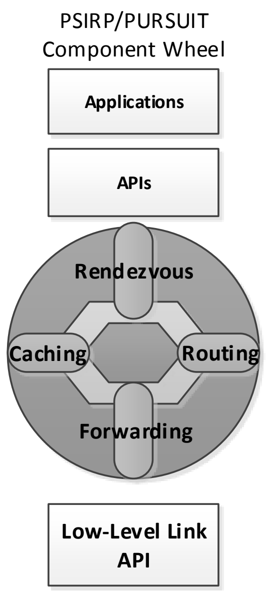


Figure 3.2: PURSUIT Component Wheel

As it shows on the diagram above below the application is the APIs who improves the accessibility to the networking applications, which also helps with simpler implementation for various networking components. Connecting applications across the network, PURSUIT network architecture relies on a special class of identifiers such as Rendezvous Identifiers (RId).

PURSUIT places the notion of information in the centre of attention through addressing information directly via Rendezvous Identifiers and allowing for building networks of information via the concept of scopes. The PURSUIT aims to modify the routing and forwarding paradigm of the current internet. This is to make sure that it will be able to work based on the concept of information and labels assigned to each pieces of information specifying the scope of each piece and addressing all pieces of information through rendezvous identifiers. Unlike the current internet architecture that

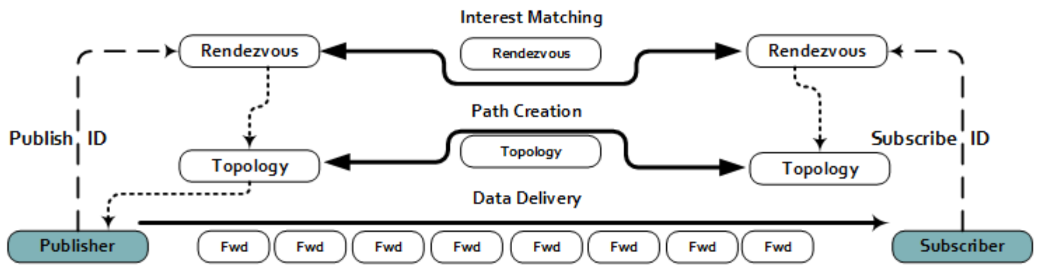


Figure 3.3: Architectural functions in PURSUIT

is based on assigning IP addresses to end hosts.

Architectural functions demonstration on a diagram as follows:

- Rendezvous at the top matches publishers and subscribers events
- Topology below is the network topology that knows all the paths which creates paths
- Forwarding forwards the data to the subscribers which is fast data delivery

3.1.2 Rendezvous system

The rendezvous is the function which is responsible for matching publications and subscriptions. Note that the solution implements a middle-ware implementation of pub/sub and the purpose of the Rendezvous and Topology manager is to take the middle-ware commands and turn them into packet level operations. The rendezvous can be used by any layer to implement pub/sub operations. For example, it could be used to replace a traditional middleware pub/sub system [47] with an example being a voice or video application that uses the pub/sub operations directly [48]; alternatively, in the

POINT project it is used to implement IP networking by using publish to send packets to a server that subscribes to receive a packet [49] by using middleware boxes that operate as gateways between traditional IP and the ICN network. Thus, rendezvous is a somewhat abstract concept that needs to be placed in the context of the particular application, please see the examples just given for more detail. There are a variety of publish/subscribe networking approaches [50] . The topic-based mechanism seems to offer the best scalability. In this scheme, participants publish events and subscribe to topics that are identified with a keyword. In our case, the keyword is a randomly looking flat label called RId [50]. The concept of topic is very similar to the concept of group. Subscribing to a topic T can be viewed as becoming a member of the group T and publishing an event on the topic T can be viewed as broadcasting that event among the members of the group T.

The topic in this context is viewed as a particular event service identified with a name. Having flat labels, the use of hierarchies in organising topics proposed in PSIRP appears to be a key improvement to the original scheme. The hierarchy offers the possibility to organise topics according to relationships. A subscription made to some node in the hierarchy implies a subscription to all the subtopics of the node. The role of the rendezvous function can be summarised in this way: an application first subscribes to a local rendezvous service. If the application has any data that it wants to make available to other applications, it publishes a list of the publications at the rendezvous service. Similarly, if the application wants to receive data that has been published by some other applications, it subscribes to those

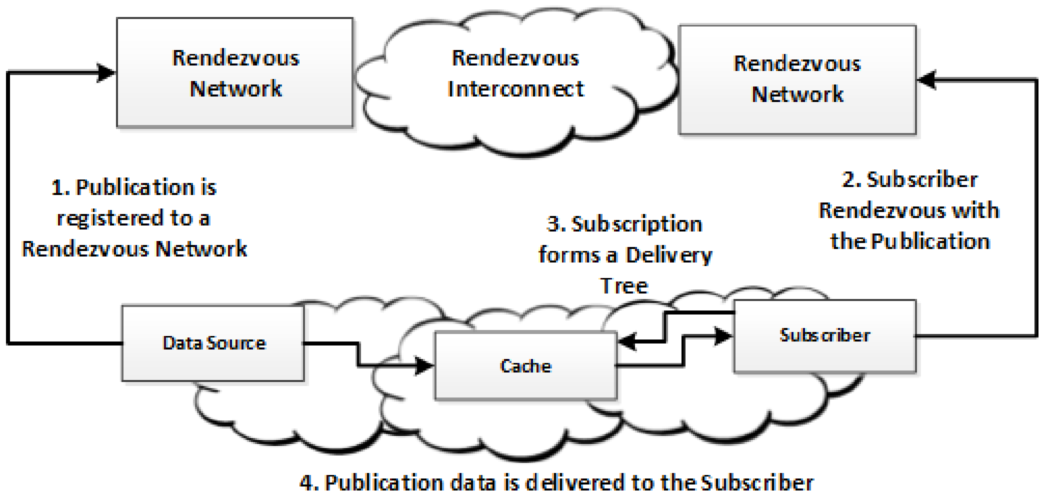

Rendezvous Network
Rendezvous Interconnect
Rendezvous Network
1. Publication is registered to a Rendezvous Network
3. Subscription forms a Delivery Tree
2. Subscriber Rendezvous with the Publication
Data Source
Cache
Subscriber
4. Publication data is delivered to the Subscriber

Figure 3.4: Rendezvous system in process

publications through the rendezvous service. In this way, the rendezvous service maintains a database containing all the publications labelled with flat labels (randomly looking labels) called RIds.

There will probably be a large number of rendezvous systems corresponding, for example, to various communities, networks, interest groups, and managing domains. Typically, the local rendezvous has to establish subscriptions to other rendezvous systems that the applications have expressed interest in. Whenever the rendezvous identifies a publication that has a publisher and one or more subscribers, it requests the topology manager function to construct the delivery tree connecting the publisher and the subscribers. In this way, rendezvous functions and topology functions constitute the control plane in the architecture. The rendezvous function has reached a mature stage but still needs work to provide scalability when working at the scale of the Internet [51]. The Rendezvous system is shown in Figure 3.4

The rendezvous system is identified as a well-defined zone that enforces policy decisions completed by the policy decision points in addition to a process allowing network endpoints to be able to make their own decisions. Additionally permitting the subscriptions and publications to derive together in the stated scope formerly rendezvous system may apply them to develop a part of the forwarding path to receivers/subscribers that is from the senders/publishers.

Now the forwarding path continues until it completes the state, on the assumption that there is at least one active subscriber/receiver and a publisher/sender through relating the RId to intra- and inter-domain identifiers. Hence dynamic policies like caching and inter-domain routing can be performed applying this way. A number of forwarding routers cache copies of publications to improve data accessibility and to reduce the load of referring to the publisher.

Briefly going into the security in PURSUIT the author's goals are to protect the data source and rendezvous system from unwanted traffic. [52] Rendezvous signaling messages are protected by PLA (Packet Level Authentication).

In today's internet information is assumed to be valid because the sender appears legitimate, where the Information Centric Security approach enables a much better level of security and trust. The security is based on the information itself rather than via authentication and encryption, the information objects carry security meta data for verifying the objects integrity and authenticity therefore objects are verifiable without having to trust the source.

3.2 Bloom Filter-based Forwarding

Bloom filter forwarding is used by the PURSUIT ICN architecture as the primary forwarding mechanism [5] and is the basis for the forwarding used in this thesis.

3.2.1 Bloom filters

Bloom filter-based source routing has been suggested as a solution for multicast forwarding [7]. Using Bloom filters to encode the multicast forwarding tree into the packet removes the need for the routers to maintain state per multicast tree. This solves an important scalability problem in multicast and can be used to enhance the control receivers and senders have over multicast communications. By decoupling group management from multicast routing, it also makes it possible for sources to deploy different end-to-end group management solutions, depending on need. We next introduce Bloom filters and then go through the relevant related work in using in-packet Bloom filters for packet forwarding.

A Bloom filter is a probabilistic data structure. It is an m-bit long array and initially, all bits are set to 0 as shown in Figure 3.5(a). Elements can be added into a Bloom filter by computing a set of array positions in the Bloom filter that are set to 1, as shown in Figure 3.5(b). The presence of an element is tested by checking if those array positions are set to one. This means that new elements can always be inserted into a Bloom filter, but no elements can be deleted. False positives are possible, i.e. a membership test can return positive even if the element has not been added to the Bloom filter.

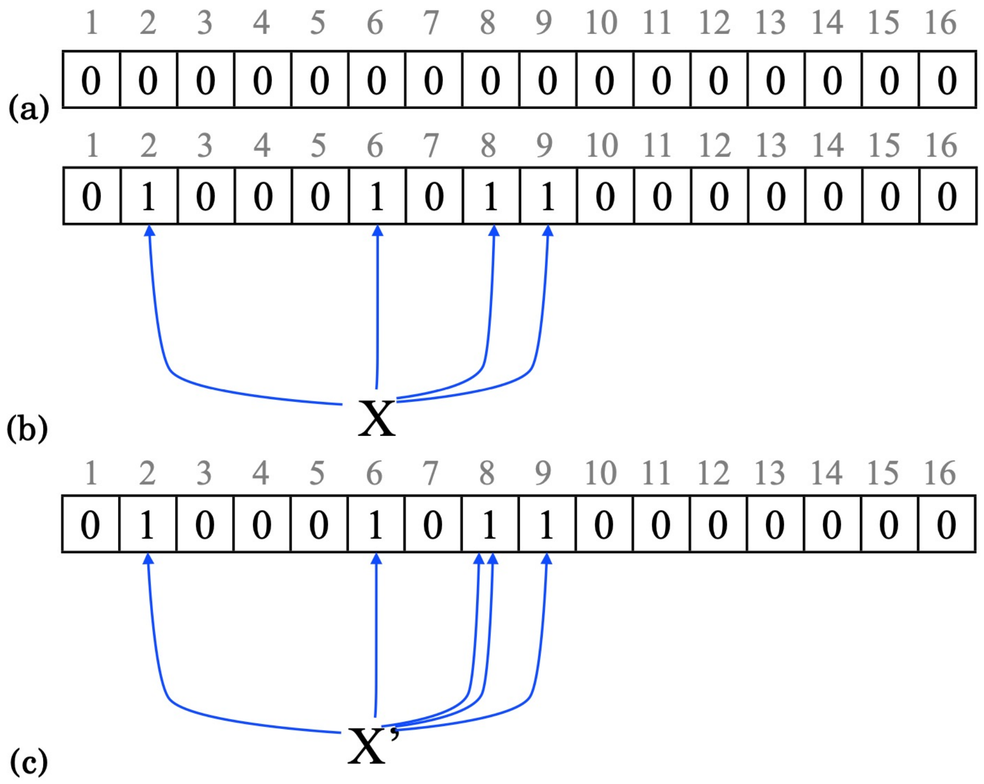


Figure 3.5: (a) Empty Bloom filter (b) X is added to Bloom filter by setting the hash value array positions to 1, $k = 4$. (c) shows a hash collision. Two hashes for element X' both yield location 8, $k = 5$

However, false negatives are not possible. Each element e is represented with k positions in the array. For example, k separate hash functions can be used to compute k array positions - each hash function giving and output value in the range $[0, m-1]$. The element can be encoded as an m-bit long vector in which the array positions denoted by the k hash values are set to 1 as shown in Figure 3.5(b). Two hash values can collide, as shown in Figure 3.5(c) bit 8. As the figure shows, $k = 5$ hash values are inserted into the Bloom filter, but only 4 array positions are marked to one. An element can be added to a Bloom filter by bitwise ORing the element's m bit vector together with the Bloom filter. The presence of an element is tested by checking if the k array positions are set to one. This can be efficiently done with the test $F \in B \quad | \quad m \wedge e == e$.

Figure 3.6 (a) shows a Bloom filter after elements X and Y have been added to it. The membership of an element, such as W, is tested by checking if each array position set to 1 in W is also set to 1 in the Bloom filter. The membership testing for W shows that W is not a member in the filter, since the bit in array position 10 is set to 0. When an element has not been added to the Bloom filter, but the array positions of the element are set to 1, a false positive happens. As an example, Figure 3.6 (b) shows a false positive. Only two elements, X and Y, have been added to the Bloom filter. F is denoted by the array positions 2, 4, 8, 9, which have all been set to 1 due to X and Y. Hence, membership testing will indicate that F is in the Bloom filter, while it has, in fact, not been added. In the context of multicast forwarding using Bloom filters, the process involves encoding a multicast tree each link on the forwarding tree is locally named a Bloom filter element. These links are

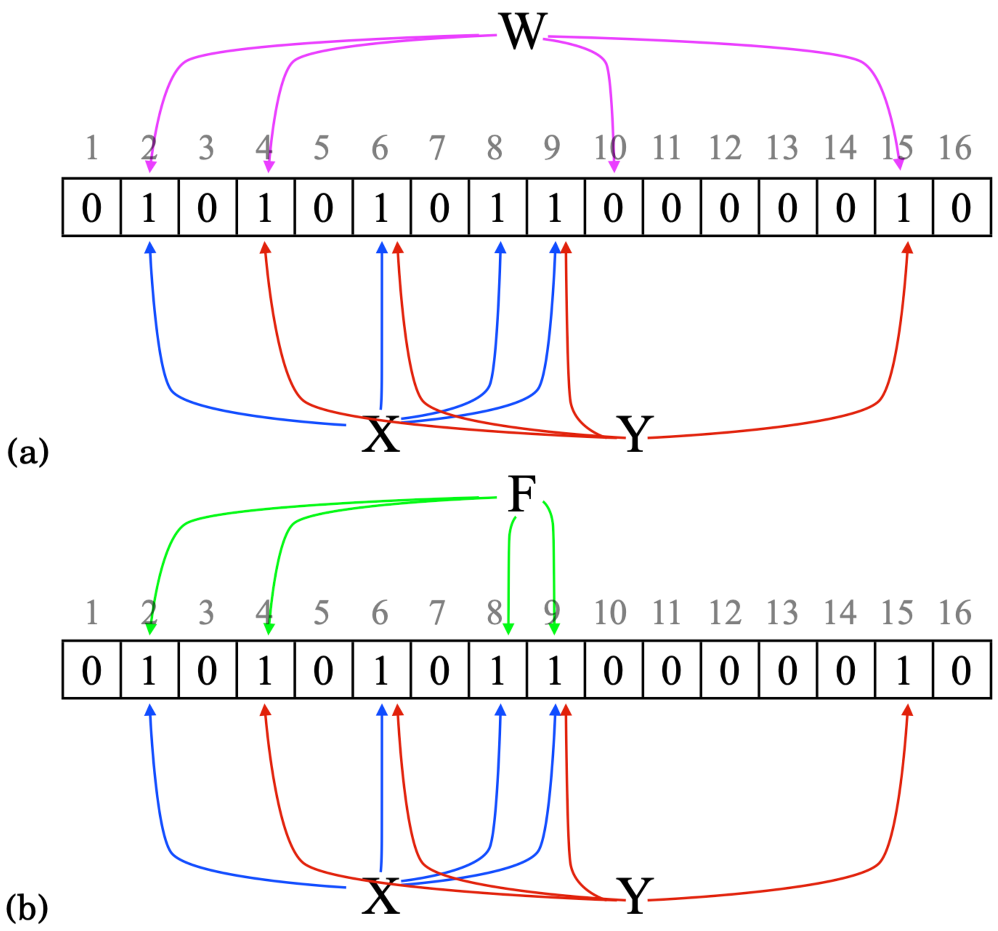


Figure 3.6: a) Shows a Bloom filter to which elements X and Y have been added. The corresponding array positions denoted by the blue and red arrows have been set to 1. The element W is not in the Bloom filter, since the bit in array position 10 is 0. (b) Shows the same Bloom filter and element F that has not been added to the Bloom filter. However, the test for membership indicates that F has been added, since all the corresponding array positions are set to 1. F is a false positive

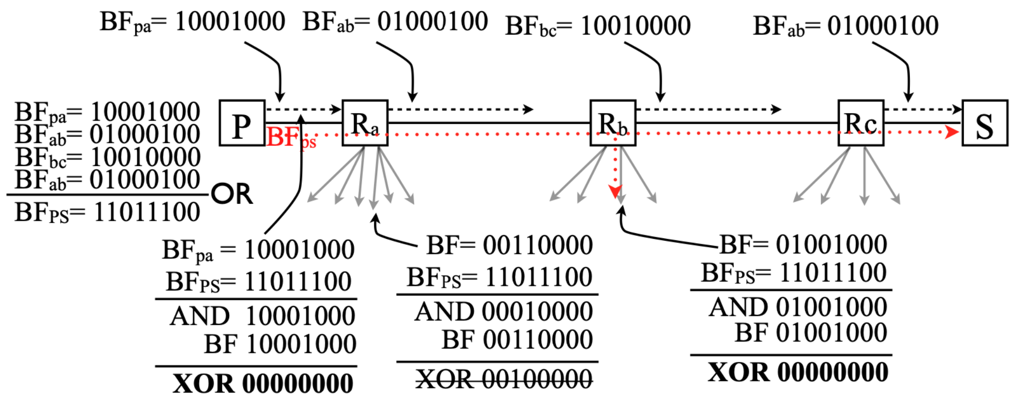


Figure 3.7: Bloom filter based forwarding. Each link has a directional identifier, e.g. BF ab . BF PS on the left shows the construction of a Bloom filter for path from P to S. The other columns show how each router tests the presence of a link in the Bloom filter. Red dotted line shows the path that the packet is forwarded.

then inserted into the Bloom filter, as shown in Figure 3.7. The resulting Bloom filter is placed into the packet header. This makes the Bloom filter a compact representation of the source tree from source to the set of receives. The routers forward the packet by checking which outgoing links are included in the Bloom filter, i.e. for each outgoing link a router makes a membership test. Each router names all its links locally with an m-bit long string with k bits set to one 3 with $k << \text{m}$. In some cases, Bloom filter with fill factor 0.5 represents a good choice of values. It is assume that each of the k bits is in a random position. Collisions are allowed. Hence, it is possible that the resulting string has less than k bits set to 1. The length (m) of the Bloom filter is constrained by the available size. The longer the filter, the larger the packet overhead is from using Bloom filters, but also more links can be encoded in a single Bloom filter. As an example, with $m = 256$,i.e. 32 bytes and $k = 5, \approx \begin{pmatrix} m \\ k \end{pmatrix} \approx 5 \cdot 10 \setminus 12$ different link identifiers. A Bloom filter for a multicast tree is formed by bitwise ORing the links together, as shown in Figure 3.7. Three variables that affect the fill factor are the number of elements n, the number of hashes used k, and the length of the Bloom Filter m. Choosing these three variables well, results in a Bloom filter that has a fill factor of ≈ 0.5. Intuitively, having a filter with approximately 50 % of array positions set to 1 maximises randomness and minimises redundancy in the Bloom Filter.

3.2.2 Bloom filter encoding a multicast tree

To encode a multicast tree as a Bloom filter, each link on the forwarding tree is locally named as a Bloom filter element. These links are then inserted into the Bloom filter. The resulting Bloom filter is placed into the packet header making the Bloom filter a compact representation of the source tree from source to the set of receivers. The routers forward the packet by checking which outgoing links are included in the Bloom filter, i.e. for each outgoing link a router makes a membership test. Each router names all its links locally with an m-bit long string with k bits set to one. Traditional IP Multicast architectures place the group maintenance at the routers (as described in Section 2.2.1. This requires routers to maintain per group state, which leads to additional complexity, and scalability issues. However, Bloom filter-based multicast [7] has been proposed as a way to create scalable multicast by pushing the group management to the source and thus removing the per-group state at the transit routers. In Bloom filter-based multicast, the links belonging to a multicast tree are encoded as a Bloom filter, which is then placed in the packet header. Each router checks for the presence of outgoing links and forwards the packet accordingly. Our work builds on this solution-based forwarding via multicast trees inheriting LIPSIN forwarding mechanism as follows:

LIPSIN [7] proposes a Bloom filter-based forwarding method for PURSUIT, a publish subscribe based internet, architecture [5]. The architecture is divided into three parts: rendezvous, topology, and forwarding, and explained in more detail in Section 3.1.1.

The novel forwarding fabric, termed LIPSIN, was firstly proposed by Jokela et al. [7] within the framework of the PSIRP project where the path and the links composing the path are expressed through Bloom filters. A Bloom filter is a probabilistic data structure that allows encoding the path in a Boolean vector [11].

3.2.3 Stateless Multicast Switching (LIPSIN)

In this section, we briefly explain the forwarding method, used in our work, a number of details such as two-direction traffic, loop prevention, and multiple link protection, which are expanded and described in Chapter 4. The most vital part of our forwarding method is adopted from [7]. LIPSIN uses LIDs encoded as m-bit length binary identifiers that are statistically unique into a Forwarding ID (FID) via Bloom Filter.

The forwarding path is performed in the following steps: an FID is created using a Bloom Filter via performing a bitwise OR of each LID in the path discussed in section 2.1 in detail. To forward the packet the FID is sent in the header that is then checked in each forwarding node. The forwarding decision for a particular packet is made by testing for a membership of the LID in a Bloom Filter contained in the packet header. Membership is accomplished by performing bitwise logical AND of each LID with the FID, if the result matches the LID, then the membership is confirmed then the packet is forwarded out of that link, if membership fails LID is not a match the packet is not forwarded on to that link.

In this section, we briefly explain the forwarding method, used in our work, a number of details such as two-direction traffic, loop prevention, and multiple link protection, which are expanded and described in section 3.2.

To encode a multicast tree as a Bloom filter, each link on the forwarding tree is locally named as a Bloom filter element. These links are then inserted into the Bloom filter. The resulting Bloom filter is placed into the packet header making the Bloom filter a compact representation of the source tree from source to the set of receives. The routers forward the packet by checking which outgoing links are included in the Bloom filter, i.e. for each outgoing link a router makes a membership test. Each router names all its links locally with an m-bit long string with k bits set to one 3, with k m.

The length (m) of the Bloom filter is constrained by the available size. The longer the filter, the larger the packet overhead is from using Bloom filters, but also more links can be encoded to a single Bloom filter. As an example, with m = 256, i.e. 32 bytes and $k = 5$, mk 5 1012 different link identifiers. A Bloom filter for a multicast tree is formed by bitwise ORing the links together.

For practical and security reasons, explained in detail below, it is useful to enforce a maximum fill factor, e.g. slightly above 50%. If the number of links is greater than can be inserted into the Bloom filter without exceeding the fill factor, at least three possible solutions exist as described below:

1. Using a longer Bloom filter. The capacity of Bloom filter, given a target false positive probability and k grows linearly with the size of the Bloom filter. Hence, by doubling the size of a Bloom filter, it is possible to approximately double the number of links the Bloom filter can contain.

2. Splitting the multicast tree into sub-trees that are small enough to fit in the field [7]. Arbitrarily large multicast trees can be supported by splitting them into several separately encoded multicast trees, each with its own Bloom filter. For large groups, this approach may be needed. The source will have to send a copy of every packet separately for each group.

3. Using virtual links [7]. It is possible to encode a path or a tree in the network so that it will be treated as a single link for the Bloom filter. As an example, an MPLS path or tree could be used as a single virtual link in a Bloom filter.

3.3 CCN Network Architecture

Above we have described the PURSUIT ICN architecture, but it is not the only ICN architecture, this section describes CCN and the next describes the closely related NDN architecture. Although these are not used in this thesis, they are the main competing ICN approach so it is important to include a review of this in the thesis.

CCN network architecture is focused on data rather than where the data is located, the aim of CCN is to replace the “where” (the data is), with “what” (is the data) without the requisite to distinguish the location which means the packet “address” names content and not location of the data.

This investigation has discovered numerous, current research projects starting from CCN and ICN all focusing on Publish/Subscribe paradigm on the basis of information/data in networking, mentioned above, the only difference is in its network architecture and approach taking to achieve networking based on information/data. This is since the demand is very high accessing data as fast as possible.

CCN Van Jacobson [53] argues that named data is a better abstraction in today’s communication problems than named hosts, and propose to shift the internet from a host centric to Information Centric Architecture. The authors of [54] claim that CCN achieves scalability, security and performance, through their results.

Jacobson [53] highlights the history of networking in three generation: (i) “Phone System” (Telephony) which focus on the wires, (ii) “Internet” (TCP/IP) focuses on connecting endpoints via existing phone system (iii) “Dissemination” focus on the data .

Content-Centric Networking (CCN) [55] defines a novel communications architecture built on named data - a packet ‘address’ names content, not location. The new waist’ of the stack (i.e., the traditional network layer) is now based on named data, and becomes the only layer requiring global

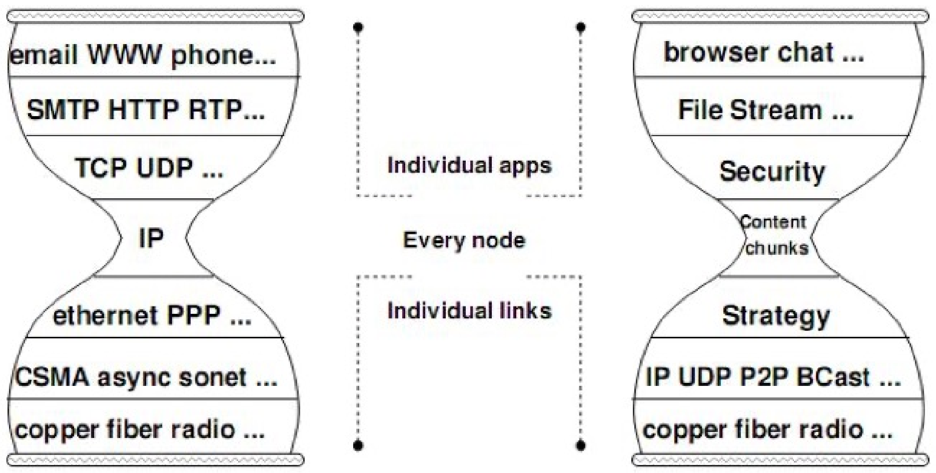
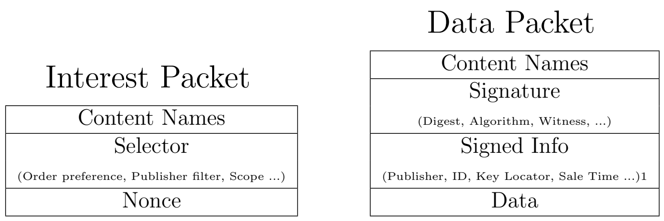


Figure 3.8: CCN replaces the global component of the network stack (IP) with chunks of named content.

agreement. Figure 3.3 compares the IP and CCN protocol stacks. Like in the traditional layers of the hourglass model of IP, layers of the stack reflect bilateral agreements (e.g., L2 framing protocol between ends of a physical link, L4 transport protocol between data producer and consumer). CCN preserves the design decisions that make TCP/IP simple, robust and scalable, including minimal demands on layer 2 (i.e., stateless, unreliable, unordered and best-effort delivery). As a consequence, CCN can be layered over anything, including IP itself.

The CCN architecture consists of two packet types, 'Interest' and 'Data'. This mechanism works through a consumer requesting for content by broadcasting its Interest in the internet/network, any node hearing the Interest which holds the data that matches the Interest, can respond with a Data


Interest Packet
Content Names
Selector
(Order preference, Publisher filter, Scope ...)
Nonce
Data Packet
Content Names
Signature
(Digest, Algorithm, Witness, ...)
Signed Info
(Publisher, ID, Key Locator, Sale Time ...)1
Data

(a) Interest Packet (b) Data Packet

Figure 3.9: Data & Interest Packet example

packet known as (content chunk). The Data is only transmitted if it was requested via Interest across the network [56]. Furthermore multiple nodes who are interested in the same content can share the transmissions over a broadcast link using standard multicast suppression techniques, given that the Interest and Data detect the content chunks are exchanged by name, hence the possibility of multiple nodes interested in the same content can share the transmission [56].

The drawback of this approach is request flooding inefficient resource utilization due to redundancy and scalability of data naming system.

In CCN network architecture, Data has to satisfy an Interest if the ContentName that is the (CCN name) in the Interest Packet as shown in Figure 3.9a on page 50 is a prefix of the ContentName in the Data Packet in Figure 3.9b.

The following displays the Internet Packet and Data Packet followed by example of a data name:

Packets are forward in CCN including former approaches, via content routers; it uses the Longest Prefix Match (LPM) when an Interest is being requested. LPM performs a look up match on the name, when a packet arrives on the interface and action is taken based on the results of that lookup. There are few ways of implementing LPM either in the hardware using Ternary Content Addressable Memory (TCAM) or in software using a multi-bit tree or a Bloom Filters.

Ternary Content Addressable Memory (TCAM) is a hardware device that supports high-speed table look-ups [57]. This is a smart solution for application such as packet forwarding and classification. Forwarding in CCN is based on three main data structures: FIB (Forwarding Information Base), ContentStore (buffer memory), and PIT (Pending Interest Table).

FIB (Forwarding Information Base) is used in CCN to forward the 'Interest' packets on the route to matching 'Data' requested by the user. A table of destination for Interests is organised for retrieval by the longest prefix match lookup on names rather than IP addresses, each prefix entry in the FIB can point to a list of destination rather than only one found in the current IP.

Content Store (CS) is almost identical as buffer memory of an IP router the only difference is that it has a different replacement policy. Buffer memory organises the retrievals by the longest prefix match lookup on names.

Pending Interest Table (PIT) is a table of sources that keeps track of Interests forwarded, it organises the Interest packets for retrieval by the longest prefix match lookup on names. CCN only routes the Interest packets. When transmitting the Interest Packet in the direction of potential Data sources, this leaves a trail of 'bread crumbs' for a matching Data packet to follow back to the original requester(s).

Therefore each PIT entry is a bread crumb and PIT entries are removed as soon as they have been used to forward a matching Data Packet in other words the Data 'consumes' the Interest. For Interests Packets that never find a matching Data are eventually timed out rather than being held indefinitely [56] CS and PIT are the same as buffer memory same contents different replacement policy, FIBs are also most identical except CCN has list of output interfaces (Faces), there is no TTL decrement since nothing can loop as CCN packets are never modified in transit.

CCN approaches a wide range of advantages involving content caching, the technique implies that the congestion is reduced and it improves delivery speed, in addition it requires a simpler configuration of the network devices and this will build security at the data level of the network.

However this change to CCN networking will require high speed hardware and software to support named based data forwarding and packet level caching. Today's hardware and software can only support a fraction of the routing state required aimed at forwarding operations at internet scale.

3.4 NDN Network Architecture

Similar to all aforementioned ICN architectures and unlike the current IP-based Internet, NDN focuses on content as the primary citizen in the network [37]. For that purpose, NDN names content items and uses those names for addressing, routing and retrieving data.

NDN advocates the use of human-readable and hierarchically structured names. In fact, a content name in NDN is represented as a "/" delimited path-like representation. For example, the name of UCLA's main homepage would be /ndn/ucla/home/index.htm.

On the other hand, large content such as videos can be split into chunks where chunk 50 of Bob's youtube video could possibly have the name: /ndn/youtube/bob/vid15.mp4/50. This hierarchy allows applications to represent relationships between data objects and more importantly allows for aggregation, which is a necessity for scaling.

There are two types of packets in NDN: interest and data packets. Since NDN is a consumer-driven architecture, a node interested in a particular content sends an interest packet specifying the name of the content needed. The packet is forwarded through the network until it reaches a node that has the content of interest. The data will then be sent, in a data packet, on the same path back to the original requester.

For packet forwarding, each node in the network has three main data structures: a Pending Interest Table (PIT), a Forwarding Information Base (FIB) and a Content store (CS). The PIT stores all the interests the node has forwarded but for which the corresponding data packet has not arrived yet.

The FIB is the routing table equivalent for NDN which is populated using a name-based routing protocol. As for the CS, it is the node's repository of cached content [39].

When a node receives an interest packet, it first checks if it has the requested content in its CS. If so, it will send a data packet back to the original requester of content. Otherwise, it looks up the interest in its PIT table. In the case when the interest does not exist, the node will add a new PIT entry which includes the interest name and the interface from which the interest was received, then it will forward the interest packet towards the data source using the FIB.

Otherwise, if the interest exists in the PIT, the node adds the incoming interface to the existing PIT entry and discards the interest packet.

A data packet, whether originating from the source or from an intermediate node is forwarded back to the requester using the PIT table entries, i.e., using the reverse path taken by the interest packet.

Specifically, when a node receives a data packet, it looks up its content name in the PIT. If the look up succeeds, the node forwards the packet to all the interfaces listed in the PIT entry and also caches the content. If not, the node considers the data packet unsolicited and discards it.

3.5 PURSUIT vs CCN/NDN for protection

The drawback of the CCN and NDN approach is that request flooding is inefficient with respect to resource utilization due to the redundancy and scalability of the data naming system [58]. Additionally, the nature of CCN/NDN forwarding is stochastic in nature using interest in the PIT so that it is difficult to embed fast deterministic protection paths, instead recovery from failure requires much slower updates of the PIT tables. In contrast, the PURSUIT ICN architecture uses deterministic "paths" set up by the topology manager so that it is possible to embed deterministic fast protection paths. Consequently, this thesis uses the PURSUIT ICN architecture to enable fast protection switching for ICN.

Chapter 4

Multicast Protection-cycles Through Path-Based Switching

4.1 Overview

This work is focused in ICN taking the PSIRP/PURSUIT network architecture approach in exploring how existing path protection mechanisms can be reused in ICN networks via Bloom filter. The work makes use of p-cycles [33], as introduced in Chapter 2 is incorporated to achieve fast path protection. However, unlike existing p-cycle work, the work of this chapter provides multicast protection as a natural part of the Bloom filter forwarding. Care needs to be taken to ensure that this protection does not introduce loops and additionally that it maintains the multicast-tree, assuming that this is possible given the alternative paths.

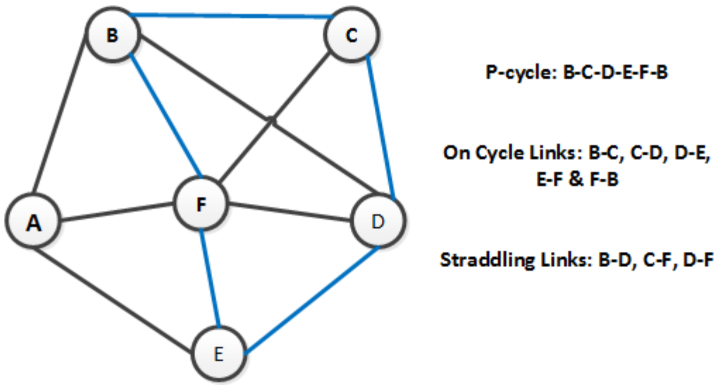
This chapter assumes the use of publish-subscribe (Pub/Sub) architecture using the PURSUIT/POINT model as described in in Chapter 2. Following that architecture subscribers and publishers are anonymous to each other and there are no explicit source and destination addresses as we find in the current network; instead the unicast path, or multicast tree, is uniquely defined using the Bloom Filter. This chapter will show how the p-cycle concept can be merged with the Bloom filter forwarding by the addition of a cycle identifier that is added during the redirection of the path/tree around a broken link. Additionally, it will be noted that this mechanisms could be implemented using an SDN infrastructure with no change to hardware switches. SDN infrastructure is described in Chapter 2.

This chapter is structured as follows in Section 4.2 supporting concepts includes path protection with p-cycle mechanism that we apply using path protection via cycle. Section 3.3 explains on the fly protection of bloom filter based flows and explains the path protection principle via multicolor cycle tree. Section 3.4 gives a formal description of the proposed protection mechanism using graph theory to explain the problem. The evaluation in 4.5 includes the explanation of the results that we present.

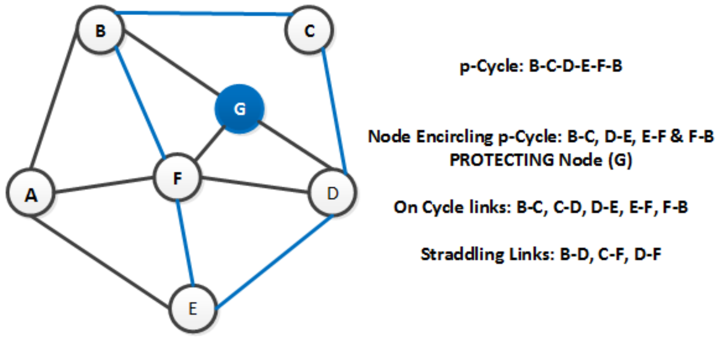
4.2 Supporting concepts

4.2.1 Path Protection with P-cycles

As described in Chapter 2, Grover and Stamatelakis first introduced the conception of pre-configured p-cycles in 1998 [59]. A p-cycle is a ring based protection scheme and is a promising protection technique, it achieves the speed of a ring with the efficiency of mesh [33], as described in Chapter 2. Here we use the concept of p-cycles to provide protection in the Bloom filter


B
C
A
F
D
E
P-cycle: B-C-D-E-F-B
On Cycle Links: B-C, C-D, D-E, E-F & F-B
Straddling Links: B-D, C-F, D-F

(a) Illustration of Link p-Cycle Protecting on Cycle links


B
C
G
A
F
D
E
p-Cycle: B-C-D-E-F-B
Node Encircling p-Cycle: B-C, D-E, E-F & F-B PROTECTING Node (G)
On Cycle links: B-C, C-D, D-E, E-F, F-B
Straddling Links: B-D, C-F, D-F

(b) Illustration of Node encircling p-Cycle Protecting Node G and on Cycle Links

Figure 4.1: P-cycles showing both on cycle links and node encirling links

forwarding, as used in ICN and described in Chapter 2. We should note that the aim here is to provide protection, not only for unicast traffic, but also for the multicast traffic that is switched efficiently using a Bloom filter.

Link failures are the most common failures that cause substantial network downtime as described by Turner et al. [60], due to a varying range of problems, but often due to failure in the link itself. When a link failure occurs, protection switching is performed at the two end nodes of the failed link. In the original p-cycle work the aim is to protect both on cycle links, straddling links and nodes as shown in Fig. 4.1. Noting that link failures are the most common, this work will concentrate on only protecting link failures. One reason for link failures being a more common and long-lived problem is that node failures can often be mitigated through duplicate equipment with the operator premises, whereas link failures are less under an operator's control (e.g. construction work along a fibre path that causes a catastrophic failure). For this chapter only on cycle-link failures will be considered. This means that every link (that has a possible cycle) will be on a cycle. The effect of this is to use what we will call *minimal cycles.* For example in Fig 4.1a, instead of the larger p-cycle B-C-D-E-F-B, instead we have the smallest possible cycles *e.g.* E-F-D which protects links E-F, D-F or D-E. Chapter 5 will go further by considering optimising the use of the cycles and thus will use straddling links and larger cycles.

In this scheme, at the event of a failure, the packets are re-routed via a p-cycle to the next point of failure, where the packets were intended to leave at. To employ the p-cycle technique in the Bloom filter mechanism we will introduce a cycle identifier (CID) as explained in the next section.

4.3 On-the-Fly Protection of Bloom Filter-based Flows

Fig. 4.2 shows a stylised network which will be used to explain the path protection principle. Consider the working tree (the blue tree) shown in Fig. 4.2 which subsequently has a link failure on the link x↔y. The p-cycle associated with the link x→y is x→y→i→j→k→x, noting that the cycle is in the reverse direction to the link that it protects. We assume that the

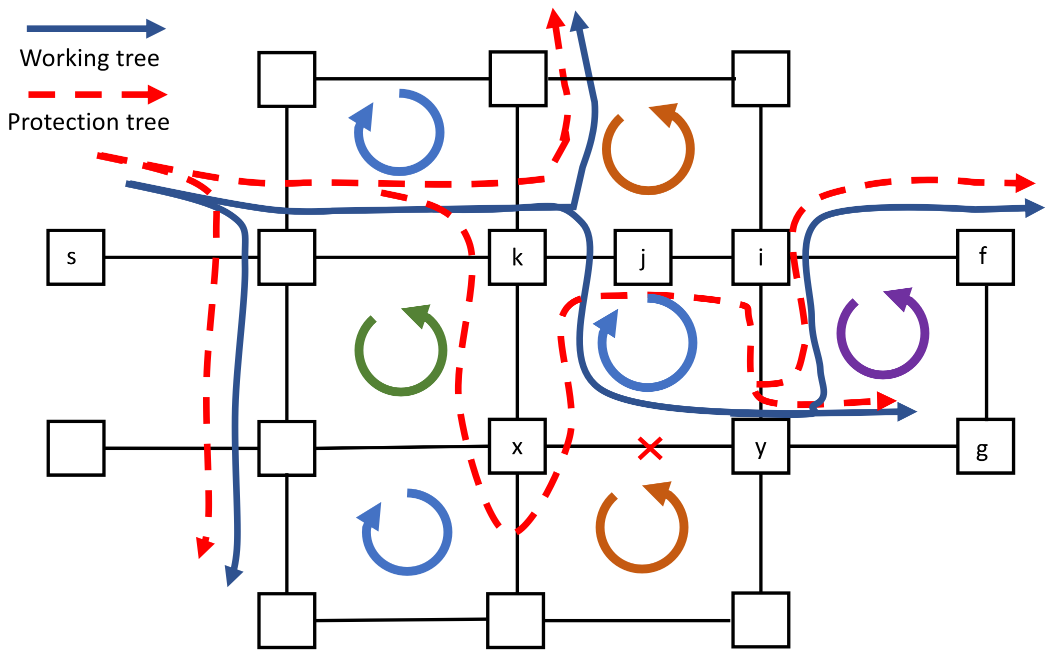


Figure 4.2: Path protection mechanism showing example working and protection trees.

p-cycles are all predetermined (and updated during topology change). There are a potentially large number of cycles and counting all of them is an NP-complete problem (a Hamiltonian is one such possible cycle and finding it or showing it does not exist is NP-complete [61]). However, in practice a set of *minimum* cycles that serve as a p-cycle for a particular link can be determined easily using Dikstra's algorithm by simply removing the link to be protected. Determining the optimum cycle for a link depends upon the optimisation criteria e.g., minimising path length and/or minimising capacity reserved for protection. Here we will assume that minimising protection path length is the criterion and refer the reader to Chapter 5 and also [62] as an alternative strategy that minimises reserved protection capacity. Consequently, we will assume a set of minimum cycles C_{min} that are formed from the set of all minimum cycles $\mathcal{C}$ as can be simply determined using Dijstra's algorithm and link removal.

To protect the link x→y it is necessary to send the traffic around the cycle x→k→j→i→y→x. The way this is proposed is to insert a cycle identifier (CID) in addition to the FID, so that now the header is {CID|FID}. When the failure occurs in the x→y the node x notes that any packets that are to be sent over x→y need to have the appropriate CID inserted and this will be continued until the failure is corrected, *i.e.* when the traffic reaches the end of the cycle. At the end of the cycle, the node (at the other end of the failure) notes that the cycle is identified with the link failure and as the node is aware of the failure (through link monitoring) it can remove the CID and forward the packet as normal. We should note that here we are assuming that there is only a single link failure. This is a reasonable assumption in most fixed networks at operator scale where typical link downtime is measured in minutes per year, thus to have two link failures at the same time is highly unlikely. If there were to be additional link failures on the same cycle, the system will do its best to forward packets from the first node on the cycle that has a link failure and the cycle will not continue to the second failure. Further work could look at mechanisms to cope with multiple failures on a cycle. Multiple failures in different cycles will be handled correctly by this mechanism. A further question is: where should this additional header be located? To answer this, we first need to investigate the size of the header as will be analysed below.

Note that in a large network there could be a very large number of cycles, in the example shown in Fig. 4.2 there are seven cycles. However, it is not necessary to have an equally large number of CIDs as it is only a requirement that the CID is unique on an edge, not for the whole network (as will be proved in Section 4.4). For example, in Fig. 4.2, although there are seven cycles, there only needs to be four unique CIDs as indicated by the four different colours used to identify the cycles. In fact, for all planar graphs, as with the graph in Fig. 4.2, there need only be up to four unique CIDs as will be formalised in the next section. As noted earlier, the Bloom Filter based forwarding of multicast/unicast traffic can be implemented in practice in existing SDN capable switches with substantially less overhead than even conventional switching mechanisms (e.g. MAC based or IP based switching) [21]. While [21] showed how the standard FID can be encoded by overloading the existing IPv6 header, it did not address protection switching. We note here that it would be straightforward to overload the MPLS header field in the Ethernet shim label [19] which is implemented from OpenFlow v1.1 [63].

4.4 Formal description of the proposed protection mechanism

We will consider a graph $G(V, E)$ and an edge as the ordered pair $f = (x, y)$ which is part of a directed and ordered protection cycle c where

$$c = \{\{(x, y), (y, i), (i, j), \ldots, (k, x)\} \in E\} \tag{4.1}$$

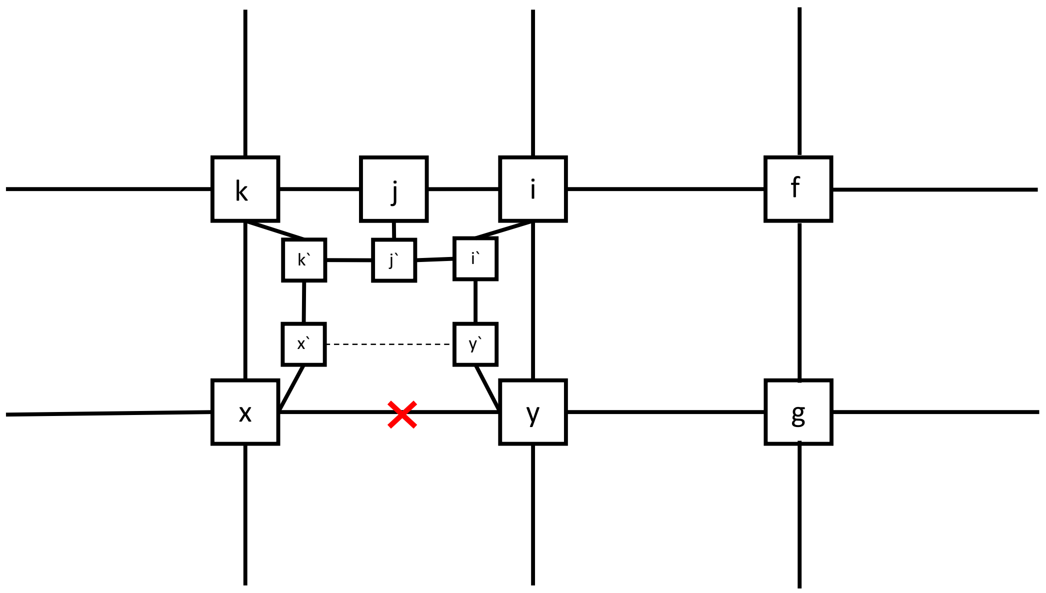


Figure 4.3: Path protection mechanism showing how the protection cycle can be modelled as an augmented graph

noting that not every edge in G may have a valid protecting cycle. To be precise, using our definition of c above, f is actually protected by the reverse cycle which we denote c'. Note the notation used here matches the example shown in Fig. 4.2.

Consider that the protection mechanism proposed in Section 4.3 can be represented as an augmented graph $G'(V' \bigcup V, E' \bigcup E)$. The modification adds virtual nodes $V' = \{v \in V' | v \notin V \ \wedge V' \longrightarrow c'\}$ that have a one-to-one correspondence (termed a *bijection*, $\longrightarrow$) with the cycle c' nodes. Using the notation above for c, and noting its reverse c' is the bijection to V' then we can define the new protection cycle as $b' = \{(x', k'), \ldots, (j', i'), (i', y')\}$ where the $s' \in V' \longrightarrow s \in c \in V$. An example of such a modified graph is shown in Fig. 4.3 which matches the earlier example in Fig. 4.2. Additional edges provide the protection cycle and exits from the protection cycle so we have the final definition of $E' = \{(x, x') \bigcup b' \bigcup (y', y)\}$. We then note that a packet has a FID which operates on the graph (G or G') to select forwarding edges from each node, we denote the forwarding action, F, that defines a sub-graph P on a graph G as $P = F(G)$ and note that in the case of a multicast delivery P is a directed acyclic graph (DAG) [64]. We define that P has a source s and destination nodes $D \in V$. During a failure condition we have an alternative tree $P' \neq P$ with the same source S and destinations D'. For the failure to be successfully protected we require that the failure tree is fully connected between the source s and the original destinations, *i.e.* $D = D'$, and that the protection mechanism does not introduce loops into the network, *i.e.* P' is a DAG.

Lemma 1. *In the case of a failure of edge f on a DAG $P \subset G$ (G not DAG), where f is protected by cycle c', and P results from the forwarding action $P = F(G)$ connecting the source s to destinations D; then, the resulting protection tree $P' = F(G', f)$, which connects S to destinations D' through the protection graph G', is a DAG and $D' = D$.*

Proof. As, by definition, P defines a DAG on G we denote an order to each node $s \in P$ starting from s. Consider that under the failure condition on $f = \{x, y\}$ that the associated reverse edge $f' = \{y, x\}$ is also disconnected and that traffic in P that matches the link f is automatically forwarded into the associated protection cycle b' and has the associated CID set (*i.e.* the CID is not zero). The original tree P can be broken into three parts, P_A, part of the tree that is before the failed link (*i.e* a tree pruned at $x, y,$), the failed link $\{x, y\}$, and the remainder of the tree, P_Ω, thus $P = \{P_A, \{x, y\}, P_\Omega\}$.

Without loss of generality, as P is a DAG, we can label all of the nodes of this tree in an ordered sense [64] starting at s such that P_A is less than the label of x, that X is labelled with $|P_A| + 1$, y we choose to label with $|P_A| + 1 + |b'| + 1$ and the labels of P_Ω are ordered and all greater than $|P_A| + 1 + |b'| + 1$. Now consider the protection tree P', as b' provides an alternative path linking x and y, we can say that P' is fully connected and consists of a tree $P' = \{P_A, b', P_\Omega\}$, thus $D' = D$. The order of labels in P_A and P_Ω have not changed and the labels in b' can be ordered naturally along the cycle such that they are greater than the label of x and less than the label of y. Thus, the labels of P' are ordered which means that it is a DAG [64]. □

To provide connectivity through the protection cycle we have introduced a new label to the packet header that identifies the cycle. This is to ensure that the packets are forwarded on the cycle. It is important that the packet header is kept reasonably short, we have already used some of this resource for the Bloom filter for the multicast tree. The question then is: how many additional bits are required for the protection cycle identifier? We can find a bound for this in the following Lemma 2 which uses *graph colouring* theorems, and the later Corollary will tighten this bound for most networks.

Lemma 2. *An upper bound to the number of unique CIDs is given by the largest node degree, Δ, or $\Delta + 1$ in the case of a full mesh (a complete graph).*

Proof. We note that a unique CID for each cycle passing through a node can be satisfied by a graph colouring problem of a unique colour of a neighbouring vertex, the chromatic number of the graph. Brook's theorem [65] states that the chromatic number of a graph is bounded by the maximum node-degree, Δ, for all graphs except a fully connected graph or a cycle with an odd number of nodes in which case it is $\Delta + 1$. In the case of a cycle we note that only one CID is actually required as there is only one cycle in this graph showing that Brook's theorem is a pessimistic bound in at least this case. Thus, Brook's theorem satisfies the constraint for the CID numbering problem, but is not exactly equivalent, as shown in the cycle case. □

It has been noted that in most cases real-world graphs are planar graphs [66], and in this case we can significantly reduce the bound presented in Lemma 2 as follows below.

Corollary 1. *For a planar network the upper bound of the unique CIDs is four.*

Proof. As with Lemma 2 we note that the problem can be satisfied by the graph colouring problem, i.e. the chromatic number of the graph. For a planar graph the chromatic number is at most four as proven from the four colour map problem [67]. □

Corollary 1 is useful for planar graphs, but does not answer the question for non-planar networks. The following evaluation section will consider experimental analysis for non-planar networks and also consider how many graphs are likely to be non-planar.

4.5 Evaluation

In the evaluation we have, mainly, concentrated on the difference between implementing multicast forwarding using either a Bloom Filter or using MPLS. In terms of protection switching times, these have not been evaluated as they are strongly dependent on hardware implementation issues. However, generally, hardware systems that simply note a hardware failure, update a forwarding table and redirect a packet can generally operate in a number of milliseconds. As both the Bloom Filter mechanism and MPLS require a similar order of table operations the difference in protection switching times between these two is likely to be minimal, and probably dominated by the detection of the failure which will be the same in both instances. Thus, we proceed by firstly enumerating key operational parameters such as the number of CIDs and false positive performance of the Bloom Filter switching; secondly we compare key performance issues between the Bloom Filter and MPLS multicast implementations.

Fig. 4.4 shows a scattergram of the maximum number of cycle *colours* for each of the networks within the Internet Topology Zoo network database [68]. For brevity, when mentioning specific networks by name it should be assumed they are from this database. As proven in Section 4.4 the number of unique CIDs required in each network is directly related to the notion of *graph colour* when treating each cycle as a region. Planar networks conform to the classic Four Colour Map theorem [67], and Bowden *et al.* show that for the 147 networks in the Internet Topology Zoo only 14% (21) were non-planar. Thus, for most of the networks that exist (and with some conjecture, are likely to exist) only require four unique CIDs. For non-planar graphs, the

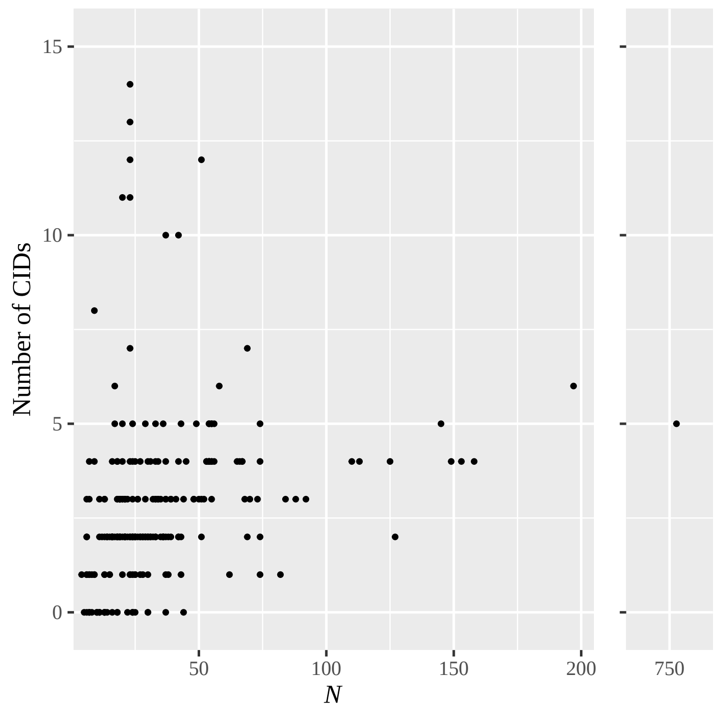


Figure 4.4: The number of unique CIDs for networks from the Internet Topology Zoo against number of network nodes N

map colouring problem is non-trivial, in fact is has been shown to be NP-complete [69]. Consequently, to check the number of CIDs for the all the networks (including the non-planar networks) in the Internet Topology Zoo dataset we need an algorithm that can find an upper-bound; we have selected the greedy DSatur algorithm [70] as it is readily available. Consquently, taking all of the Internet Topology Zoo (now expanded to 225 networks) and allocating p-cycles that correspond to minimal cycles, the number of CIDs was enumerated and shown in Fig. 4.4.

In this case the p-cycles were optimised to reduce the likely protection traffic using the algorithm by Drid *et al.* [62]. The results shows that the

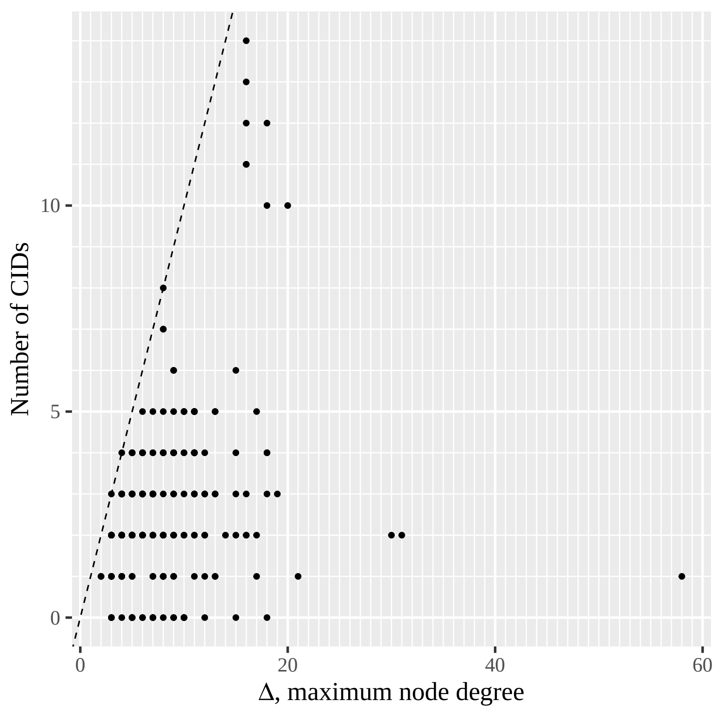


Figure 4.5: The number of unique CIDs for networks from the Internet Topology Zoo against maximum node degree Δ

number of graph colours is mostly four or less, as would be expected. Out of 225 in the Internet Topology Zoo, 24 had no valid protection cycles (they were pure trees, and are indicated with Graph Colours of zero), 171 had four or fewer unique CIDs, 30 required more then four unique CIDs. Interestingly, those that required more than four where not the larger networks, rather they were unusual in some aspect, *e.g.* full-mesh (as with GlobalCentral) or implemented with existing 1:1 protection links. A good example of the latter being Belnet 2004, which is star network with 1:1 protection needing 14 unique CIDs, but which by 2010 had transitioned to a mesh network needing only two unique CIDs; one might point out that if the network has

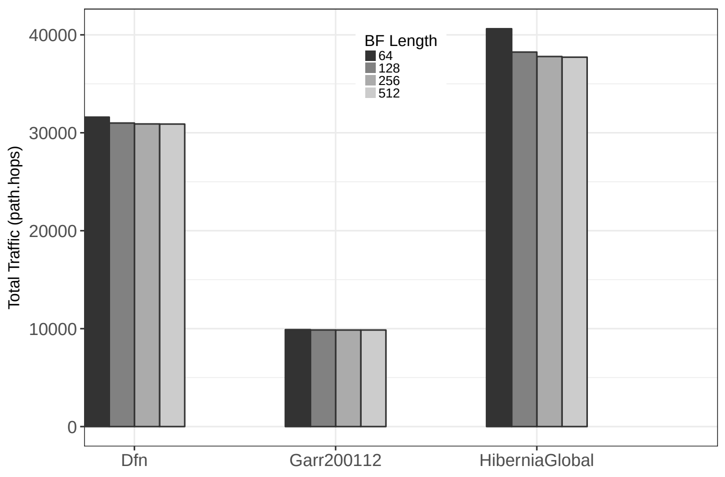


Figure 4.6: Total Traffic for 1000 Multicast Tree using ICN. Traffic variation by Bloom filter length and different networks. Traffic is measured by the number of hops in each path (path.hops).

1:1 protection then the p-cycles would not be needed anyway. The largest network, KDL, needs only five unique CIDs. Fig. 4.5 shows how the number of CIDs compares to the maximum node degree, Δ, which is a bound on the number of CIDs for any graph (except full-mesh) as given in Lemma 2. As suspected from the construction of the Lemma, this is a pessimistic bound for most cases. However for three networks the number of CIDs does match this bound. In one case, Globalcenter network, it is a full-mesh and thus the number of CIDs is one less than the bound.

Use of a Bloom filter for forwarding can result in false positives. We investigated this for 2000 randomly generated multicast trees and each case with different Bloom filter lengths and across three Internet Toplogy Zoo networks (node degree N, diameter D): *Dfn* (N=58,D=6), *Garr200112* (N=24, D=3) and *HiberniaGlobal* (N=55,D=16). The graphs were chosen to reflect a range of network types, but avoid star (or mostly star) networks which have little opportunity for mesh protection. The trees were generated using a Zipf distribution for source node selection and tree size. This is motivated by observations that source selection tends to follow a Zipf law and this approach has been used by others investigating multicast trees [71]. While this does not significantly affect this result, it will be important in later results. If there are false positives in the Bloom Filter, this results in higher traffic as additional traffic is sent on links that are not needed (as with aggregated MPLS that we will discuss later). Fig. 4.6 shows the results for this. As can be seen there is a substantial difference between a 64-bit Bloom Filter in the HiberniaGlobal network and longer Bloom Filters. Generally, the traffic is higher in this network as it is the network with the largest diameter, so when measuring traffic as the total hops in all the paths it is higher as the paths tend to have a larger stretch. Additionally, with the larger stretch comes more opportunities for false positives and hence the greater diversity in traffic compared to Bloom Filter length. It is notable that there is generally little benefit in increasing beyond 256 bits for the Bloom Filter length, and this is a useful length as it has been shown to be implementable in existing SDN switches [21].

The remaining set of results compare the state and traffic with Bloom Filter based protection mechanism against a multicast implementation that uses MPLS. MPLS does not natively support multicast and consequently there are a number of approaches to transporting multicast [72], essentially these are a compromise between minimising state or minimising traffic. In the two extremes either all traffic is duplicated from the multicast source (min-

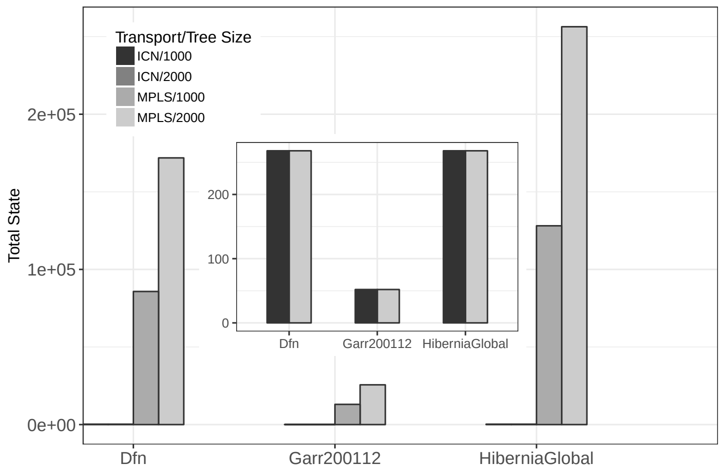


Figure 4.7: Total State for 1000 and 2000 Multicast Trees for MPLS and ICN with AD=0.5

imising state), or alternatively tunneling state is needed as every multicast replication point (minimising traffic). In practice operators may choose to implement trees that *aggregate* a set of multicast trees to reach a compromise between state and traffic. This compromise is described in the key MPLS implementation standards [72] but more clearly enumerated by Martinez-Yelmo *et al.* [73].

The Figs. 4.7 and 4.8 show the key differences between the Bloom Filter and MPLS multicast implementations when using a mid-way compromise for the MPLS trees, an *aggregation degree* of 0.5 in the terminology of Martinez-Yelmo *et al.* [73]. Figure 4.7 shows a key difference between the Bloom Filter and MPLS techniques with regard to the switching state when implementing either 1000 or 2000 multicast trees. While the number of trees was chosen somewhat arbitrarily, it should be noted that a typical catalogue size of streaming services is in the order of 1000; although with the provision virtual

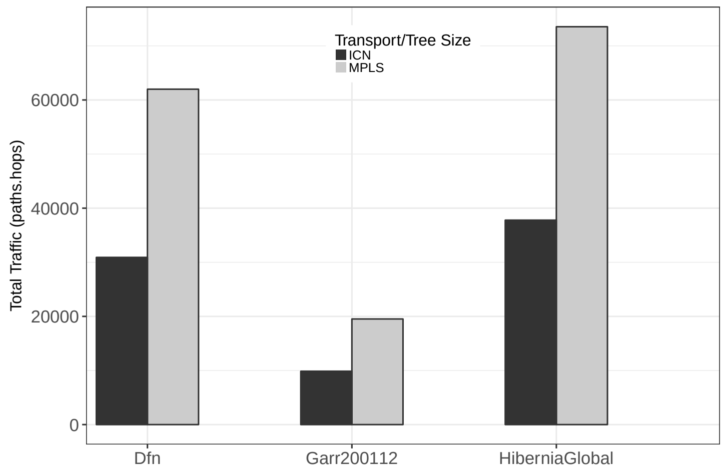


Figure 4.8: Total Traffic for 1000 Multicast Tree for MPLS and ICN

private multicast trees this number could grow much larger. In particular it can be noted that the state size is, comparatively, very small for the Bloom Filter implementation (as duplicated in the inset graph) and does not increase with the number of trees; meanwhile, the MPLS state size is many orders of magnitude higher and is, unsurprisingly, proportional to the number of multicast trees. Difference in traffic is shown in Fig. 4.8 showing that MPLS has, very roughly, twice the volume of traffic compared to using Bloom Filter switching for the aggregation degree of 0.5; this is because of the compromise through tree aggregation in MPLS.

In practice, an operator might wish to change the aggregation degree used in their multicast deployment, and this changes the compromise between traffic and state. The results in Figs. 4.9 and 4.1011 show this compromise between state and traffic, respectively. Fig. 4.9 shows that for the highest aggregation degree of 1 the state in MPLS is a similar order to that of using Bloom Filter approach, whereas with an aggregation degree of zero MPLS

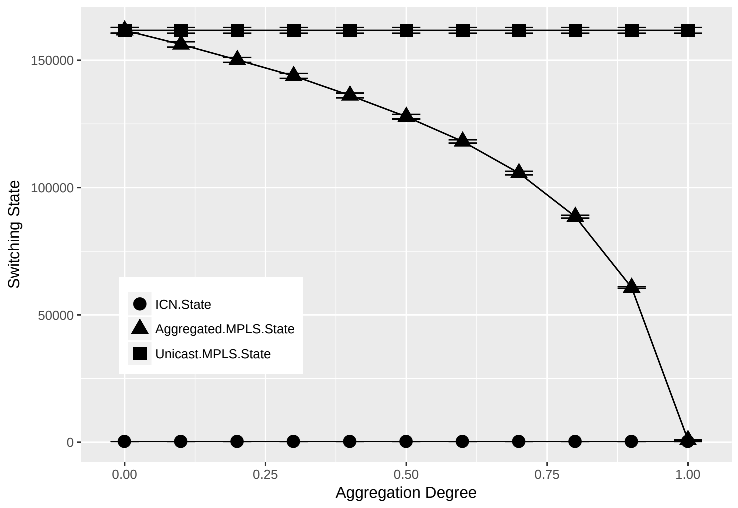


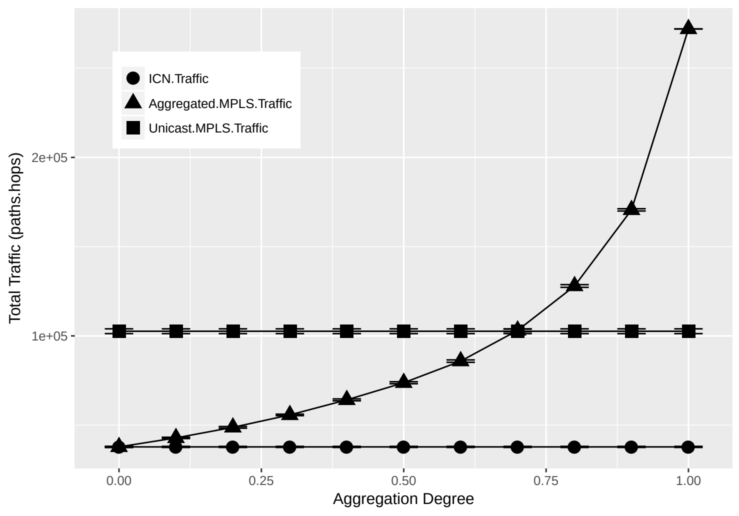
Figure 4.9: Total State for 1000 Multicast Tree using either MPLS or ICN in HibernianGlobal network

is the same as implementing the multicast as unicast (as indeed this is the implication). The error bars show 95 % confidence intervals.

4.6 Practical implications

4.6.1 Header insertion

Having now investigated the likely number of CIDs we can now answer the question about the practical implementation of the CIDs. We note that, at first site, it is not obvious that the Bloom filter forwarding is compatible with existing switching systems. However, on closer analysis is has been found that the Bloom filter forwarding can be directly implemented on existing SDN hardware as demonstrated by [21]. This mechanism uses the IPv6 address space to implement the Bloom filter FID and uses the MAC address



11 corrected figure

Figure 4.10: Total Traffic for 1000 Multicast Tree using either MPLS or ICN in HibernianGlobal network

to separate the use of the IPv6 address space for Bloom filter forwarding from any "normal" IP routing, so that the two mechanisms can operate independently on the same switches. This then leaves the question about the CID field. Fortunately, this can be simply implemented using the MPLS header feature in SDN switches. This is very convenient as the switches support "pushing" this header on and "popping" this header off very easily [19]. The MAC Ethertype field is also used to denote MPLS Ethernet frames so that this can also be easily used to mark the packets as *on the protection cycle*. Additionally, the 10-bit MPLS header is much larger than the CID requirements of 16 values as was suggested by the analysis so that it would be possible to allocate a small part of the MPLS label space to the CIDs while allowing existing MPLS forwarding to exist at the same time.

4.6.2 Dissemination of cycle IDs

The cycle IDs need to be distributed to the nodes from the topology manager as this entity has knowledge of the links within the network. Thus, the topology manager is the entity that will carry out the algorithms associated with the cycle ID calculations (and optimisation as described in Chapter 5). One important consideration is: how often would cycle IDs have to be optimised and distributed? Certainly, new cycle IDs need to be computed when the network topology changes. In the context of large-scale transport networks, changes in physical topology occur in the context of days rather than seconds, in other words: major changes to topology occur infrequently [74]. Simply adding or removing leaf nodes in the network has no significant effect on the cycle IDs as only adding or removing links that change the minimum cycle set has an effect. In Chapter 4 we will see that the optimisation of the cycles takes into account the traffic demands on the links to ensure the minimum "overbooking" of link capacity to allow for link failures. While this could be computed continuously, this would generally be wasteful as it is only the peak traffic that has a significant effect on the computation: i.e., if we ensure capability for the peak load it will ensure there is capacity for lower loads. An in-depth guide to traffic matrices in networks is provided by Tune and Roughan in 2013 [74] and while this is now 10 years old, the anecdotal reports from operators are that, while the trend has been moving upwards, the general day-to-day patterns follow similar trends. Tune and Roughan show that traffic patterns tend to peak once each day and are generally stable over a few hours of the peak period [74]. Thus, we would propose that cycle IDs are optimised once a day based upon recent days' traffic patterns ahead of the forthcoming peak traffic and probably distributed in early morning periods when there is often an "at risk" period for possible maintenance tasks and when demands, and thus failure impacts, are low. Given that the assignment of cycle IDs is a relatively straightforward task, the overall computational impact of this work on the topology manager performance will be negligible.

4.7 Summary

This chapter has introduced the proposition of using p-cycle switching in combination with the PURSUIT ICN architecture. It has shown that it can have significantly less switching state and traffic overhead than MPLS. This was demonstrated by simulating the network scenario in a number of realistic networks. The work has shown that there can be a provable bound on the additional switching overhead in terms of switching state and packet header size represented by the CID; in the case of a planar network this is limited to simply four different identifiers, i.e., two bits. In the case of non-planar networks the highest number of unique CIDs was found to be 14.

Chapter 5

Optimising protection cycles

In Chapter 4 the simplest, smallest protection cycles were used, every node in the network was included in a protection cycle, assuming that a possible minimal cycle exists. The main difference between p-cycle and ring protection is that, p-cycle protection not only protects the cycle links but protect all links whose end nodes belong to the p-cycle. In other words, p-cycle protects the cycle links and the straddling links in the cycle. A straddling link is a link which does not belong to the p-cycle but whose end-nodes are both on the p-cycle. While this was not the case in Chapter 4 as minimal cycles were chosen, a larger p-cycle can allow the protection of a straddling, as described in Chapter 2. Each p-cycle can offer two restoration paths to the failed straddling links without requiring any additional spare capacity. This propriety reduces effectively the required protection capacities. Thus, the hypothesis is that larger cycles can be more efficient. The major problem of this method of protection resides in finding the optimal set of p-cycles which protects the network for a given working capacity distribution.

The method to generate p-cycles with better efficiency is inspired by the work of Drid et al [62] and has two stages: in the first stage the set of shortest cycles is generated (as used in Chapter 2), in the second stage, the cycles are incrementally aggregated to look for more efficient cycles.

5.1 Overview of the approach

In Chapter 4 we assumed that the protection cycles where drawn from the set of minimum cycles $\mathcal{C}$, where these are defined as the smallest cycle that can be used to protect each edge (assuming that each edge has a cycle to protect it). We noted there that finding the optimum set of cycles is a different matter and depends upon the optimisation criteria. In Chapter 4 this was assumed to be simply the smallest cycle. This is often an important criteria in optical networks where it can be difficult to guarantee physical layer constraints. However, here we are using electronic switching, in this case smaller cycles might be useful to minimise latency, but otherwise it is less important to minimise the cycle length. Instead a more useful metric is to minimise the amount of capacity assigned for protection formulated as a relative *redundancy overhead*, *R*.

The *redundancy overhead*, *R* can be expressed as below:

$$\min(R) \text{ s.t.} \tag{5.1}$$

$$R = \frac{\sum_{e \in E} a_e}{\sum_{e \in E} w_e} \tag{5.2}$$

$$a_e = \arg\max_{i \in c}(w_i), \quad e \in c, \ (i, c) \in A \tag{5.3}$$

where a_e is the allocated capacity on edge e for all the protection cycles that include e and w_e is the working traffic allocated to edge e. Eq. 5.3 defines a_e as the maximum working capacity of any edge using the cycle c that protects edge e and noting that there could be more than one cycle that uses e. Here we introduce the set A as the set of tuples (e, c) that denote that e is protected by cycle c, noting that each $c \in (e, c) \in A$ is unique, but, that there are multiple edges typically allocated to each cycle. We can informally state additional constraints for (5.2) are that any cycles used to form protection must maintain connectivity for the traffic matrix T under the condition of a single edge failure on edge e. In practice, not every edge may be protected by a cycle, but we will simply ignore these edges in the formulation as it does not change the exposition.

Fortunately, it has been proven that a ring-like structure, such as that used by p-cycles, is the most efficient protection structure [33]. Unfortunately, the aforementioned work by Stamatelakis and Grover does not determine the optimum set of cycles, which we will prove is NP-complete below, in Theorem 1. The aim of this chapter is to find a good set of cycles to lower R compared to a trivial cycle set such as a minimal cycle set [75]. The probability of increasing the traffic to use this increased protection switching cost is dependent upon the link (or node) failure rate in the network.

3 cost to network The value R represents the cost to the network to introduce the protection scheme, and this is the most significant cost of the scheme; we will see in Section 5.2.1 that the algorithmic complexity added to the ICN topology manager is relatively small. Specifically, R is the increase in planned capacity to enable protection switching. If the operator is to offer

protection switching to cope with a link (or node) failure there is no option but to add this planned protection cost. The p-cycle scheme here assumes that there will be at most one link (or node) failure at any one time. Given the typical availability of modern core networks this is a usual assumption as the probability of two link failures affecting the same network is very small. Obtaining figures for real network failure rates is difficult to obtain as this is often protected business knowledge. However, a good study on failure rates in practical carrier networks is given by Mello et al. [76], their study shows that for typical carrier networks (they investigate a national Italian and USA network) a single link failure probability rate, at some point in the network, is of the order of between $10^{-1} - 10^{-2}$ with repair times approximately 12 hours for each failure. While the actual probability of failure and repair times vary greatly, Mello et al. [76] show that these assumptions – from working knowledge of practical networks — lead to the 0.99999 reliability figures which are commonly used for carrier networks. Thus, we can assume that the network will be called on to activate the protection switching a small number of times each year and in each case for the order of 12 hours [76].

An example showing how the differences in cycle association with edges gives rise to different redundancy overhead is shown in Fig. 5.1. Assuming working traffic is assigned to each edge then we can assign the required spare capacity that must be allocated to allow for the possible use of the p-cycle. For example in Fig. 5.1 (b) edge (b, e) needs four units of traffic and thus all the other edges on the right cycle need four units of additional capacity. The edge (b, e) only needs three units of protection overhead as the remaining edge with the largest traffic in the cycle is (b, c) with three units. Alternatively,

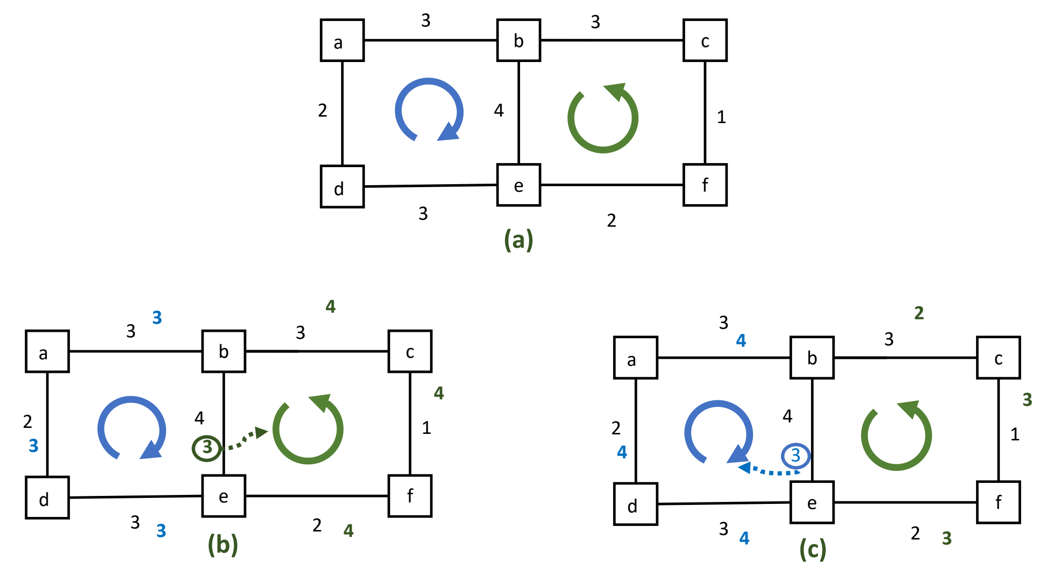


Figure 5.1: Example showing redundancy overhead and differences in allocation of redundancy for different edge/cycle selection: (a) shows the working traffic, (b) allocates edge (b, e) to the right cycle, (c) allocates edge (b, e) to the left cycle; (b) and (c) show both working traffic and the redundancy overhead in the corresponding colour.

as shown in Fig. 5.1 (c) edge (b, e) is allocated to the left cycle so now all the other edges in that cycle need four units of traffic. We can now compute the redundancy overhead for the scenario Fig. 5.1 (b), R_b and Fig. 5.1 (c), R_c as:

$$R_b = \frac{3+4+4+4+3+2+4}{3+3+1+2+3+2+4} = \frac{24}{18}$$

$$R_c = \frac{4+2+3+3+4+4+3}{3+3+1+2+3+2+4} = \frac{23}{18}$$

Thus, we can see that the scenario in Fig. 5.1 (c) has lower redundancy overhead and would be the preferred choice.

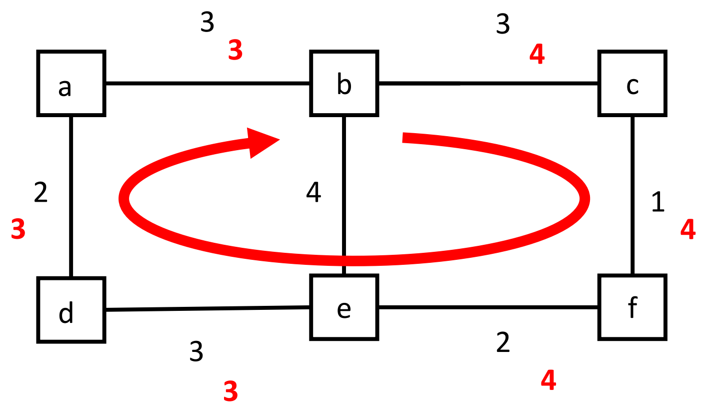


Figure 5.2: Example of aggregating the two earlier cycles from Fig. 5.1 into one larger cycle

Fig. 5.2 However, there is one other possibility of choice of cycle as shown in Fig. 5.2. Here the two cycles shown in Fig. 5.1 are aggregated into one larger cycle; using the p-cycle terminology, edge (b, e) is converted to a straddling link. Now the capacity for edge (b, e) still has to be covered by edges (b, c), (c, f) and (f, e), but edge (b, e) does not have to cover any protection traffic. Now the redundancy overhead, R_{agg} is:

$$R_{agg} = \frac{3+4+4+4+3+3}{3+3+1+2+3+2+4} = \frac{21}{18}$$

It can be seen that this now has an even lower redundancy overhead than either of the minimal cycles.

The examples shown above demonstrate that it is possible to have a lower redundancy overhead with an aggregated cycle, but this is not always the case and it is not, immediately, obvious which cycles are best to select. In practice the minimisation problem in (5.1), under its associated constraints, is NP-complete.

Theorem 1. *Determining the optimum set of p-cycles for an arbitrary network is NP-complete.*

Proof. Firstly there may be a, practically, uncountable number of possible cycles in an arbitrary graph. One of these possible cycles is a Hamiltonian cycle. Finding (or disproving the existence of) such a Hamiltonian is known to be NP-complete [61] completing the proof. □

Following Theorem 1, a heuristic algorithm is required to find a good set of cycles, $\hat{C}$ that lowers R compared to the minimum cycle set $C_{\min} \subset \mathcal{C}$. The algorithm here builds upon that by Drid et al. [62] with one important change that is highlighted below and in the results. The algorithm by Drid et al. [62] is described for wavelength division multiplexed networks, but can be adapted for electrically switched networks here as shown in Algorithm 1. The algorithm starts with a set of *minimum cycles* denoted $C_{\min}$ as was used in the previous chapter. A minimum cycle is the smallest cycle that includes a specific edge e. Finding a minimum cycle (assuming one exists) for each edge is very straightforward to determine, unlike the largest cycle, a Hamiltonian. To determine a minimum cycle to protect an edge is simply a matter of deleting the edge $e = s, t$ from the graph G and using a shortest path algorithm (such as Dijkstra's) to find a path that connects s, t. However, as there maybe multiple paths, this means that C_{min} is not unique, and as it is possible that multiple edges can be protected by a cycle there maybe multiple combinations of cycles.

5.2 Algorithmic description

We note that the algorithm by Drid et al. [62] may not select an optimum choice of cycles to start the algorithm. Consequently, we propose an improvement to the algorithm which greedily chooses a good choice of minimum cycles, $\hat{C}_{\min}$. First we will describe the algorithm by Drid [62] which is shown in Algorithm 1. The algorithm takes as input: a set of minimum cycles C (or rather $\hat{C}_{min}$ that we will later describe); the working traffic allocated on each edge, $W = \{w_e \in W\}$ on the graph $G(E, V)$; and, the allocation of edges to protection cycles, $A = \{(e, c) | e \in E, c \in C\}$.

The algorithm is recursive and modifies the input variables so copies are made in lines 1–3. Then the algorithm works on trying to increase the cycles one-by-one by, merging them aiming to achieve a lower redundancy overhead, R; if the merged cycle results in a lower R it accepts this as a better cycle to replace the two selected for testing. The cycles tested then have their working traffic that they are protecting set to zero (whether they were merged or not) and the loop (lines 4–14) continues until there is no traffic left (line 4). The setting of the traffic to zero is just one way of "counting" the edges that have been searched for lowering the redundancy overhead, it does not have any other significance in the algorithm.

The first part of the loop (line 5) selects a set of cycles that have the least working traffic assigned to them $C_{\min W}$. This is a good choice to start as a consideration for merging as it will add the least amount of traffic to a larger cycle (potentially not adding any additional overhead if the cycle already was covering that amount of traffic for other edges). As there could be more than one of these cycles, the algorithm selects the largest, $\hat{c}$, (line

```
OptCycles($C, W, G, A$)
 1  $\hat{W} = W$
 2  $\hat{A} = A$
 3  $\hat{C} = C$
 4  while $\hat{W} \neq 0$ do
 5      $C_{\min W} = \arg\min_{c \in \hat{C}} (\min(w_e | e \in c,\ w_e \neq 0))$
 6      $\hat{c} = \arg\max_{c \in C_{\min T}} (|c|)$
 7      $C' =$
        $\left\{ c' \in \hat{C} \mid e \in \hat{c} \wedge e' \in c',\ \hat{c} \neq c',\ R(\hat{W}, \hat{C}, G) > R(\hat{W}, M(\hat{C}, c', \hat{c}), G) \right\}$
 8      if $C' \neq \emptyset$ then
 9          $c' = \arg\max_{c \in C'} \left( R(\hat{W}, \hat{C}, G) - R(\hat{W}, M(\hat{C}, c', \hat{c}), G) \right)$
10          $\hat{C} = M(\hat{C}, c, \hat{c})$
11          $\hat{A} = M(\hat{A}, c', \hat{c})$
12          $w_e = 0,\ \forall e \in (e, M(c', \hat{c})) \in A,\ w_e \in \hat{W}$
13      else
14          $w_e = 0,\ \forall e \in (e, \hat{c}) \in \hat{A},\ w_e \in \hat{W}$
15  if $|\hat{C}| < |C|$ then
16      $\hat{C} =$ OptCycles($\hat{C}, W, G, \hat{A}$)
17  return $(\hat{C}, \hat{A}$
```

Algorithm 1: Algorithm to optimise cycles

6) as this has the potential to merge the most number of edges. Once this cycle has been selected then it is compared with every cycle $c' \in \hat{C}$ that shares common edges with $\hat{c}$, looking for a combination that has a lower R if the cycles are merged. Although this would be implemented as a loop in practice, it can be expressed as one line in set notation (line 7). In line 7 e' is the reverse edge of e and thus $e \in \hat{c} \wedge e' \in c'$ denotes that we are looking for cycles, c', that share an edge (and reverse edge) in their cycles; these are cycles that can be merged. In line 7 we introduce a generic merge function

M which denotes that the two cycles are merged, deleted from the cycle set and returned with the new merged cycle added. There may be more than one cycle c' that lowers R if it is combined with $\hat{c}$ and these are denoted C' (line 7). Consequently: lines 8-12 find the particular cycle that has the best reduction in R (line 9); then records this new merged cycle in the updated cycle set, $\hat{C}$, (line 10); and, updates the allocation of edges to this new merged cycle $\hat{A}$ (line 11, with an overloaded form of the generic merge function M). Then the working traffic for edges in this new cycle is set to zero (line 12). It should be noted that the function to merge $\hat{A} = M(\hat{A}, c', \hat{c})$ (and generically used earlier) will replace the common edge that was selected in both cycles (in its bidirectional sense) and thus the allocation will record this link as a straddling link in the new merged cycle. It is possible that multiple edges are shared in two cycles, but it is likely that only one can be replaced by a straddling link, thus here we assume that only one edge at a time is removed to form the merged cycle. For example, if two consecutive links were replace, then that would leave the middle node missing from the merged protection cycle.

As stated earlier the algorithm by Drid et al. [62], in Algorithm 1 does not consider the best choice of the starting minimum cycles, $C_{\min} \subset \mathcal{C}$. It is hypothesised that a good selection are those that start with the lowest redundancy overhead. Consequently, a greedy approach is taken as shown in Algorithm 2. This starts with the list of all candidate minimum cycles, $\mathcal{C}$ and computes the "best" choice for $C_{\min} \subset \mathcal{C}$ by computing the minimum redundancy overhead, R for all the candidate cycles that protect an edge, e (line 4). Once the best cycle has been chosen for an edge this is added to the

```
GreedyCyclesSel($\mathcal{C}, W, G(E,V)$)
1 $A = \emptyset$
2 for $e \in E$ do
3     $C' = \{c \in \mathcal{C} | e \in c\}$
4     $c = \arg\min_{c \in C'} (R(W,c,G))$
5     $A = A + (e,c)$
6 $\hat{C}_{\min} = \{c \in \mathcal{C} | c \in (e,c) \in A\}$
7 $(\hat{C}, \hat{A})$ =OptCycles($\hat{C}_{\min}, W, G, A$)
8 return $(\hat{C}, \hat{A})$
```

Algorithm 2: Algorithm to greedly pre-sort all the mimum cycles $\mathcal{C}$

allocation set A. Once the allocation set has been created then the candidate cycles, $C_{\min} \subset \mathcal{C}$ can be simply determined from A (line 6). Finally, this is the candidate set of cycles that are input to Algorithm 1 (line 7) so that the optimised set of cycles, $\hat{C}$, and edge allocations to cycles, $\hat{A}$ can be returned.

5.2.1 Algorithm complexity

3 complexityThe algorithm complexity will be considered in two parts: first Algorithm 2 which implements GreedyCycleSel, this then calls OptCycles which is shown in Algorithm 1 and will be considered secondly. GreedyCycleSel consists of a for loop for each edge $e \in E$ and in each iteration the dominating item is line 4 which calculates R with complexity $\mathcal{O}(|E|)$, consequently this loop is $\mathcal{O}(|E|^2)$, the other lines are simple updates to the data structures. It should be noted that GreedyCycleSel needs the minimum cycle set as an input. The simplist way to calculate the minimum cycle set is to perform Dijkstra's algorithm for each edge (removing the edge to find the shortest loop for each edge), thus the complexity

of this is running Dijkstra's algorithm [77] for each edge, i.e., complexity $\mathcal{O}(|E|^2 + |E||V|\log|V|)$. The last step in GreedyCycleSel is to call OptCycles. Excluding this last step we can thus give the complexity as dominated by calculating the minimum cycle set.

The complexity of OptCycles is harder to compute as it varies opon the while loop in line 4 and the recursion in line 16, which are dependent upon the exact scenario. However, as the combination of this loop and recursion are to convert each minimum cycle to a larger cycle the worst case would be to start with all minimum cycles and iterate one-by-one until there is a single (Hamiltonian) cycle. Given that the number of cycles is at most $\mathcal{O}(|E|)$ we can unroll the worst case of OptCycles as a loop of $\mathcal{O}(|E|)$. In each of these loops the dominating lines are calculating C_{min} which is $\mathcal{O}(|E|)$ and line 7 which calculates the redundancy overhead R which is also $\mathcal{O}(|E|)$. The other lines are are either updates to data structures or line 9 which can be computed while calculating line 7. Thus the worst case complexity of OptCycles is $\mathcal{O}(|E|^2)$. Now noting that this is comparable to (or dominated by) the complexity of GreedyCycleSel we have the final complexity as:

$$\mathcal{O}(|E|^2 + |E||V|\log|V|) \tag{5.4}$$

While this complexity is of a power two in the number of edges, it should be noted that the number of edges are reasonably small for example the largest network in the Internet Topology Zoo is the KDL network [68] with 899 edges. The algorithms described above would be implemented in the Toplogy Manager as this would assign the cycle IDs, as described in Section 4.6.2. Given that the cycles are only updated periodically, for example daily based upon the peak hour estimation from previous days, then this complexity is not highly significant compared to the general LID allocation task of the Topology Manager.

5.3 Results

To evaluate the algorithm, the redundancy as calculated according to Eq. 5.2 was computed for three different graphs from the Internet Topology Zoo [68], namely: Dfn, HibernianGlobal and Garr200112. These were chosen as they are representative of typical operator networks i.e. they are mesh networks rather than some of the networks in Internet Topology Zoo which are simple star networks. The properties for the networks are presented in Table 5.1. The graphs were tested 100 times, with a different traffic matrix each time that was generated between all pairs using a Log Normal distribution as is commonly seen in operator networks [78].

The results are shown in Figure 5.3, where the algorithms are denoted as:

Unopt is the assignment of minimal cycles, as used in Chapter 3

Ran is the algorithm as proposed by Drid et al.

Greedy is the algorithm presented in this chapter as Algorithm 2.

The results are presented as the ratio of the redundancy overhead R improvement between the Unopt case and either the Ran or Greedy algorithms. It can be seen that both Ran and our Greedy algorithm have a mean improvement of between 1.08 and 1.2 depending upon the network however, in all cases our Greedy algorithm outperformed the Ran algorithm and in some

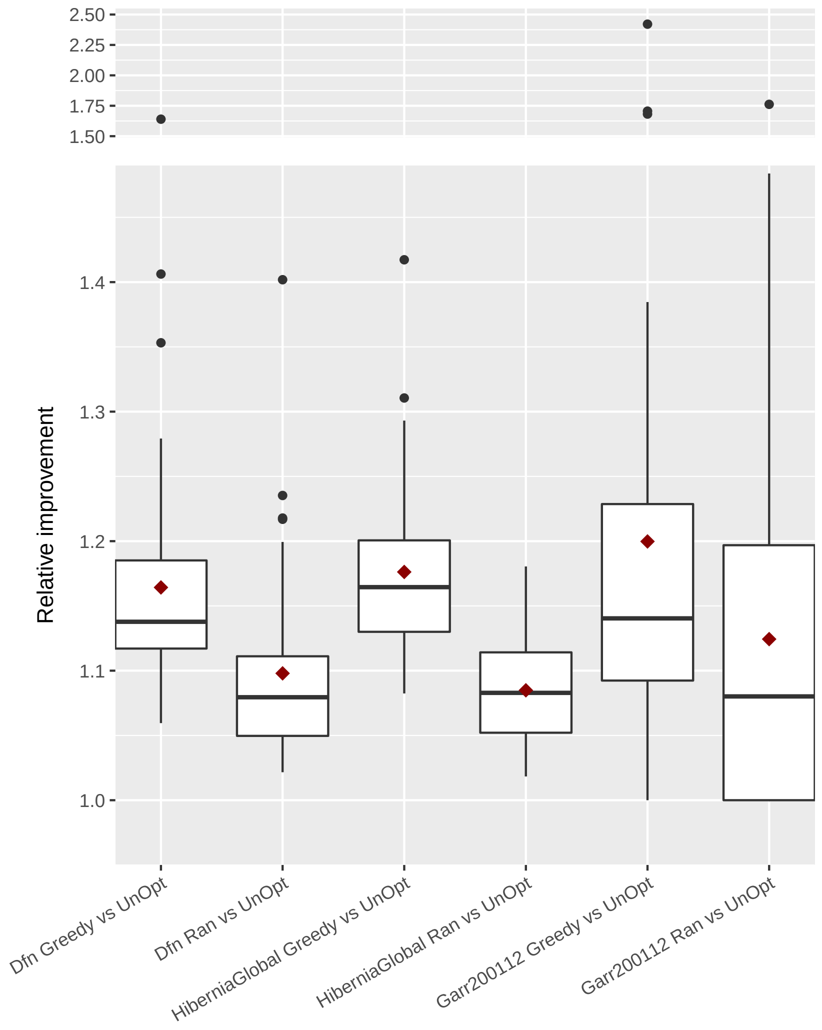


Figure 5.3: Comparison between algorithms showing the relative improvement in redundancy overhead R compared to the Unopt algorithm

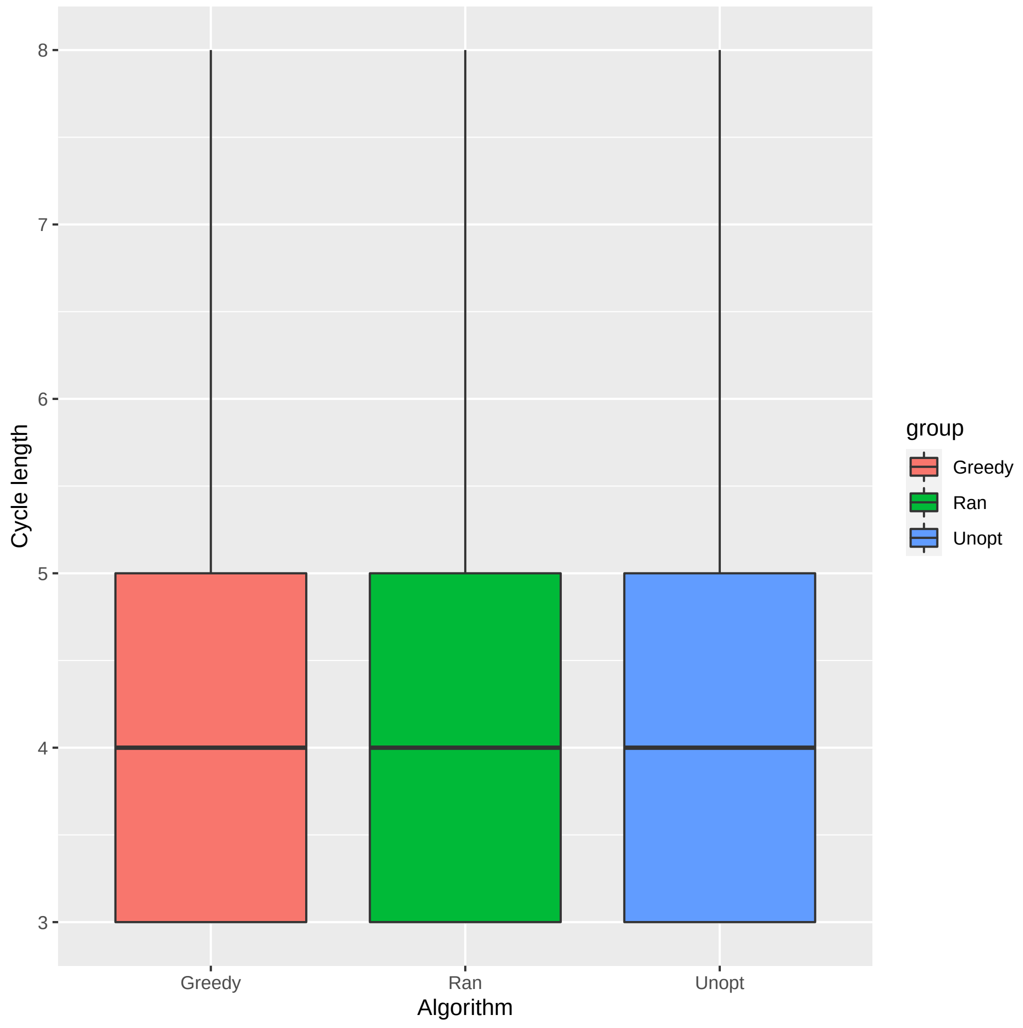


Figure 5.4: Box plot showing the distribution of cycle lengths is highly similar across all algorithms (for the Dfn network).

Network	No. Nodes	No. Edges	Mean Node Degree
Dfn	58	87	3.0
HiberniaGlobal	55	81	2.95
Garr2001112	24	26	2.17

Table 5.1: Basic properties for the chosen networks from Internet Topology Zoo

Algorithm	Mean Num. Cycles	Mean Cycle Length
Greedy	60.24	3.987
Ran	60.14	3.989
Unopt	62.00	3.968

Table 5.2: Analysis showing number of cycles and mean cycle length in finer granularity

cases achieved a substantial improvement of up to 2.4 times (compared to the best for Ran of 1.8).

The distribution of cycle lengths is shown in Figure 5.4 which shows that there is no clear difference between the algorithms at this level of view. More detailed analysis shows that the cycles only differ by a small amount. For example, for the Dfn network, in the Unopt case there are 62 cycles, but in the Ran algorithm there typically between 58 and 62 cycles, whereas in our Greedy algorithm there are between 57 and 62. The mean statistics, across the 100 experiments, are shown in Table 5.2. Thus, only a small difference in cycles accounts for the significant savings shown in the results in Figure 5.3. In practice this is because aggregating the cycles for the paths with significant overheads is enough to reduce the redundancy overhead, and once these cycles have been aggregated there is no further benefit in aggregating.

5.4 Summary

In Chapter 4 the set of minimal cycles were assumed without considering the optimal cycles. Now this chapter considered optimising the cycles to reduce the overhead when using p-cycles. The approach extended the work by Drid et al. [62] by aggregating minimal cycles into larger cycles where this reduced the overhead. The work found that Drid et al. did not consider the choice of minimal cycles to use in their algorithm. Consequently, here a greedy algorithm was added which pre-selected the best minimal cycles, based upon initial lowest initial overhead. Then these cycles were used in the process of cycle aggregation. The analysis, using synthetic traffic but realistic networks, showed consistently better performance compared to the algorithm by Drid et al. and typically a factor reduction in the redundancy overhead of between 1.17 and 1.2 compared to using minimal cycles.

Chapter 6

Conclusion

The forth chapter introduced the p-cycle switching in combination with the PURSUIT ICN architecture. It showed that it can have significantly less switching state and traffic overhead than MPLS multicast switching. This was demonstrated by simulating the network scenario in a number of realistic networks. The work has shown that there can be a provable bound on the additional switching overhead in terms of switching state and packet header size represented by the CID. In the case of a planar network it was proven that this can be achieved using only four different identifiers, i.e., the overhead in the frame for the CID is only two bits. In the case of non-planar networks, the highest number of unique CIDs was proven to be at most the largest node degree in the network. In the case of practical networks the largest number of unique CIDs was found to be 14 in a common database of real networks provided by the Internet Toplogy Zoo [68].

The second contribution was presented in Chapter 5 which optimised the cycles to reduce the overhead caused by capacity planned for possible protection switching. This adapted the cycles that were used in Chapter 3, which assumed minimal cycles without considering the optimal cycles. The approach extended the work by Drid et al. [62] by aggregating minimal cycles into larger cycles where which reduced the protection capacity overhead. The work found that Drid et al. did not consider the choice of minimal cycles to use in their algorithm. Consequently, a greedy algorithm was added which pre-selected the best minimal cycles, based upon the initial lowest initial overhead. Then these cycles were used in the process of cycle aggregation. The analysis, used synthetic traffic but realistic networks and showed consistently better performance compared to the algorithm by Drid et al. and typically a factor reduction in the redundancy overhead of between 1.17 and 1.2 compared to using minimal cycles.

In both Chapters 4 and 5 the practicalities of implementing the system were discussed. Firstly in Chapter 4, it was shown that the mechanism can be simply implemented in existing SDN switches by simply overloading the MPLS header to use as the CIDs. This is compatible with earlier work that showed that SDN switches can be used to implement the Bloom filter switching [21]. As the mechanism only requires the CIDs to be allocated when the topology changes it was clear that the overhead of the protection mechanism can be simply implemented in the ICN Topology Manager as a low-level background task when compared to the online path determination that it has to perform. In Chapter 5 it was shown that the complexity of the cycle optimisation was at most a square relationship of the size of the network. Again as this optimisation is only performed when there is a topology change in the network it is of relatively low cost and can again be implemented in the Topology Manager.

6.1 Future directions

The work presented in this thesis also reveals that multiple failures and node failures may be further considered by extending the theories described here to include multiple failures and node failures and other aspects as well. Additionally, some simulation results on real time impairment monitoring and analysis of switching times would be a useful extension, but would require a new experimental framework to be developed which simulates actual outages.

6.2 Final conclusion

This work is the first, and only, contribution on protection switching in ICN architectures, and as far as the author is aware is the only stateless multicast protection solution. The work has shown that the state and traffic costs are significantly less for multicast switching in the chosen ICN architecture compared to MPLS. Furthermore, the thesis has shown that the optimisation of p-cycles can be improved to give a highly efficient multicast protection mechanism for next generation networks.

Bibliography

[1] L. Ceci, “Hours of video uploaded to youtube every minute as of february 2020.” https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/ [accessed 12th January 2022], 2022.

[2] I. Sodagar, “The mpeg-dash standard for multimedia streaming over the internet,” *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[3] D. Trossen, A. Rahman, C. Wang, and T. Eckert, “Applicability of BIER multicast overlay for adaptive streaming services,” 2021. IETF Internet Draft draft-ietf-bier-multicast-http-response-06 [available online https://datatracker.ietf.org/doc/html/draft-ietf-bier-multicast-http-response-06].

[4] D. Trossen, M. J. Reed, M. Georgiades, N. Fotiou, and G. Xylomenos, “IP Over ICN - The Better IP? An Unusual Take on Information-Centric Networking,” in *Networks and Communications (EuCNC), 2015 European Conference on*, 2015.

[5] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing Information Networking Further: From PSIRP to PURSUIT," in *Broadband Communications, Networks, and Systems* (I. Tomkos, C. J. Bouras, G. Ellinas, P. Demestichas, and P. Sinha, eds.), (Berlin, Heidelberg), Springer Berlin Heidelberg, 2012.

[6] A. Kodian and W. D. Grover, "Failure-independent path-protecting p-cycles: Efficient and simple fully preconnected optical-path protection," vol. 23, no. 10, pp. 3241–3259, 2005.

[7] A. Zahemszky and S. Arianfar, "Fast reroute for stateless multicast," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pp. 1–6, 2009.

[8] D. Clark, *Designing an Internet.* MIT Press, 2018.

[9] S. Pasqualini, A. Iselt, A. Kirstädter, and A. Frot, "Mpls protection switching versus ospf rerouting," in *Quality of Service in the Emerging Networking Panorama* (J. Solé-Pareta, M. Smirnov, P. Van Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, and R. J. Gibbens, eds.), (Berlin, Heidelberg), pp. 174–183, Springer Berlin Heidelberg, 2004.

[10] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the aggregatability of router forwarding tables," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, 2010.

[11] M. Mitzenmacher, *Bloom Filters*, pp. 252–255. Boston, MA: Springer US, 2009.

[12] Y. Chen, A. Kumar, and J. J. Xu, "A new design of bloom filter for packet inspection speedup," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, pp. 1–5, 2007.

[13] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Trans. Comput. Syst.*, vol. 8, p. 85–110, may 1990.

[14] H. Holbrook and S. Systems, "Source-Specific Multicast for IP." RFC 4607, Aug. 2006.

[15] B. Zhang and H. Mouftah, "Forwarding state scalability for multicast provisioning in ip networks," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 46–51, 2003.

[16] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. S. Thyagarajan, "Internet group management protocol, version 3." RFC 3376, 2002.

[17] B. Fenner, M. J. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. J. Zhang, and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)." RFC 7761, Mar. 2016.

[18] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic engineering with mpls in the internet," *IEEE Network*, vol. 14, no. 2, pp. 28–33, 2000.

[19] D. Tappan, Y. Rekhter, A. Conta, G. Fedorkow, E. C. Rosen, D. Farinacci, and T. Li, "MPLS Label Stack Encoding." RFC 3032, Jan. 2001.

[20] A. Zahemszky, P. Jokela, M. Sarela, S. Ruponen, J. Kempf, and P. Nikander, "MPSS: Multiprotocol Stateless Switching," in *2010 IN-*

FOCOM IEEE Conference on Computer Communications Workshops, Mar 2010.

[21] M. J. Reed, M. Al-Naday, N. Thomos, D. Trossen, G. Petropoulos, and S. Spirou, "Stateless multicast switching in software defined networks," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2016.

[22] S. A. Ladhe and V. T. Raisinghani, "A resource efficient common protection path approach for mpls-based recovery," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–9, 2013.

[23] F. Blouin, A. Sack, and W. Grover, "Benefits of p-cycles in a mixed protection and restoration approach," in *Fourth International Workshop on Design of Reliable Communication Networks, 2003. (DRCN 2003). Proceedings.*, pp. 203–210, 2003.

[24] H. Alazemi, S. Sebbah, and M. Nurujjaman, "Fast and efficient network protection method using path pre-cross-connected trails," *Journal of Optical Communications and Networking*, vol. 5, no. 12, pp. 1343–1352, 2013.

[25] M. S. Kiaei, C. Assi, and B. Jaumard, "A survey on the p-cycle protection method," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 53–70, 2009.

[26] O. Lemeshko and O. Yeremenko, "Linear optimization model of mpls traffic engineering fast reroute for link, node, and bandwidth protec-

tion," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pp. 1009–1013, 2018.

[27] D. Tipper, "Resilient network design: challenges and future directions," *Telecommunication Systems*, vol. 56, pp. 5–16, May 2014.

[28] J. Moy, "OSPF Version 2." RFC 2328, Apr. 1998.

[29] L. Aguirre-Torres and G. Rosenfeld, "A comparison of layer-2 protection and restoration mechanisms for data-aware transport networks," in *Optical Fiber Communication Conference and Exposition and The National Fiber Optic Engineers Conference*, p. NThN2, Optica Publishing Group, 2005.

[30] G. Maier, A. Pattavina, S. De Patre, and M. Martinelli, "Optical network survivability: Protection techniques in the wdm layer," *Photonic Network Communications*, vol. 4, pp. 251–269, Jul 2002.

[31] F. Blouin, A. Sack, and W. Grover, "Benefits of p-cycles in a mixed protection and restoration approach," in *Fourth International Workshop on Design of Reliable Communication Networks, 2003. (DRCN 2003). Proceedings.*, pp. 203–210, 2003.

[32] D. Stamatelakis and W. Grover, "Ip layer restoration and network planning based on virtual protection cycles," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1938–1949, 2000.

[33] D. Stamatelakis and W. D. Grover, "Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles ("p-

cycles"")," *IEEE Transactions on Communications*, vol. 48, no. 8, pp. 1262–1265, 2000.

[34] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 186–204, 2010.

[35] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), p. 1–12, Association for Computing Machinery, 2009.

[36] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 1024–1049, Second 2014.

[37] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, p. 66–73, jul 2014.

[38] X. Fu, D. Kutscher, S. Misra, and R. Li, "Information-centric networking security," *IEEE Communications Magazine*, vol. 56, pp. 60–61, 11 2018.

[39] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: A survey," *Computer Science Review*, vol. 19, pp. 15–55, 2016.

[40] M. Al-Khalidi, N. Thomos, M. J. Reed, M. F. AL-Naday, and D. Trossen, "Anchor free ip mobility," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 56–69, 2019.

[41] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, "Information-centric networking for the internet of things: challenges and opportunities," *IEEE Network*, vol. 30, pp. 92–100, 2016.

[42] B. Nour, H. Khelifi, H. Moungla, R. Hussain, and N. Guizani, "A distributed cache placement scheme for large-scale information-centric networking," *IEEE Network*, vol. 34, pp. 126–132, 2020.

[43] *Service-Oriented Architecture*, pp. 89–113. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[44] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, (New York, NY, USA), p. 271–282, Association for Computing Machinery, 2014.

[45] M. Pathan, R. Buyya, and A. Vakali, *Content Delivery Networks: State of the Art, Insights, and Imperatives*, pp. 3–32. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[46] E. B. Nascimento, D. D. de Macedo, and E. D. Moreno, "Resources optimization in icns through distributed cache using software defined

networking – sdn," in *2018 Symposium on High Performance Computing Systems (WSCAD)*, pp. 268–268, 2018.

[47] S. Kul and A. Sayar, "A survey of publish/subscribe middleware systems for microservice communication," in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 781–785, 2021.

[48] G. Parisis, B. Tagger, D. Trossen, D. Syrivelis, P. Flegkas, L. Tassiulas, C. Stais, C. Tsilopoulos, and G. Xylomenos, "Demonstrating an information-centric network in an international testbed," in *Testbeds and Research Infrastructure. Development of Networks and Communities* (T. Korakis, M. Zink, and M. Ott, eds.), (Berlin, Heidelberg), pp. 400–402, Springer Berlin Heidelberg, 2012.

[49] G. Xylomenos, Y. Thomas, X. Vasilakos, M. Georgiades, A. Phinikarides, I. Doumanis, S. Porter, D. Trossen, S. Robitzsch, M. J. Reed, M. Al-Naday, G. Petropoulos, K. Katsaros, M.-E. Xezonaki, and J. Riihijarvi, "Ip over icn goes live," in *2018 European Conference on Networks and Communications (EuCNC)*, pp. 319–323, 2018.

[50] S. Tarkoma, M. Ain, and K. Visala, *The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture*, pp. 102–111. Netherlands: IOS PRESS, 2009.

[51] Z. Ahmad, Z. ul Abidin Jaffri, and I. Ali, "A survey on publish-subscribe internet routing paradigm," *International Journal of Informatics and Communication Technology*, vol. 2, pp. 144–154, 2013.

[52] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[53] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard, "Networking named content," *Communications of the ACM*, vol. 55, pp. 117 – 124, 2012.

[54] A. Shinde and S. M. Chaware, "Content centric networks (ccn): A survey," in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*, pp. 595–598, 2018.

[55] X. Qiao, H. Wang, W. Tan, A. V. Vasilakos, J. Chen, and M. B. Blake, "A survey of applications research on content-centric networking," *China Communications*, vol. 16, no. 9, pp. 122–140, 2019.

[56] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[57] K. Kogan, S. I. Nikolenko, O. Rottenstreich, W. Culhane, and P. Eugster, "Exploiting order independence for scalable and expressive packet classification," *IEEE/ACM Trans. Netw.*, vol. 24, p. 1251–1264, apr 2016.

[58] T. A. Wibowo, N. R. Syambas, *et al.*, "Named data network (ndn) scalability problem," in *2019 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pp. 112–118, IEEE, 2019.

[59] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration," in *ICC '98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No.98CH36220)*, vol. 1, pp. 537–543 vol.1, 1998.

[60] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: Understanding the causes and impact of network failures," *Computer Communication Review*, vol. 40, no. 4, pp. 315–326, 2010.

[61] G. Narasimhan, "A note on the Hamiltonian circuit problem on directed path graphs," *Information Processing Letters*, vol. 32, no. 4, pp. 167–170, 1989.

[62] H. Drid, B. Cousin, and M. Molnar, "Heuristic Solution to Protect Communications in WDM Networks using P-cycles," in *Workshop on Traffic Engineering, Protection and Restoration for Future Generation Internet*, (Oslo, Norway), 2007.

[63] ONF, "OpenFlow switch specification version 1.2," tech. rep., Open Networking Foundation, December 2011.

[64] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, Algorithms and Applications.* Springer Publishing Company, Incorporated, 2nd ed., 2008.

[65] R. L. Brooks, “On colouring the nodes of a network,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 37, no. 2, p. 194–197, 1941.

[66] R. Bowden, H. X. Nguyen, N. Falkner, S. Knight, and M. Roughan, “Planarity of data networks,” in *Proceedings of the 23rd International Teletraffic Congress*, ITC ’11, p. 254–261, International Teletraffic Congress, 2011.

[67] K. Appel and W. Haken, “Every planar map is four colorable. part I: Discharging,” *Illinois J. Math.*, vol. 21, pp. 429–490, 09 1977.

[68] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The internet topology zoo,” *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 1765–1775, October 2011.

[69] R. M. Karp, *Reducibility among Combinatorial Problems*, pp. 85–103. Boston, MA: Springer US, 1972.

[70] D. Brélaz, “New methods to color the vertices of a graph,” *Commun. ACM*, vol. 22, p. 251–256, Apr. 1979.

[71] M. Castro, M. B. Jones, A. . Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, “An evaluation of scalable application-level multicast built using peer-to-peer overlays,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, pp. 1510–1520 vol.2, March 2003.

[72] E. Rosen and R. Aggarwal (eds), "Multicast in MPLS/BGP IP VPNs." IETF RFC 6513, February 2012.

[73] I. Martinez-Yelmo, D. Larrabeiti, I. Soto, and P. Pacyna, "Multicast traffic aggregation in MPLS-based VPN networks," *IEEE Communications Magazine*, vol. 45, no. October, pp. 78–85, 2007.

[74] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[75] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *Vol, Siam J Comput*, vol. 4, no. 1, pp. 77–84, 1975.

[76] D. A. Mello, H. Waldman, and G. S. Quitério, "Interval availability estimation for protected connections in optical networks," *Computer Networks*, vol. 55, no. 1, pp. 193–204, 2011.

[77] M. Barbehenn, "A note on the complexity of dijkstra's algorithm for graphs with weighted vertices," *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 263–, 1998.

[78] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating ip traffic matrices: Initial recommendations," *SIGCOMM Comput. Commun. Rev.*, vol. 35, p. 19–32, jul 2005.