Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

Routing Algorithms for Time Sensitive Networks

Subarna Singh

Course of Study:	Information Technology/InfoTECH
Examiner:	Prof. Dr. Kurt Rothermel
Supervisor:	M.Sc. Naresh Nayak
Commenced:	25. April 2017
Completed:	26. October 2017
CR-Classification:	C.2.2

Abstract

IEEE 802.1Qbv standard is an enhancement introduced by Time Sensitive Network Task Group to provide real time communication in a converged Ethernet network capable of transmitting both time critical traffic and best effort traffic. The standard includes a gating mechanism which controls the access to the transmission medium at the egress port of a switch, allowing transmission of frames from the specified queue. The gating events are time triggered and are programmed using a transmission schedule.

A transmission schedule is a cyclic schedule that can be generated using various independent scheduling methods. However, the first step to each method involves routing the traffic across the network, followed by computation of gate schedules along the route. Due to lack of a standard, assessment of the quality of a schedule is difficult. Therefore independent metrics are used to measure the quality of schedule generated by different scheduling methods. However, the aim of each scheduling method is to transmit maximum amount of data traffic in a single cycle of a schedule.

This thesis proposes that routing impact the quality of a transmission schedule irrespective of the scheduling method used. To begin evaluations showing the impact of routing algorithms on the data load distribution across the network are present. Followed by evaluations of the impact of data load distribution on the quality of a transmission schedule. Consequently establishing a direct relation between routing of the data traffic and the quality of a schedule. In response a generic routing algorithm aimed at optimizing the data load distribution is presented. The algorithm is based on heuristic algorithm Tabu Search to make it scalable in nature, allowing its use across different scenarios. We evaluate the algorithms with respect to data load distribution, impact on the quality of a schedule and scalability.

Acknowledgement

It gives me great pleasure in acknowledging the contributions of all those without whom this thesis would not be possible. First and foremost, I wish to thank Prof. Dr. Kurt Rothermel for giving me an opportunity to do my thesis in the Department of Distributed Systems.

This work would not have been possible without the guidance of my supervisor Naresh Nayak, he has taught me more than I could ever give him credit for here. He has shown me, by his example what a good researcher should be like. He has patiently corrected my writing and supported my research.

I would like to thank my parents and my brother, for the constant support and love, which gives me the strength to pursue my goals. Finally I would like to thank my fiancé, Lt Cdr Anurag Sharma for motivating and supporting me.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

There is a growing requirement for a network capable of providing real time communication in areas such as Industrial Internet of Things (IIoT), automotive networking, power/utility grid networks, telecommunication networks etc. The applications in these fields require strict adherence to deadlines, this includes transmission and delivery of data. A real time communication ensures the transmitted data is delivered at its destination before the specified deadline. Using Industrial IoT (IIoT) as an example let us try to understand the significance of such a network. Industrial IoT (IIoT) provides automation of physical processes in a manufacturing industry with the added advantages of saving cost, operational efficiency and improved safety standards. An essential component of this is Internet of Things (IoT), which is a concept enabling physical devices to communicate with each other, thus allowing collection and exchange of data. The physical devices in IoT are a general class of Cyber Physical Systems (CPS), embedded with software, electronics, sensors, actuators and network connectivity. The sensor in a CPS monitors and captures the state of the physical device, and sends the collected data to a controller. The controller is used as an interface to send commands to the actuator, which affects the state of the physical device. The Cyber Physical Systems (CPS) may be distributed across the plant and since they are controlled by the controller, they need to be connected via a network, hence distributed real time control system form an integral part of Industrial IoT (IIoT) [12]. A distributed real time control system requires

- a converged network, capable of classifying the data traffic into classes and providing services based on the requirement of each class

and the following two essential services:

- a strict real time communication, to enable applications with hard deadlines such as motion control.
- a high bandwidth, to support a large number of sensors, and a large amount of data which is required to drive Industrial IoT applications like predictive maintenance and data analysis.

We shall discuss the available network technologies and their viability to offer these services.

Ethernet (IEEE 802.3) is one of the most popular and widely adapted network protocol. Its reputation for providing high performance in Local Area Networks (LAN) has made it a ubiquitous standard. It provides operational simplicity, interoperability with other technologies, and cost effectiveness due to its widespread adoption. It is also capable of providing flexible topology, thus allowing future expansions. Ethernet would be the preferable choice of network media while designing a network. However, Ethernet is non-deterministic in nature, it is unable to predict when a node will be allowed to access the network medium for transmission of data. It also does not provide in-built mechanisms to guarantee bounded latency while transmitting data through the network. These drawbacks limit its viability to provide a strict real time communication in an environment such as Industrial IoT (IIoT) [4].

Industrial networks were originally developed using point to point links between devices in a manufacturing plant and their controller. Fieldbus technology was developed with the intent to replace these networks. It allowed connection of multiple devices and their controller over a single digital link by transmitting information in serial order and multiplexed in time. Ethernet based Fieldbus technologies like Profibus[18], EtherCAT[15], and SERCOS III[11] are widely adopted in the industry. They offer real time, deterministic and low latency communication, but a network built using either technology is incompatible with other propriety technologies. This has led to a high cost of investment. Using proprietary technology such as these requires investment in devices compatible with the technology and also faces the risk of loss of investment if a better technology were to be adopted in future.

The shortcomings of these technologies were addressed with the advent of Time Sensitive Network (TSN). Time Sensitive Network(TSN) is part of IEEE 802.1 group, a set of specification aimed at providing deterministic services through IEEE 802 networks, which includes guaranteed packet transport with bounded low latency, low packet delay variation, and low packet loss. It is a layer 2 technology that can be used in any environment to carry the payload of any industrial application. Since it can be used over standard Ethernet it offers the advantage of utilizing high bandwidth in automation networks. Thus, successfully fulfilling both the service requirements of a distributed real time control system.

To understand how a Time Sensitive Network (TSN) achieves the above-mentioned goals we need to understand the underlying architecture and the protocols used. The network components of a Time Sensitive Network (TSN) architecture and their functionality are as follows[17]:

- TSN Flow: real time communication between end devices, which can be uniquely identified by network devices.

- End Device: end nodes (source and destination node) for a TSN Flow. They run applications demanding deterministic real time communication.
- Bridges: Ethernet switches capable of transmitting and receiving Ethernet frames of a TSN flow. It controls the transmission of a TSN frame depending on the specification used. For example, a IEEE 802.1Qbv compliant Ethernet switch controls transmission via gate opening event. Each port in the switch has multiple queues, these have a gating mechanism which determines if the queue is allowed to transmit into the port. The gate opening events are programmed in a programmable gate driver present on the switch. The events are programmed based on a schedule.
- Central Network Controller (CNC): is a proxy for the Network (the TSN Bridges and their interconnections) and the control applications that require deterministic communication. It defines the schedule on which all TSN frames are transmitted.
- Centralized User Configuration (CUC): is an application that communicates with the CNC and the end devices. It represents the control applications and the end devices and makes requests to the CNC for deterministic communication with specific requirements.

These network components require a common set of protocols to deliver real time communication. These protocols include time synchronization, selection of communication paths, path reservation, and fault tolerance, and scheduling and traffic shaping. Following are a few standards published under the umbrella of Time Sensitive Network (TSN) task group that can be categorized into the set of protocols mentioned above :

- IEEE Std 802.1AS Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks
- IEEE Std 802.1Qca Path Control and Reservation.
- IEEE Std 802.1Qbv Enhancements for Scheduled Traffic

IEEE Std 802.1AS Timing and Synchronization is a set of protocols and procedures for transportation of synchronized time, selection of the timing source (best master) and indication of timing impairment [1].

IEEE Std 802.1Qca Path Control and Reservation is a set of protocols for explicit path control, bandwidth and stream reservation, redundancy (protection or restoration) for data flows (set of frames to be transmitted from node to another). It also includes a new control protocol which will provide explicit forwarding path control, enabling the use of non-shortest paths [3].

IEEE Std 802.1Qbv Enhancements for Scheduled Traffic specifies time-aware queue-draining procedures, and provide extensions to existing protocols that enable bridges and end stations to schedule the transmission of frames based on timing derived from IEEE Std 802.1AS. It also supports Virtual Local Area Network (VLAN) tag encoded priority values, allowing support of scheduled traffic, credit-based shaper traffic and other bridged traffic over Local Area Networks (LANs) [2].

The discussed specifications along with others in Time Sensitive Network (TSN) have been successful in achieving deterministic real time communication on 802 networks. However, there exists no standard for generating schedules for transmission of TSN frames. Before dwelling into the methods for generating these schedules we need to understand its definition. A transmission schedule is a set of gate opening and closing events on individual switches. These events allow control over the access of the transmission medium. Only if the gate of a queue is open can the data packets buffered in the queue transmit. However, an IEEE 802.1Qbv compliant switch does not allow preemption of transmitting frame. If a low priority frame is being transmitted, in spite of an open gate the high priority frame will be unable to access the medium. This introduces delay in transmission. To tackle this issue IEEE specification 802.1Qbv introduced guard bands. They are of the size of an Ethernet frame and are placed before and after gate events (open and close) of a scheduled traffic. A guard can be computed explicitly or implicitly. An explicit computation means the guard bands are computed by the Central Network Controller during schedule generation and implicit computation means they are introduced by the switches.

On going research has resulted in several independent methods to generate transmission schedules, however in this thesis, we will be focusing on:

- No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN) developed by Duerr et al. at University of Stuttgart [10]. It solves the problem of transmission schedule generation using No-wait Job Shop Scheduling Problem.
- Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks developed by Craciunas et al. at TTTech Computertechnik AG [6]. It uses Satisfiability Modulo Theories (SMT) to generate transmission schedules.

Both methods successfully generate transmission schedules, given a network and set of TSN flows to be transmitted, but there is no standard metric to assess the quality of a transmission schedule. The decision to use a scheduling method lies purely on preferential basis.

In the paper No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN) [10], researchers argue that guard bands are a wastage of bandwidth for best effort traffic. Minimizing the number of gate events will reduce the number of guard bands, increasing the amount of best effort traffic transmission and also increasing the number of scheduled traffic frames transmitted in a single gate opening event. Therefore a transmission schedule is of good quality if it has minimal duration. A minimal duration implies fewer gate opening events. They have introduced a metric called Flowspan, which indicates the finishing time of the last flow transmitted through the network. Flowspan indicates the minimum duration for a schedule. This metric, however, is applicable only to the NW-PSP method of schedule generation. The method introduced in Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks offers no such metric or parameter to evaluate the quality of a transmission schedule generated by it.

In the paper Routing Algorithms for IEEE 802.1Qbv Networks [9] researchers at the University of Stuttgart established a direct relation between the routing of time triggered traffic and quality of schedule (flowspan). The routes of time triggered traffic are an input to the scheduling method. Based on these routes the switches on which gate opening events are to be scheduled are determined. The standard routing algorithm used is Shortest Path First (SPF), which routes the data traffic over the shortest available paths, concentrating the data load over selective few links in the network. This leads to contention between traffic on switch ports. Higher contention implies longer time is required to transmit the time triggered traffic. The researchers have introduced a metric called Maximum Scheduled Traffic Load (MSTL) [9] to measure this contention. It is defined as the maximum amount of scheduled (deterministic real time) traffic transmitted through a network. They have established a direct relationship between the MSTL and Flowspan, presenting proofs that decrease in the value of MSTL results in decrease in Flowspan of the transmission schedule. In this thesis, we will establish the validity of the metric MSTL on the other transmission schedule generating method.

They have also established that the value of MSTL is influenced by the routing algorithm used. For instance routing all scheduled flows over a single link causes a bottleneck and increases the MSTL value, in comparison to routing algorithm where the scheduled flows are distributed over the entire network. Hence they introduced two Integer Linear Programming (ILP) based algorithms for routing TSN flows [9], aimed explicitly at minimizing the MSTL of the network. The first algorithm is MSTL based routing, it optimizes the routes for time triggered traffic based only on the MSTL of the network. The second algorithm is MSTL + Hops based routing, it generates the routes of TSN flows while optimizing the number of hops in each route and the resulting MSTL of the network. The result produced reduced Flowspan for MSTL based routing, and MSTL + Hops based routing by up to 30% and 60 % respectively in comparison to shortest path routing. Thus establishing the direct relation between MSTL and routing of the TSN flows.

However, the runtime of ILP based algorithms is very high. The scalability of the algorithms was tested with respect to the number of flows and the size of the topology. In each case the increase in the runtime of MSTL based routing was gradual, but the runtime of MSTL + Hops based routing increased steeply. This is due to the complexity of combinatorial optimization problem which is exponential in nature. One of the methods that can be employed to tackle this issue is heuristics. Heuristics are approximate techniques to solve complex problems like optimization problems. They are developed to have low time complexity. Some examples of heuristic and meta heuristic algorithms are tabu search, simulated annealing, ant colony optimization etc. The aim of this thesis is to develop routing algorithms using heuristic and meta heuristic algorithms.

Based on our understanding of the problems in Time Sensitive Network (TSN), we have outlined the goals of this thesis as follows:

- To benchmark Maximum Scheduled Traffic Load (MSTL) as a metric that be used by any scheduling method to quantify the maximum traffic that is scheduled on a link in the network. Consequently allowing assessment of the data load distribution across the network.
- To identify individual metrics that can be used to evaluate the quality of a schedule generated by scheduling methods mentioned earlier. To establish a direct relation between MSTL and the quality of a schedule independent of the scheduling method used.
- To develop a generic routing algorithm based on heuristic algorithm Tabu search aimed at optimizing the MSTL value, while being scalable in nature.

Thesis Organization

The remaining part of the thesis is organized into the following six chapters :

Chapter 2 presents the main concepts which form the basis of the thesis. It begins with the principles of IEEE 802.1Qbv standard, followed by a brief discussion on the scheduling methods used in this thesis. An overview of Software Defined Network (SDN), IEEE 802.1Qca standard and heuristic algorithm Tabu Search is presented.

Chapter 3 provides the formal specification of the system model used and the problem statements of the thesis.

Chapter 4 discusses the impact of routing algorithms on the scheduling methods with respect to the quality of the schedule generated by them.

Chapter 5 presents the routing algorithms that have been realized in this thesis. It includes the details on the implementation of the algorithms and its impact on MSTL.

Chapter 6 presents the result of all the evaluations performed. Chapter 7 concludes the thesis with summary and proposes the possible future work .

2 Background

The objective of this chapter is to introduce the core concepts which form the basis of this thesis. We shall discuss the basic principles of IEEE 802.1Qbv, transmission scheduling methods, Software Defined Network (SDN), IEEE 802.1Qca and Heuristic algorithms.

2.1 IEEE 802.1Qbv

Some applications have hard deadlines for data delivery for example industrial control applications, where data is used to operate critical operations of the plant or machinery involved. Delay in delivery of this type of data may result in instability, inaccuracy or failure of the operation. Such applications require a network capable of providing predictable data delivery in terms of time, and determine the overall latency and jitter experienced during the propagation of the data through the network. Earlier, transmission of time critical traffic was achieved by dedicated networks, but the bandwidth utilization by time critical traffic was low and the cost of installing and maintaining the network was significantly high. Hence there was a need for a converged network capable of supporting different classes of traffic on the same physical network, while providing different quality of services (QoS) and meeting the strict requirements of a time critical traffic.

The IEEE 802.1 Time-Sensitive Networking Task Group (TSN-TG) is a major standard body in networking, working on a set of specification to provide deterministic services through IEEE 802 networks. These specifications will allow transmission of data of different traffic classes (time-sensitive and non-time-sensitive) over IEEE 802 compliant network, thus allowing maximum utilization of the bandwidth. Originally, they were known as Audio Video Bridging (AVB) task group and were tasked to provide time synchronized low latency streaming. This ongoing development drew the attention of industrial and automotive communities and soon the Audio Video Bridging (AVB) task group's focus shifted towards industrial and automotive applications. The goal of Time-Sensitive Networking (TSN) is to provide time synchronized, low-latency transmission of

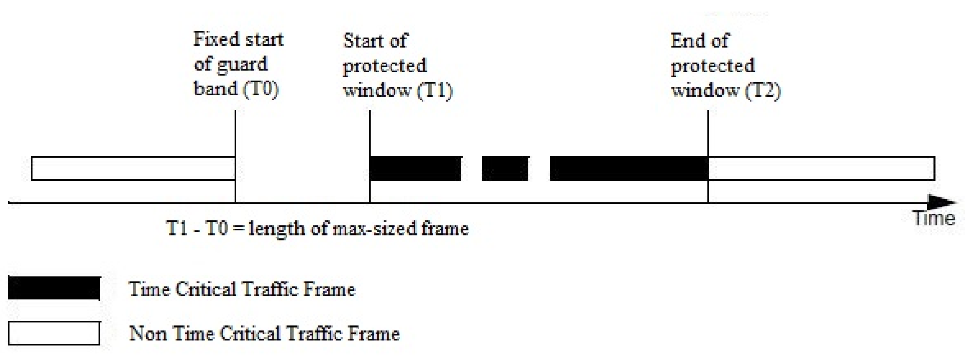

Fixed start of guard band (T0)
Start of protected window (T1)
End of protected window (T2)
T1 - T0 = length of max-sized frame
Time
Time Critical Traffic Frame
Non Time Critical Traffic Frame

Figure 2.1: Guardband between Time Critical Traffic Frames and Non Time Critical Traffic Frames in Time Senstive Network (TSN) [14]

data over Ethernet networks. This goal is achieved via protocols developed under Time-Sensitive Networking (TSN) umbrella which can be categorized as time synchronization protocol, stream reservation protocol, scheduler and shaper protocols etc.

Time Sensitive Networking (TSN) employs the technique of prioritization to attain deterministic delivery of time critical traffic. Prioritization is achieved using traffic shaping in combination with differentiated services. Traffic shaping is the practice of delaying transmission of frames of lower priority traffic to allow high priority traffic to be transmitted first. For example, applications like email which use best effort service are not affected by delay in delivery of frames, but time critical applications have hard deadlines, delay in frame delivery can result in system failure, damages or even loss of life. Therefore, in such systems time critical traffic frames are given priority over best effort traffic frames and are transmitted first. Differentiated services provides mechanism to classify traffic into aggregate traffic classes. It provides relative quality of service (QoS) guarantees, which are based on hop by hop prioritization of traffic classes and does not provide absolute performance guarantees. The performance of differentiated services depends on the overall traffic in the network.

However prioritization has its limits in providing deterministic delivery of time critical traffic. To guarantee deterministic delivery in terms of time, access to transmission medium at specific instant in time is critical. This access can be hindered by a transmitting frame belonging to a lower priority traffic, a higher priority frame cannot access the medium until the completion of transmission of the current frame. This can result in delay of up to a maximum-sized frame. Accumulation of such a delay at every hop will lead to an unacceptably large latency. This behavior is not limited to contention between a low priority and high priority traffic stream, but also amongst multiple streams of time critical traffic. These traffic streams can interfere with each others timely delivery

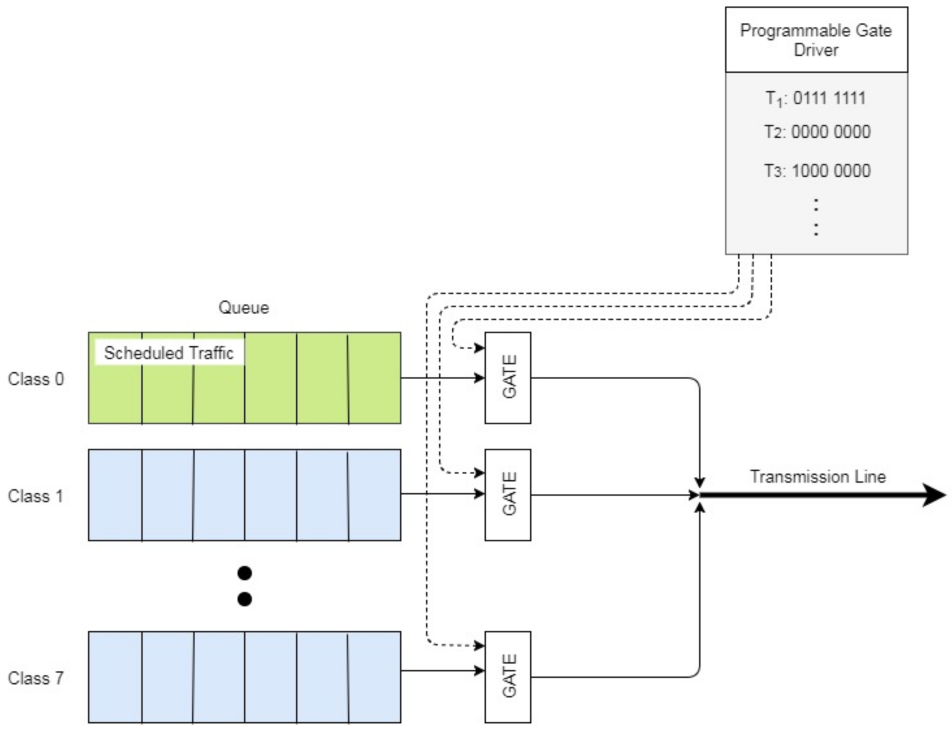


Figure 2.2: IEEE 802.1Qbv compliant Switch Architecture on Single Port

of frames. A mechanism is required to ensure coordination between interfering traffic streams.

To ensure a high priority traffic has guaranteed access to a medium at a specific instant in time, a protection window can be created allowing transmission of a specific traffic class over the medium for the duration of the time window. The implication of this mechanism is that the transmission of other traffic classes has to be stopped in advance to avoid interference. This means the transmission of lower priority traffic should be completed prior to the time (T1) when time critical traffic is granted access to the medium. The difference between T1 (time at which time critical traffic gains access to the medium) and T0 (time of completion of transmission of last frame of non time critical traffic) should be equal to maximum-frame size in the worst case. This difference can be guaranteed by introduction of guard bands. Guard bands ensure non time critical traffic is not transmitted between the start of guard band (T0) and beginning of transmission of time critical traffic (T1) as can be seen in Fig 2.1. The size of the guard band can be fixed to the size of one maximum-sized frame or be flexible if the implementation can determine the size of the queued frame, and the queued frame can be transmitted completely before the start of transmission of time critical traffic. [14]

Switches in a network should be compliant with the specification developed by Time-Sensitive Networking Task Group (TSN-TG) to be able to provide the above mentioned protocols. The architectural model of a 802.1Qbv compliant switch is shown in figure 2.2. Each port in the switch has eight queues, a transmission gate for each queue and a programmable gate driver. An incoming frame is processed by the switch and categorized in to one of the following traffic classes, scheduled traffic, credit-based shaper traffic and other traffic. These traffic classes are uniquely identified by 3 bit (VLAN) tag encoded priority values and are placed in separate queues. A traffic class is able to access the transmission medium only if the transmission gate of the queue it is placed in is open. The gate opening event allows the transmission of a queued frame and the gate closing event terminates the access to the medium. These events are controlled by the gate driver program. The gate driver program is a set of SetGateStates operations. Each operation consists of a time delay parameter and GateState parameter. The time delay parameter indicates the time delay after the gate operation is executed until the next operation occurs. The GateState is a eight bit vector, each bit reflecting the state (Open or close) of each gate. These GateState operations are set according to the transmission schedule. [14]

The Time Sensitive Network Task Group (TSN-TG) does not define a standard method to generate transmission schedules. Ongoing research has resulted in development of several methods to generate transmission schedules. In this thesis we have considered two independent methods which are discussed in the next section.

2.2 Scheduling

The Time Sensitive Network switches are designed to handle frames of Time Critical Traffic. However, in a 802.1Qbv compliant switch the time critical traffic is referred to as time triggered traffic, since the transmission of each of these frames is dictated by the time stamp at which they gain access to the transmission medium. Computation of transmission schedule for time triggered traffic in time sensitive network requires as input the set of time triggered flows and their routes. The schedule generated should guarantee delivery of time triggered traffic data before its deadline and optionally try to achieve minimal network delay for each flow. Of the several method formulated to solve the problem of generation of transmission schedule, we shall discuss the following two.

2.2.1 No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)

It was developed at University of Stuttgart by Dürr et al. Their approach of solving the problem of generating a transmission schedule for time critical traffic flows in time sensitive network is to map it to the classic Job Shop Scheduling problem.

Job Shop Scheduling problem is an operational research problem. It comprises of a set of machines and jobs, each job consisting of a sequence of operations that have to be executed in an order. The constraint is that only one operation can be processed on a particular machine at any given time, additionally the no-wait property implies that once a job has started it cannot be interrupted. The aim is to generate a schedule specifying the starting time of each operation of each job. Job shop scheduling additionally tries to minimize the makespan. Makespan is the maximum of finishing time of the last operation of the given set of jobs. This means it tries to finish the given set of jobs as fast as possible.

To map packet scheduling problem to job shop scheduling problem the ports of the switches and NIC of end nodes map to the machines. The time triggered flows corresponds to the jobs. Each flow is a sequence of transmission operations thus, each transmission operation maps to an operation in Job Shop Scheduling (JSP). Both sets of operations are required be executed in a sequential order. The network delay can be broken down into four components, transmission delay, processing delay, propagation delay and queuing delay. The operation time of a machine is mapped to the transmission delay of a packet on a link. Propagation delay and processing delay are also taken into consideration, but for each switch and each link in the network the values taken are the same. This approach adopts the no wait property by eliminating queuing delay. The frames of time critical flow are to be transmitted immediately after processing of the frame without waiting in the queue of outing going port.

The objective of the method "No wait packet scheduling problem (NW-PSP)" is to minimize the flowspan. Flowspan is a term equivalent to makespan in Job Shop Scheduling (JSP), it is defined as the finishing time of the flow finishing last. It is subjected to the constraint that no two flows can be transmitted over the same link at the same time. The objective and the constraint are formulated using Integer Linear Program (ILP) and solved using an ILP solver. Thus, successfully generating a transmission schedule for time triggered traffic for a single hyper-cycle. A hyper-cycle is a time period of length equal to the Least Common Multiple (LCM) of period of all the time triggered flows to be scheduled.

The researchers have successfully proven that minimizing the flowspan will result in a compact schedule. A compact schedule means that the time triggered flows will be pressed towards the start of the scheduling cycle instead of being spread across the whole cycle. A compressed schedule also means frames of different time triggered flows will be scheduled back to back thus, reducing the number of gate events. This helps reduce the number of guard bands required before the gating events. Thus, saving bandwidth which can be better utilized for transmission of best effort traffic.

However this method has a high time complexity. Their evaluations show that calculating an exact solution efficiently for large scenarios with large number of flows cannot be expected. Therefore, in oreder to improve the scalability of the algorithm they have developed heuristic, a Tabu search based algorithm for NW-JSP. [10]

2.2.2 Scheduling Real Time Communication in IEEE 802.1Qbv Time Sensitive Networks

It was developed at TTTech Computertechnik AG by Craciunas et al. The researchers aimed at solving the problem of generating transmission schedule for time triggered traffic in time sensitve network using Satisfiability Modulo Theories (SMT). Satisfiability Modulo Theories (SMT) are designed to determine the satisfiability of first order logical formulas against certain background theories like linear arithmetic etc.

The objective of this method is to find the values of frame offsets and queue assignment at each egress port over which flows are routed. The frame offset and queue assignment map to a gate opening event. While the queue assignment determines which gate at the egress port the event refers to, the frame offset is the time at which the event takes place. The gate closing event is marked by the duration of the frame itself.

In this method they have taken into account two set of constraints which is used to model the network over which the time triggered flows are to be transmitted. The first set of constraints are based on basic deterministic Ethernet constraints and second the 802.1Qbv constraints. The basic deterministic Ethernet constraints include:

- Frame constraint: The frame offset of all frames has to be greater than or equal to zero. The transmission period of the frame should be less the period of the flow.
- Link constraint: No two frames can be routed over the same physical link during the same time period.
- Flow transmission constraint: The propagation of frames of a flow must follow a sequential order along the routed path. Additionally a frame can be scheduled on the next link in its route only after it has been received completely from the previous link.

- End to end constraint: The difference between the arrival time at the destination and sending time at the source should be less or equal to the specified maximum time.

The 802.1Qbv constraints address the issue of egress interleaving. The transmission schedule controls the gate opening events, independent of the order of frames in the queue. The arrival of two flows from different links into a scheduled queue is non-deterministic in nature. This is due to the unavoidable synchronization error between devices in a network. The disturbance in order will introduce jitter during transmission of a frame. While this is benign in a single instance, accumulation of jitter over the entire path will impact the overall end to end latency of the flow. This problem can be solved either introducing deterministic order in a scheduled queue or to place the flows in separate queues. The researchers introduced the following set of constraints to tackle the issue of egress interleaving:

- Ideal scenario: Assuming the given network is perfect and guarantees no loss of frames, and frames are of constant size. A deterministic ordering can be guaranteed by ensuring the frames of the two flows are scheduled to arrive at different time.

Networks in reality are incapable of providing such guarantees. Loss of frame, frames of varying sizes and dropping of frame is common. In these cases one of the following constraints are used:

- Flow isolation constraint: if a frame of a flow has entered the queue, frame from another flow is not allowed to enter the queue until all the frames of the previous flow have been fully dispatched. The drawback of this constraint is that its very restrictive and reduces the search space for valid solutions.
- Frame isolation constraint: this constraint allows interleaving of frames of different flows. It ensures deterministic ordering by ensuring that frames of only one flow are present in a queue at a given time. The frame from another flow is allowed to enter the queue only if the queued frames has already been serviced.

These constraints are defined in term of frame offset and queue assignment. The constraints are the input to an SMT solver. The solver determines the satisfiability of the given set of constraints and if satisfiable generates a model. The model is a possible set of solution values for frame offset and queue assignment.[6]

We performed evaluations on the SMT method for different scenarios, with respect to varying number of flows, varying number of destinations for a multicast flow, varying number of links in the topology and varying the period of a flow. The period of a flow is the cycle of time with in which the flow must be delivered to its destination. The results of the evaluation are shown in graphs in Fig 2.3. Graph a) shows the time required by the SMT solver to generate a schedule with respect to number of unicast flows. The

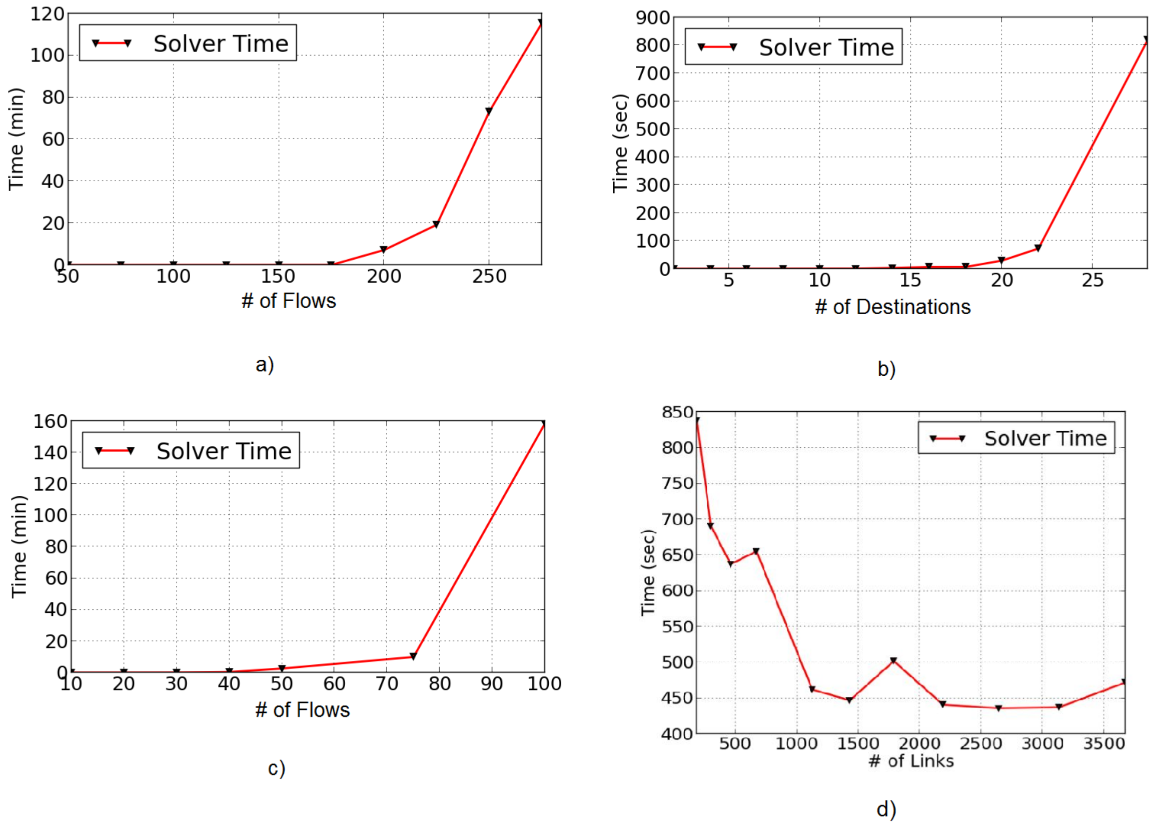


Figure 2.3: Graphs for evaluation of SMT method

number of flows range between (50-300). Graph b) shows the calculation time required by the solver with respect to the number of destinations. The solver was given a set of 50 flows as input and each flow had number of destinations ranging from (2-30). Graph c) shows the time required by solver to generate a schedule of flows with varying period. In each scenario the time complexity of the SMT is exponential in nature which means it cannot be used in large scenarios with large number of flows. Thus restricting its use. We also found that there was no direct relation between the number of links in a topology and the time required to generate a schedule, as observed in Graph d).

2.3 Software Defined Network (SDN)

The OSI model is used as the basic network architecture. Over the years as per the demand of the growing requirements, several protocols and standards have been developed based on this model. This has significantly increased the complexity of the networking infrastructure. As a result the following problems have been introduced:

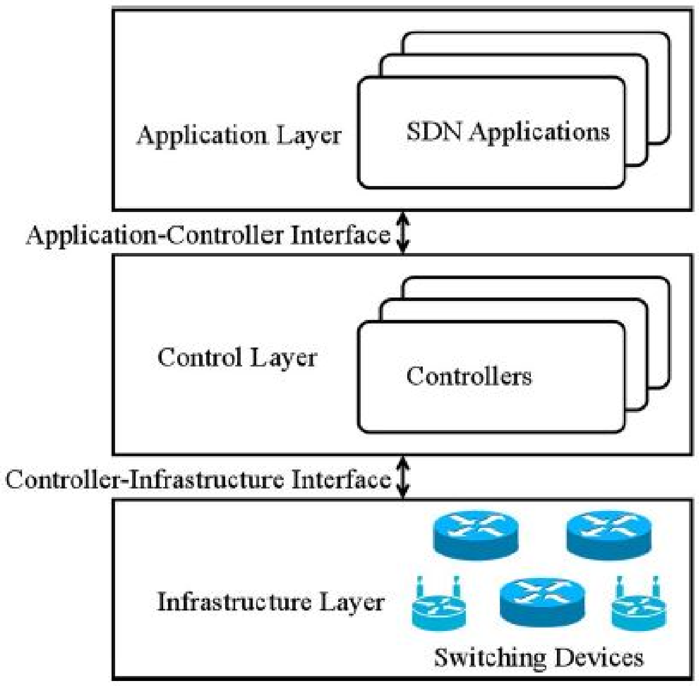


Figure 2.4: Software Defined Network (SDN) Reference Model [20]

- Limited flexibility: Protocols are hard coded into switches and routers. Introduction of new protocols require support of the manufacturers and are not easy to implement.
- Increased complexity: Switches and routers are provided with a large set of protocols. Most of which the user does not require.
- Separation of network and application: Application views the network as a medium for transmitting bytes of data. Network views the application as a generator of load. An integrated view could result in higher performance of application and higher utilization of the underlying network.

To address these issues the Open Networking Foundation (ONF) has been focused towards development and standardization of Software Defined Network (SDN). It defines Software Defined Network (SDN) as a network architecture with the primary goal of decoupling control plane from data plane and allow efficient management and operation of the network using software programs. [20]

The reference model of Software Defined Network (SDN) consists of three layers, infrastructure layer, control layer, and application layer as shown in Fig 2.4. The infrastructure layer is also known as the data plane, it comprises of network devices like switches and routers. The function of data plane devices is processing and forwarding of packets based on the rules defined by the controller, and collecting network status (network topology, traffic statistics, and network usages) and sending it to the controller. The controller is present in the control plane, it act as a bridge between application layer and infrastructure layer. The control layer communicates with infrastructure layer via south bound interface.

Interaction with application layer is done via north bound interface. The application layer contains Software Defined Network (SDN) applications designed to fulfill user requirements. Through the programmable platform provided by the control layer, Software Defined Network (SDN) applications are able to access and control switching devices at the infrastructure layer.

The benefits of using a Software Defined Network (SDN) can be viewed using the example of routing protocols. Traditionally routing in a network is distributed in nature, which means each node in the network is responsible for maintaining a map of the network and continuously updating it based on the control information exchanged. Such protocols are complex and result in lower resource utilization. Centralized routing algorithm offered by Software Defined Network (SDN) allows a central node to have a global view of the network. This results in optimized routing of data flows, it allows higher utilization of network resources and is simpler than the existing approaches of routing. IEEE 802.1Qca developed by Time Sensitive Network Task Group (TSN-TG) uses Path Computation Element (PCE) Architecture to discover non-shortest paths. Path Computation Element (PCE) Architecture adopts Software Defined Network (SDN) architecture which enables a global view of the network, and efficient management and operation of network elements.

2.4 IEEE 802.1Qca

IEEE 802.1Qca is a standard developed by the Time Sensitive Network Task Group (TSN TG). It is a set of protocols and procedures aimed at providing explicit path control, bandwidth and stream reservation, and redundancy for data flows and distribution of control parameters for time synchronization and scheduling. It uses Intermediate System to Intermediate System (IS-IS) to control bridged networks and provides capability beyond Shortest Path Bridging (IEEE 802.1aq). It also enables the use of non-shortest path via explicit forwarding path control. IEEE 802.1Qca uses Path Computation Element (PCE) for route computation. [13]

In the following section we will be discussing IS-IS, Shortest Path Bridging and Path Computation Element to have an overview of the components that are a part of IEEE 802.1Qca standard.

2.4.1 Intermediate System to Intermediate System (IS-IS)

IS-IS is a link-state interior gateway protocol. An interior gateway protocol is used to exchange information for routing within an administrative domain or network. A link-state routing protocol operates by flooding link state information throughout the network. This information is used by each network node to build a database of the network topology, which is used as a map by the node to calculate the shortest path using Dijkstra's algorithm.

IS-IS is similar to Open Shortest Path First (OSPF) except it is a data link layer (layer 2) protocol and uses different terminologies. Since it is a Layer 2 protocol it does not use IP addressing. IS-IS addresses are called NETs, or network entity titles and consists of three parts: Area identifier, System identifier and NET selector. ISs exchange routing information via protocol data units (PDUs). They generate Link-State PDUs (LSPs) to exchange information about its neighbors and destination directly connected to it.

On receiving a LSP each IS independently runs the shortest path algorithm. The algorithm is based on the well-known Dijkstra algorithm for finding the shortest paths along a directed graph where the ISs are the vertices of the graph and the links between the ISs are edges with a non-negative weight. A two-way connectivity check is performed before considering a link between two ISs as part of the graph. This prevents the use of stale information in the LSPDB for example, when one IS is no longer operating in the network. The output of the shortest path first (SPF) is a set of tuples (destination, next hop). Multiple equal-cost paths are supported, in which case multiple next hops would be associated with the same destination [5].

2.4.2 Shortest Path Bridging (SPB)

IEEE 802.1aq or Shortest Path Bridging (SPB) is a network protocol developed with the aim of replacing spanning tree protocols: IEEE 802.1D, IEEE 802.1w, IEEE 802.1s. The Spanning Tree Protocol (STP) aims at building a logical loop-free topology for Ethernet networks. Thus, it creates a spanning tree within a network, and disables those links that are not part of the spanning tree, leaving a single active path between any two network nodes. This is viewed as a major disadvantage because its slow convergence time and wastage of bandwidth. Shortest Path Bridging (SPB) has been developed to enable the use of multiple paths. This provides a larger layer 2 topology, which can be used by allowing traffic to load share across all paths of a network, thus making bandwidth available for applications like real-time communication [19].

2.4.3 Path Computation Element (PCE)

Path Computation Element (PCE) is an external server application or a network component responsible for computing routes between a pair of nodes, subjected to constraints such as Quality of Service (QoS), policy, or price. To be able to calculate routes efficiently it requires the knowledge of available network resources for example nodes and links, constraints, connectivity, available bandwidth and link costs. Thus, it needs a global view of the network topology.[8]

The architecture of Path Computation Element (PCE) comprises of the following components:

- Path Computation Element (PCE): component that performs complex path computations based on traffic demand and network constraints.
- Path Computation Client (PCC): A client application or component requesting a path computation to be performed by the PCE.
- Traffic Engineering Database (TED): a collection of information about the topology of the network, its nodes, links and relationships, and the details of resources and their attributes.
- Path Computation Element Communication Protocol (PCEP): a TCP-based communication protocol the PCC uses to communicate its path and resource requirements to the PCE. It is also used by the PCE to reply to the PCC and between PCEs when they need to communicate with each other.

The advantage of using Path Computation Element (PCE) over traditional traffic engineering methods is that it allows for faster updating of path computation policies, reduces costs, and provides the ability to move away from path computation algorithms that are hard coded into the router by the vendor. It reduces the complexity of path computation in large, multi-domain networks. Each PCE has a global view of its domain, and communication between PCEs has increased efficiency of inter domain routing.

2.4.4 Routing Algorithms for IEEE 802.1Qbv Networks

Most applications demand the use of standard routing algorithms like Shortest Path First (SPF) and Equal Cost Multipathing (ECMP) to calculate routes in a network. These protocols aim at optimizing the number of hops in a path but, they may not be the best approach to route time triggered traffic in a time sensitive network. In the paper Routing Algorithms for IEEE 802.1Qbv Networks by Nayak et al. [9], researchers draw a direct relation between the routing of time triggered traffic and their schedulability. The scheduling method used in this paper is the one introduced in No-wait Packet Scheduling for IEEE Time sensitive Networks (TSN). As mentioned in section 2.2.1 the metric Flowspan determines the quality of the transmission schedule generated. The researchers argue that the flowspan of a transmission schedule is dependent on the number of flows whose forwarding operations are conflicting at an egress port of a switch. Therefore, reducing the number of conflicting flows would have a positive impact on the flowspan of the transmission schedule. This can be achieved by routing the time triggered flows in a manner that reduces the number of flows with overlapping paths. This will also impact the total time triggered traffic forwarded by a port.

The paper introduces the metric Maximum Scheduled Traffic Load (MSTL) and defines it as the maximum amount of time triggered traffic that is transmitted by a switch port in a network per cycle of a transmission schedule. MSTL is dependent on the routing algorithm used route the time triggered flows. Use of algorithms like Shortest Path First (SPF) for instance will create a bottleneck link in the network. A bottleneck link is the link over which maximum data units of time triggered flows are routed, increasing the number of conflicting flows and the value of MSTL. To achieve a lower value of MSTL the flows must be spread across the network. This paper establishes a direct relation between the MSTL and flowspan. To generate a transmission schedule with lower flowspan, the routing of flows must take into account the MSTL of the network.

The researchers have developed two Integer Linear Programming (ILP) based routing algorithms aimed explicitly at minimizing the MSTL of the network. The first algorithm, MSTL based routing optimizes the routes of the time triggered flows based on only the MSTL value. The second algorithm, MSTL+Hops based routing also considers the number of hops in each path. The results of evaluations show that the MSTL based routing algorithms yield better result than Shortest Path First (SPF) and ECMP but, occasionally MSTL based routing may route over longer paths in comparison to ECMP, impacting the flowspan of the schedule. However, MSTL+Hops based routing yields better schedule than other routing algorithms every time, but with a significant increase in cost.

The scalability of the MSTL based algorithms was tested with respect to the number of flows routed and the size of the topology. The algorithms' time complexity is exponential in nature. This impacts its usability in scenarios consisting of a large topology and large number of flows. Its scalability can be improved using heuristic algorithms.

2.5 Heuristic Algorithms

An optimization problem is the search for optimal solution from set of all possible feasible solutions. When the variables of an optimization problem are discrete in nature it is referred to as a combinatorial optimization problem. The routing of flows through a given network can be defined as a combinatorial optimization problem of exponential complexity. Since the time complexity of such problems is unacceptably high, sometimes it is sufficient to find an approximate solution in acceptable amount of time instead of an exact solution. This goal is achieved using heuristic algorithms.

Dictionary defines heuristics as an approach to find solution by exploration of possibilities rather than by following set rules. Heuristic algorithms are designed to find approximate solution to a complex problem in acceptable amount time by compromising on optimality, accuracy and precision. In this thesis we have used the Tabu Search algorithm to find optimal routes for time triggered flows.

A combinatorial optimization problem has a finite set of feasible solutions and each solution is characterized by a cost. The objective is to find a solution of minimum (or maximum) cost. We consider a problem which has a set of feasible solution. The neighbor structure is a set of feasible solutions obtained by partial modification of the current solution . The cost function maps the feasible solution to a real value [7].

Tabu search is a meta-heuristic based on local search optimization. To simplify the understanding of Tabu search we shall break down the algorithm into parts. We shall begin by discussing Local search algorithm [16].

Local search begins from an initial solution generated independently and it works in iteration where it replaces the current solution by a neighboring feasible solution until the stopping criterion is met. On meeting the stopping criteria it returns the best solution found. The drawback of this algorithm is it usually gets stuck in local optima.

Tabu search prevents the problem of getting stuck at the local optima by the following two steps:

- To avoid getting stuck in suboptimal region it occasionally allows a worsening move, resulting in a candidate solution whose objective function value is worse than the current solution. However, this ensures that the search moves beyond local optima.

- To prevent cycling to previously visited solutions and guiding the search towards unexplored regions it introduces Tabu list. It is a short term memory for storing already visited solutions. If the search reaches a candidate solution already present in the tabu list it is forced to discard it and move to another solution. The recording of complete solution and checking the current candidate solution against the tabu list can be expensive both in terms of memory and time. Therefore the tabu list records the last few transformations performed on the current solution and prohibiting reverse transformations. However they can also be based on key characteristics of the solutions or of the moves.

The elements of Tabu search discussed so far are known to be effective in delivery good results, but sometimes they may be responsible for stagnating the search process. This happens when the search prohibits a move present in the tabu list in spite of the move delivering a solution with an objective value better than that of the current best-known solution. To overcome this an algorithmic device called aspiration criteria was introduced. The simplest implementation of which involves allowing a move even if it is present in the Tabu list.

Since Tabu search is an iterative search, determination of a termination criteria is essential. The most common criteria is to halt after a fixed number of iterations, but this may not guarantee the discovery of the optimal solution to the problem. The second is to stop if the value of the objective function does not improve after a few iterations. The third is to stop when the objective reaches a specified threshold value.

3 Time Sensitive Network

3.1 System Model

Time Sensitive Network (TSN) is a deterministic network capable of providing real time communication. The architectural model of a Time Sensitive Network (TSN) comprises of the following elements: Centralized User Configuration (CUC), Centralized Network Configurator (CNC), Network Elements (e.g. Switches) and End Nodes as shown in Fig 3.1. It requires protocols developed by Time Sensitive Networking Task Group (TSN-TG) and other network protocols to deliver the required functionality. In this section we shall discuss the roles and functions of each element in a Time Sensitive Network (TSN) system.

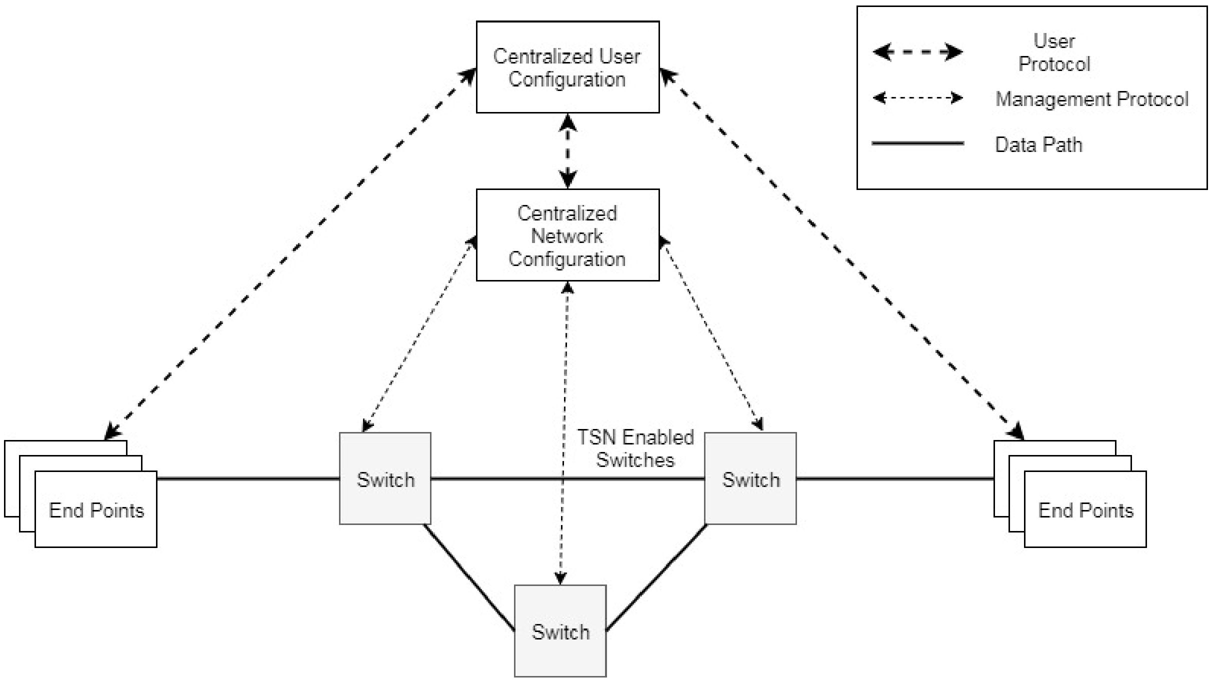


Figure 3.1: Time Sensitive Network (TSN) Architecture

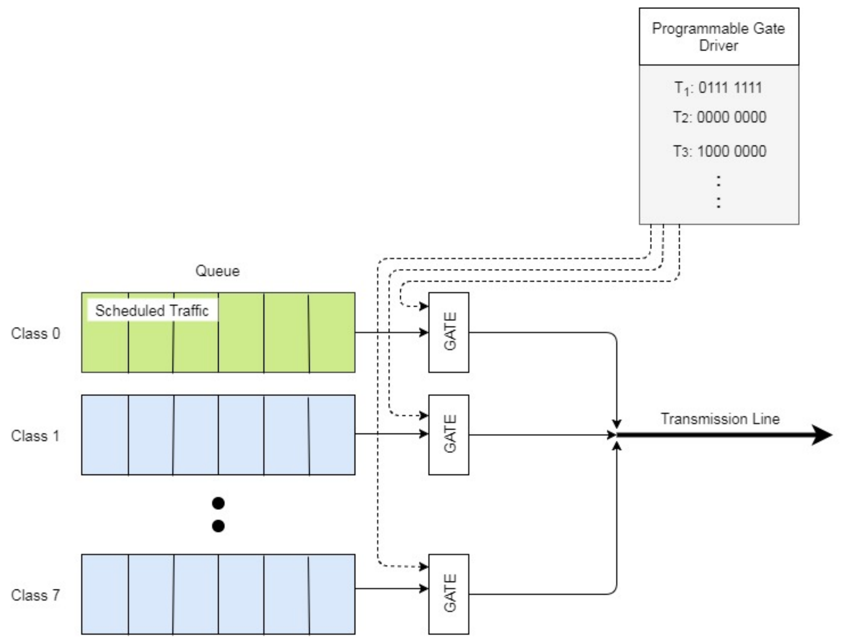


Figure 3.2: IEEE 802.1Qbv compliant Switch Architecture on Single Port

Flow: A flow is specified by a source host, destination host, the amount of data to be transmitted and period. Period is a duration of time, within which the data packet must be delivered to its destination.

Time triggered traffic are guaranteed timely delivery of frames. Real time applications with hard deadlines employ services of time triggered traffic for timely delivery of data. Best effort traffic on the contrary offer no such guarantees, therefore applications such as email which can tolerate delay in frame delivery employ the services of best effort traffic.

End Node: The End Nodes in a network are the source and/or destination of a flow. The end nodes in Time Sensitive Network (TSN) are of two types; TSN enabled and non-TSN enabled. Non-TSN enabled end nodes cannot handle time triggered traffic, it is the responsibility of the gateway switch the end node is connected to, to buffer the data packets sent from the node and transmit them into the network according to the transmission schedule. However, in this thesis we assume that all end nodes are TSN enabled which means the end node receives the transmission schedule from the Centralized User Configuration (CUC). The source node of a time-sensitive flow adheres to the assigned schedule and transmits the frames into the transmission medium at the assigned time. Additionally, it tags the packets of a scheduled traffic flow with IEEE 802.1Q header. The header includes a 3 bit Priority code point (PCP) field which refers to a traffic class and maps the frame to a priority level. The End Nodes should be compliant with IEEE 802.1AS or IEEE 1588 protocol which provides time synchronization with sub-microsecond precision.

Network elements (Switch/Router): It is responsible for the processing and transmission of data frames. A switch in Time Sensitive Network should be compliant with the following set of standards developed by Time Sensitive Network Task Group (TSN-TG):

- IEEE 802.1ASrev and/or IEEE 1588 for Time Synchronization.
- IEEE 802.1Qbv for Traffic Shaping.
- IEEE 802.1Qcc for Configuration from a Centralized Network Configurator (CNC)
- IEEE 802.1Qci for Potentially Time Based Ingress Policing

A switch compliant with IEEE 802.1Qbv specification is designed to offer differentiated services. Differentiated services classifies the traffic into classes, and allocates resources on per-class basis. This allows the TSN network to offer varying Quality of Service (QoS) guarantees to individual classes. In this thesis we restrict the network traffic classification to scheduled traffic and best effort traffic. The architecture of the switch comprises of eight queues per port, queue(s) on each port is reserved for scheduled traffic and the unreserved queues are used for best effort traffic. As an example in Fig 3.2 Class 0 is reserved for scheduled traffic and the remaining seven queues are for best effort traffic.

Corresponding to each queue is a gate which is regulated by a gate driver program. The gating events (open/close) determine if the frames in the queue have access to the transmission medium. For example, if the gate of Class 7 was open, the frames enqueued in Class 7 queue would be transmitted over the medium. The transmission would cease with the gate closing event. The gating events (open/close) are regulated by the gate driver program.

The gate-driver program is a set of SetGateStates operations which represent the gate events. The operations are a tuple entry comprising of two parts: time delay parameter and GateState parameter. The time delay parameter is relative to the start of the program. The GateState parameter is a bitmask that determines the gates which are opened at the mentioned time. For example, in Fig 3.2 at time T2 all the gates are closed and at time T3 only the gate for scheduled traffic is open. The gate driver program is a cyclic schedule of length T_{cycle}, hence after completion of a cycle it restarts. Typically, all the gate drivers in a given network are programmed with the same value of T_{cycle}.

Centralized User Configuration (CUC): It is an application that can communicate with the Centralized Network Configurator (CNC) and the end nodes. It is used to configure the end nodes by deploying the relevant configuration and programs on devices and is also responsible for distributing the transmission schedule to the end nodes.

The Centralized User Configuration (CUC) represents the control applications and the end devices in the Time Sensitive Network system. Therefore it makes flow requests to Centralized Network Configurator (CNC). A flow request comprises of the addresses of sender and receiver nodes, communication frequency, availability requirements, latency/jitter requirements and other information required to establish a time triggered flow.

Centralized Network Configurator (CNC): It is a node or an application that is the center of a Time Sensitive Network system. It has the global view of the underlying topology accumulated using Link Layer Discovery Protocol (LLDP). It is also aware of the capabilities of each network element and the overall network which includes link speed, delay in the network, PTP precision etc. The Centralized Network Configurator (CNC) is responsible for the computation of the transmission schedule for time triggered traffic based on the flow requests made by the Centralized User Configuration (CUC). It also programs the gate drives in the switches according to the generated transmission schedule. It is optional for the end nodes to receive the transmission schedule via Centralized Network Configurator (CNC). The Centralized Network Configurator (CNC) programs the network elements using management protocols like Simple Network Management Protocol (SNMP).

3.2 Problem Statement

Time Sensitive Network's ability to provide deterministic data delivery is due to the transmission schedules, which are generated and deployed by the Centralized Network Configurator (CNC). However, Time Sensitive Network Task Group (TSN-TG) provides no standard for generation of these schedules. Ongoing research has led to development of independent methods for generation of transmission schedule, of which we consider only two in this thesis refer Section 2.2. This thesis is focused on solving the following two problems.

Benchmark MSTL as a metric

The scheduling methods discussed in Section 2.2 successfully generate a transmission schedule. However, there exists no definition for what constitutes a quality schedule. In [10] by Duerr et al. researchers have introduced the parameter flowspan to access the quality of the transmission schedule generated by their method refer Section 2.2.1. They argue that a transmission schedule is of good quality if it is able to schedule transmission of time triggered traffic such that the flowspan is as low as possible. However, this parameter is unique to the scheduling method developed by them and cannot be mapped to other scheduling methods. In [9] by Nayak et al. researchers have established a direct relation between the quality of schedule (flowspan) and the routing algorithm used refer Section 2.4. They argue that in order to have schedules with lower flowspan the routing of the time triggered flow should account for the corresponding Maximum Scheduled Traffic Load (MSTL). The MSTL is the maximum amount of cumulative load transmitted over any link in the network. The parameter to evaluate the quality of a transmission schedule varies between the scheduling methods. The aim of this thesis is to prove the impact of load distribution on the quality of a schedule generated by different scheduling methods, and to benchmark MSTL as a metric that can be used to assess the load distribution by different routing algorithms across all scheduling methods.

Generic Routing Algorithm to Optimize MSTL

Input required to generate a transmission schedule is the set of time triggered flows that need to be transmitted through the network and the route of individual flow in the given set. Route is defined as a sequence of transmission operations over the links connecting the source of the flow and the destination(s) of the flow. These transmission operations are mapped to the gate opening events of a switch in a schedule. The influence of a routing algorithm on the MSTL value is demonstrated using the following example.

We have a network topology as shown in Fig 3.3 and three unicast time triggered flows that need to be scheduled:

- Flow1: src: H1, dst: H4, Data Length: 1000 bytes
- Flow2: src: H2, dst: H5, Data Length: 500 bytes
- Flow3: src: H3, dst: H6, Data Length: 500 bytes

The standard routing algorithms used is Shortest Path First (SPF). It is design to optimizing the number of hops in a route. The given flows will be routed over the Link$_{(S1,S2)}$, since it is the shortest path in the network for each flow. This results in an MSTL value of 2000 bytes on Link$_{(S1,S2)}$.

We now use the routing algorithm ECMP to route the flows. ECMP returns multiple equal cost routes. In this case it returns two routes for each flow. The first route is over Link$_{(S1,S3)}$, Link$_{(S3,S2)}$ and second route over Link$_{(S1,S4)}$, Link$_{(S4,S2)}$. The flows can be routed over either of the two routes and thus several combination to route all three flows are possible. If we take the scenario where Flow1 is routed over the first route, and Flow2 and Flow3 were routed over second route, the load distribution across the

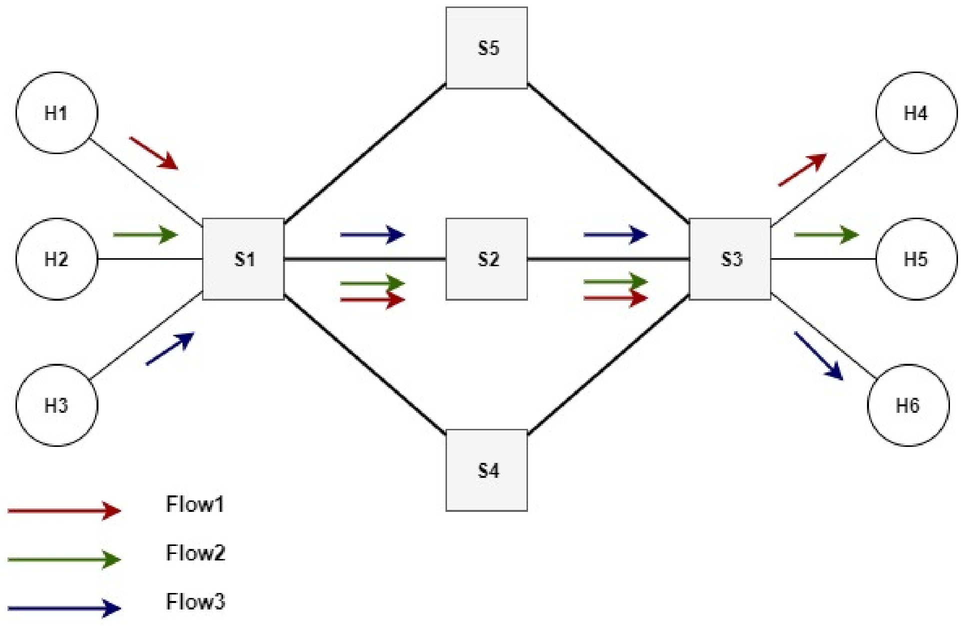


Figure 3.3: Using Shortest Path First (SPF) to route the given flows

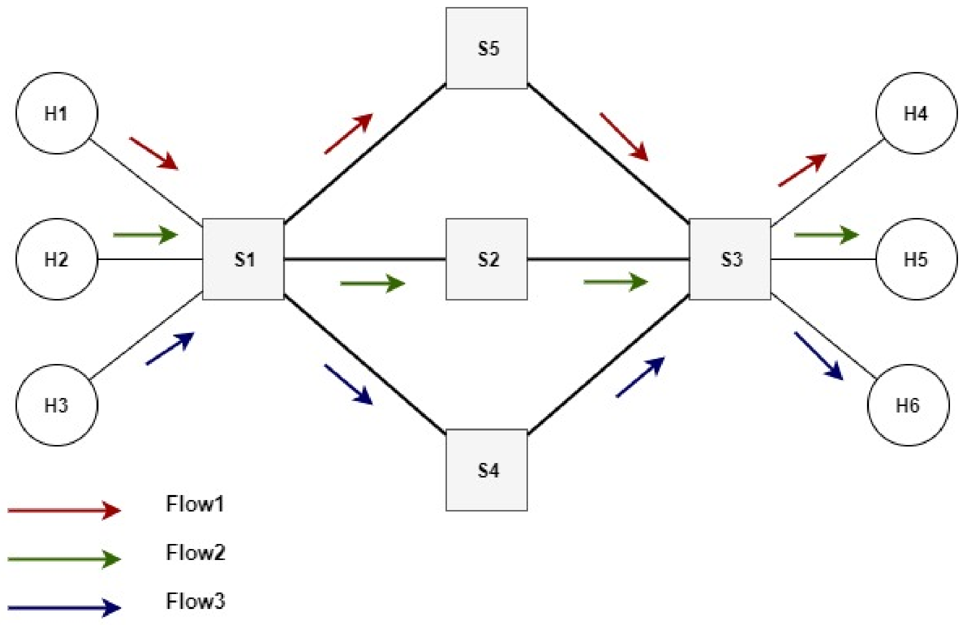


Figure 3.4: Using ECMP to route the given flows

network would be optimal. The corresponding MSTL value for this scenario would be 1000 bytes on each link of the core network.

In the paper [9] by Nayak et al, researchers have defined the routing of flows in terms of a combinatorial optimization problem, and have solved it using integer linear programming (ILP). The objective of the ILP program is to route the flows over the network such that the value of MSTL is minimal. However, the scalability of this approach is exponential in nature and cannot be used for large scenarios. Therefore, the aim of this thesis is to develop a generic routing algorithm that optimizes the value of MSTL and is scalable in nature.

4 Impact of Routing on Scheduling Methods

Due to the lack of a definition for what constitutes as a quality schedule, this section is aimed at introducing the definition of a quality schedule and a metric to measure this parameter for each scheduling method discussed in this thesis. In this section we establish the relation between MSTL value and the quality of a schedule.

4.1 No-Wait Packet Scheduling for IEEE Time Sensitive Networks (TSN)

In [10] by Duerr et al., researchers measure the quality of a schedule by the metric flowspan. They have established that a lower value of flowspan implies a better quality schedule refer section 2.2.1. However, this metric is unique to the scheduling method developed by them and cannot be mapped to other methods.

Routing time triggered flows using standard algorithms like shortest path first (SPF), which aim at optimizing the number of hops leads to high contention between the flows on the egress port of a switch. This means a number of time triggered flows are competing to gain access to the transmission medium. This delays teh transmission of flows increasing the flowspan of the schedule. To generate schedules of better quality the aim is to decrease the flowspan of the schedule, this can be achieved only if the contention at the egress port of each switch in the network is decreased. To decrease the contention the time triggered traffic load has to be distributed over the network such that the number of flows with overlapping routes is minimum.

In [9] by Nayak et al., researchers introduce the metric MSTL to measure this distribution of data load. Using algorithms like shortest path first (SPF) increases the MSTL value, consequently increasing the flowspan of the schedule. Using alternate routing algorithms which transmit flows over alternate paths and not just the shortest available path decreases the MSTL and consequently the flowspan of the schedule. Hence, we observe a direct relation between the flowspan of the schedule and the MSTL value.

In chapter 6 we present the evaluations assessing the impact of the routing algorithms discussed later in chapter 5 on the MSTL value and consequently on the flowspan of the schedule generated by No-wait Packet Scheduling Problem approach.

4.2 Scheduling Real Time Communication in IEEE 802.1Qbv Time Sensitive Networks

In [6] by Craciunas et al., the length of a transmission schedule is equal to a hyper-cycle. A hyper-cycle is a time period of length equal to the Least Common Multiple (LCM) of period of all the time triggered flows to be scheduled. The scheduling method tries to accommodate transmission of maximum number of time triggered flows in a single hyper-cycle. It must be noted in this method the end to end latency of a flow is also taken into consideration while scheduling. End to end latency is the temporal duration between the instance of time a frame is transmitted from source and the instance it is delivered at the destination. The number of flows that can be scheduled is however limited by the routing algorithm used. The SMT method utilizes shortest path first (SPF) routing algorithm as per the paper. As demonstrated by the given example the shortest path first (SPF) algorithm limits the number of time triggered flows that can be scheduled in a single hyper-cycle.

Given a topology as shown in Fig 4.1 and the following set of time triggered flows:

Flow1: (src:H1, dst:H4, size:5, period:30, end2end-Latency: 25)
Flow2: (src:H2, dst:H5, size:5, period:30, end2end-Latency: 25)
Flow3: (src:H3, dst:H6, size:5, period:30, end2end-Latency: 25)
Flow4: (src:H1, dst:H6, size:5, period:30, end2end-Latency: 25)

Assuming the processing delay, queuing delay and propagation delay are not considered, and the transmission delay is equal to the length of the data packet. Therefore, in the given example it would take 5 units of time to transmit Flow1 over a single link.

The hyper-cycle for the given example is equal to 30 units of time. The transmission for Flow1 over the shortest path $\text{Link}_{(S1,S2)}$ and $\text{Link}_{(S2,S3)}$ begins at time 0 units and ends at time 20 units. Flow2 and Flow3 are routed over the same path as Flow1. Flow2 begins at time 5 units and ends at 25 units. Flow3 begins at 10 units and ends at 30 units. If Flow4 is to be scheduled over the same path, it would begin at 15 units and end at 35 units. This is not a feasible solution since it exceeds the length of the hyper-cycle.

If an alternate routing method is used which allows routing over different routes as shown in Fig 4.2, all the flows can be accommodated in the schedule. Flow1 is

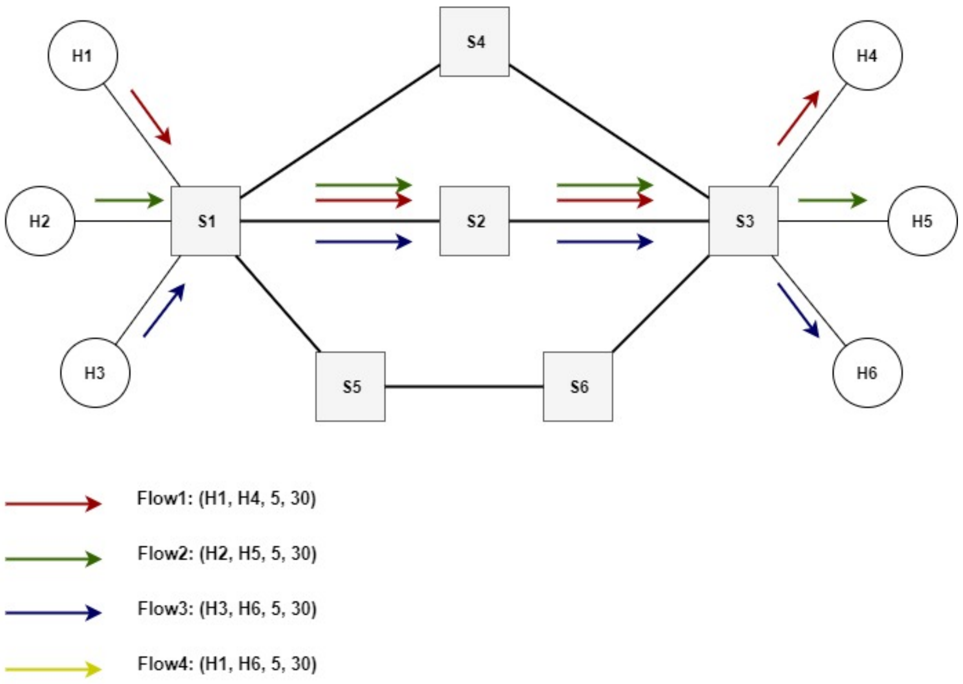


Figure 4.1: Routing using Shortest Path Routing (SPF) for the given set of flows (SMT method)

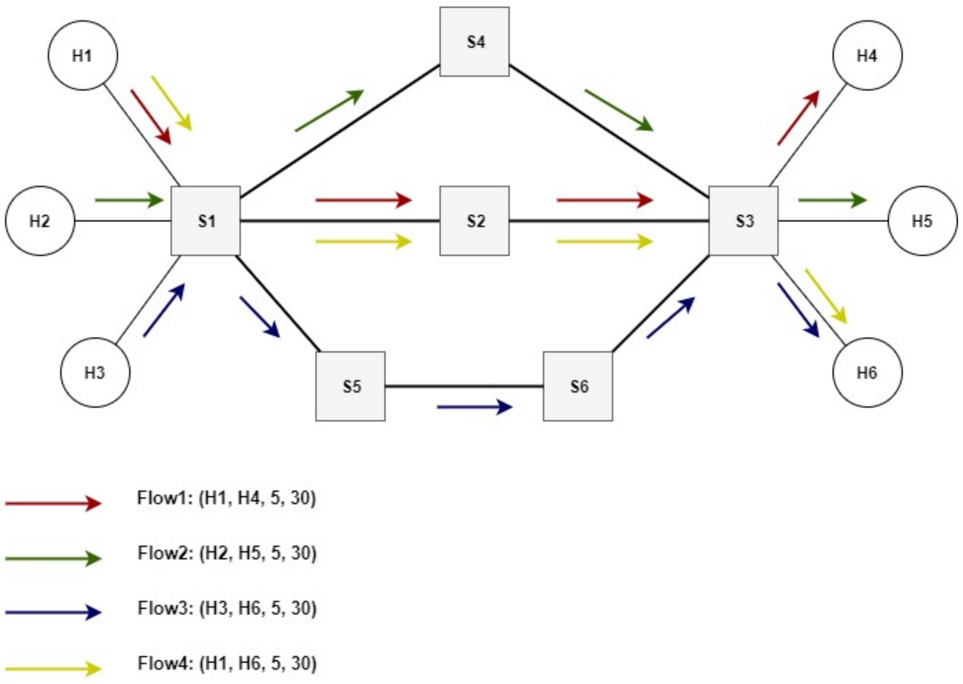


Figure 4.2: Routing using alternate routing approach for the given set of flows (SMT method)

transmitted between 0 - 20 units of time, Flow2 between 0 - 20 units of time, Flow3 between 0 - 25 units of time and Flow4 between 5 - 25 units of time.

However, the solution presented above would not be feasible if the end to end latency of Flow4 was equal to 20 units of time. Routing it over the same route as shown in Fig 4.2 would violate the end to end latency constraint refer 2.2.2. Hence, scheduling of the flow would not be possible in spite of utilizing an alternate routing algorithm. It is imperative to consider the number of hops in a route such that it does not violate the end to end latency constraint.

Hence, in this thesis we assess the quality of a transmission schedule generated by SMT approach via the number of time triggered flows it is able to schedule in a single hyper-cycle. We refer to this value by the metric "Breaking Point". Breaking point is the number of flows beyond which no more flows can be scheduled for the given hyper-cycle. Using the above example we have successfully demonstrated a direct relation between the routing algorithm used and breaking point. The routing of flows impact the MSTL value, it is feasible to establish a similar relation between MSTL and breaking point. In chapter 6 we provide evaluations proving a direct relation between MSTL and breaking point. Hence, establishing that MSTL can be used as a standard metric by any scheduling algorithm to assess the impact of load distribution by a routing algorithm on the quality schedules generated by them.

In chapter 6 we present the evaluations assessing the impact of the routing algorithms discussed in chapter 5 on the breaking point of the schedule generated by SMT approach.

5 Routing Algorithms

In this chapter we introduce the routing algorithms aimed at optimizing the MSTL value in a given network. The algorithms discussed are weighted Shortest Path First (wt SPF), weighted Equal Cost Multi Path (wt ECMP) and the heuristic algorithms based on Tabu search. In order to facilitate a clear understanding of the algorithms we begin by introducing terminologies used through out the chapter, followed by the algorithms.

5.1 Terminology

- **Graph**: A graph is syntactically defined as $G = (V, E)$, where V is the set of vertices and E is the set of edges. The vertices in the graph represent the nodes (host or switch) in the network. The edges are links that connect a pair of nodes and is defined as $E \subseteq V \times V$. If the edges in a graph are directional it is termed a directional graph else a graph consists of bidirectional edges.
- **Weighted Graph**: A weighted graph is a graph in which each edge is given a numerical weight. This numerical value is considered an attribute of the edge and is stored in a data structure called $weight$. A weighted graph is referred to as $topology$ in the discussed algorithms.
  In the following algorithms the attribute weight of an edge is equal to the cumulative data bytes transmitted over the edge. A data structure $load$ independently stores this value for each edge.
- **Flow**: A flow in a network is denoted as a tuple represented by f $\equiv$ (src, dst, period, size). src is the source node, dst is the destination node(s), period is duration of time within which the data must be delivered to its destination and size is the total number of bytes transmitted by the flow in each period.
- **Set of Flows**: It is a collection of time triggered flows that are to be scheduled in a time sensitive network (TSN). It is represented as F $\equiv$ { f_1, f_2, .. f_n }
- **Route/Path of a Flow**: It is defined as a sequence of transmission operations over the links of the network that connect the source of the flow to the destination(s) of the flow.

Algorithmus 5.1 Weighted Shortest Path First Algorithm

```
 1: procedure WEIGHTEDSPF
 2:     $route \leftarrow$ {}
 3:     $load \leftarrow$ {}
 4:     for all $flow$ in $F$ do
 5:         $(src, dst, period, size) \leftarrow flow$
 6:         $route[flow] \leftarrow$ DIJKSTRAPATH($topology, src, dst, weight$)
 7:         for all $edge$ in $route[flow]$ do
 8:             $load[edge] \leftarrow load[edge] + length$
 9:         end for
10:         SETEDGEATTRIBUTE($topology, weight, load$)
11:     end for
12:     return $route$
13: end procedure
```

- **MSTL**: Maximum Scheduled Traffic Load is the maximum amount of time triggered traffic that is transmitted by a switch port in a network per cycle of transmission schedule.

5.2 Weighted Shortest Path First Algorithm

Dijkstra's algorithm computes the shortest distances between vertices in a graph. The edges of the graph can be assigned numerical values which represents the cost of transmitting over the edge. The cost of a route is a sum of cost of each edge in the route. In Shortest Path First (SPF) routing algorithm the value assigned to each edge is one. Hence in SPF algorithm the cost of a route represents the number of edges a flow has to traverse to reach its destination. SPF is designed to optimize the number of hops in a route and return as solution a single route. It does not take into account the load on the edges of the network.

Weighted Shortest Path First (wt SPF) uses the same principles as SPF. However, instead of assigning a fixed value as the cost of an edge, it uses the load (cumulative data bytes transmitted) on each edge as its cost. The consequence of this is the cost of a route is a sum of load on each edge in the route, and the route with least cumulative load is selected. This enables distribution of load over the network and avoids the development of a bottleneck edge for as long as possible. A bottleneck edge is an edge over which maximum load is transmitted, the occurrence of which in time sensitive network (TSN) impedes the timely delivery of time critical data.

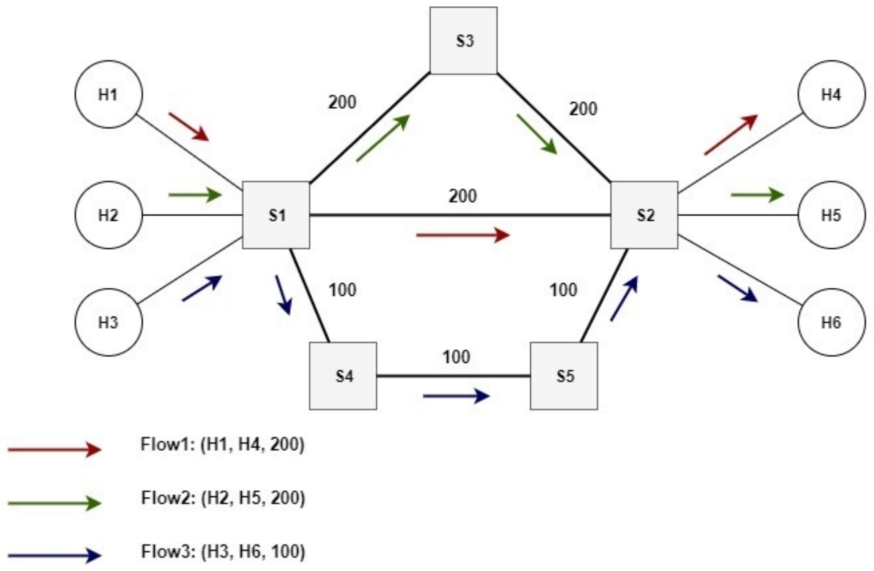


Figure 5.1: Routing flows using Weighted Shortest Path First (wtSPF) Algorithm

We use the following example to demonstrate the working of weighted SPF algorithm. We have a network topology and three time triggered flows as shown in Fig 5.1. If Shortest Path First (SPF) algorithm was used to route the flows, they would be routed over the Link$_{(S1,S2)}$ and the MSTL value would equal to 500 bytes. However, weighted Shortest Path First (wt SPF) algorithm distributes the flows over different routes in the network as shown in Fig 5.1 and results in MSTL value of 200 bytes.

A new time triggered flow (Flow4:= (H1, H5, 100)) needs to be routed over the network. There are three possible routes between nodes H1 and H5:

- route1 over Link$_{(S1,S2)}$
- route2 over Link$_{(S1,S3)}$ and Link$_{(S3,S2)}$
- route3 over Link$_{(S1,S4)}$, Link$_{(S4,S5)}$ and Link$_{(S5,S2)}$

As can be seen in Fig 5.1 the cost of route1 equals 200, route2 equals 400 and route3 equals 300. The wt SPF algorithm selects route1 due to its low cumulative cost, which results in a MSTL value of 300 bytes as shown in Fig 5.2. This demonstrates the limitation of wt-SPF algorithm of not being able to optimize the MSTL value. If the Flow4 was routed over route3, the MSTL value would be equal to 200 bytes as shown in Fig 5.3. A longer route (route3) might have a higher cumulative cost is comparison to a shorter route (route1), but routing over longer routes presents an opportunity to distribute the load of the network and reduce the MSTL value. Therefore, a routing algorithm for time sensitive network (TSN) should be able to determine the circumstances in which the routing of flows over longer routes is beneficial.

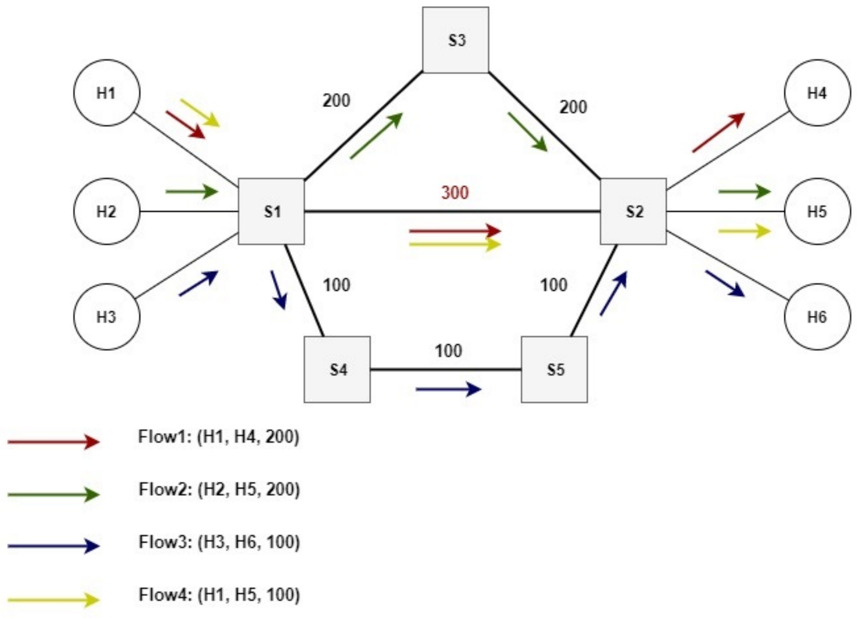


Figure 5.2: Increased MSTL due to routing of Flow4 using Weighted Shortest Path First (wtSPF) Algorithm

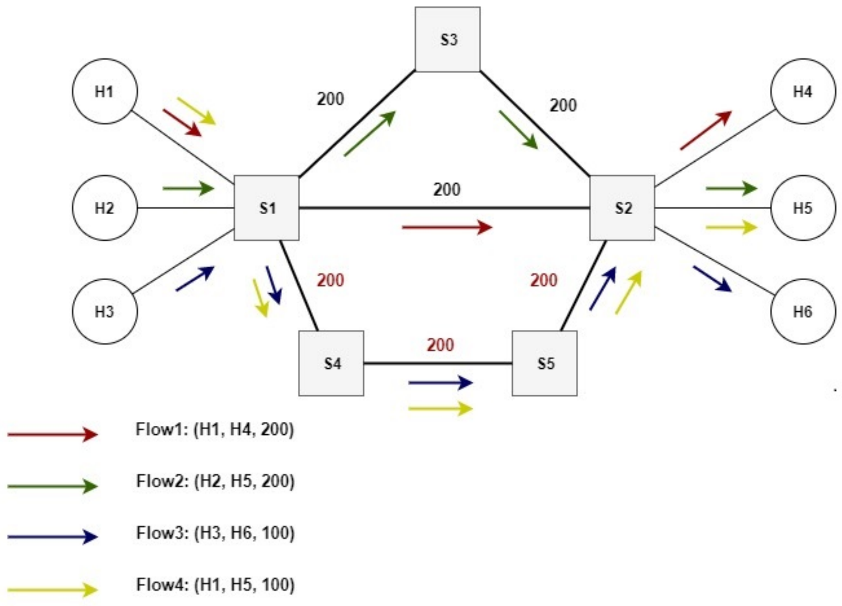


Figure 5.3: Alternate routing of Flow4 to optimize the MSTL

Algorithmus 5.2 Weighted Equal Cost Multi Path Algorithm

```
1: procedure WEIGHTEDECMP
2:     $route \leftarrow \{\}$
3:     $load \leftarrow \{\}$
4:     for all $flow$ in $F$ do
5:         $(src, dst, period, size) \leftarrow flow$
6:         $P \leftarrow$ ALLSHORTESTPATHS($topology, src, dst$)
7:         $pathLoad \leftarrow \{\}$
8:         for all $path$ in $P$ do
9:             $pathLoad[path] \leftarrow max(load[edge]$ for $edge$ in $path)$
10:        end for
11:        $route[flow] \leftarrow path$ with $min(pathLoad)$
12:        for all $edge$ in $route[flow]$ do
13:            $load[edge] \leftarrow load[edge] + length$
14:        end for
15:    end for
16:    return $route$
17: end procedure
```

5.3 Weighted Equal Cost Multi Path Algorithm

Equal Cost Multi Path (ECMP) algorithm returns multiple equal cost routes. The cost function is usually based on the number of hops in a route, similar to SPF algorithm refer Section 5.2. However, the algorithm lacks a mechanism for selecting a route from the returned list of paths. The selection is usually random, hence it provides no control over the distribution of data load.

To tackle this limitation we introduce the weighted ECMP (wt-ECMP) algorithm. It works on the same principle as ECMP algorithm, it searches for all shortest routes in a network between a pair of nodes and returns it as a list refer Alg 5.2 Line 6. A route is a sequence of edges from source to destination, each edge‘s load is the cumulative data it transmits. The edge with the highest load is determined and the value of its load is assigned to the route. The algorithm parses through the returned list of routes, for each route it identifies the maximum load transmitted and assigns this value to the route. This value of maximum load (max-load) is used as a parameter for deciding which route to choose. The wt-ECMp algorithm aims at routing over a route with the lowest max-load. This guarantees distribution of load over the network and a gradual increase in MSTL value. The wt-ECMP algorithm accounts for the number of hops as well the the load on the route while routing the flows.

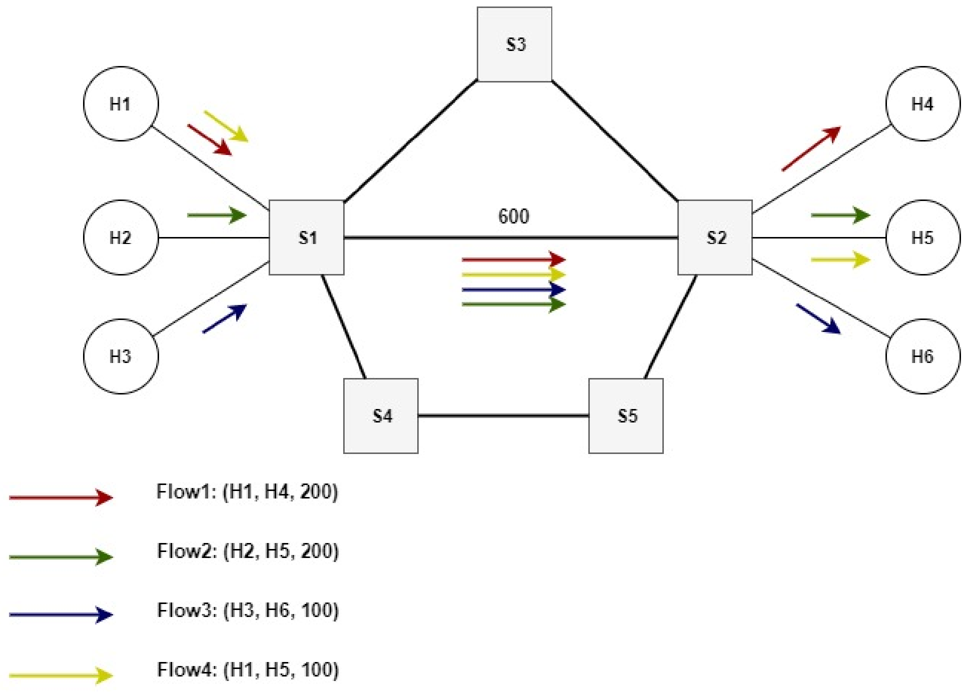


Figure 5.4: Increased MSTL due to routing of flows using Weighted ECMP Algorithm (wtECMP)

However there is a major drawback for this algorithm. We demonstrate this using the same example used for wt-SPF algorithm. As shown in Fig 5.4 we want to route the flows over the given network. The wt-ECMP algorithm returns a single route over Link$_{(S1,S2)}$ as the shortest route for each flow. This is same as SPF algorithm and the resulting MSTL value is 600 bytes. In spite of a highly connected network topology the algorithm fails to take into consideration longer routes over which the flows can be routed. Thus, making the algorithm dependent on the underlying topology to have multiple routes of same length between nodes, over which it can distribute the data load. This severely limits its use in diverse scenarios.

The results of both weighted Shortest Path First and weighted ECMP algorithm stress on the need for an algorithm which is able to route over longer routes in order to reduce the MSTL value. However, this approach comes with its own limitations. While we understand that decrease in MSTL decreases the flowspan of a schedule. A longer route leads to an increase in the number of transmission operations and consequently increasing the number of gating events. These gating events impact the flowspan of the schedule. With the increase in number of gating events the flowspan also increases. Therefore, it is imperative the routing algorithm in a time sensitive network (TSN) takes into account the MSTL value as well as the number of hops in a route while routing a flow.

Algorithmus 5.3 Tabu Search: Initial Solution

```
1: $route \leftarrow$ {}
2: $load \leftarrow$ {}
3: $flowsRoutedOnLink \leftarrow$ {}
4: for all $flow$ in $F$ do
5:     $(src,\ dst,\ period,\ size) \leftarrow flow$
6:     $MaxLoadLink \leftarrow edge$ with $max(load)$
7:     remove $MaxLoadLink$ from $topology$
8:     $route[flow] \leftarrow$ SHORTESTPATH($topology$, $src$, $dst$)
9:     add $MaxLoadLink$ to $topology$
10:    TABU SEARCH: UPDATE LOAD
11:    $lowestMSTL \leftarrow max(load)$
12:    $bestSolution \leftarrow route$
13: end for
14: return $lowestMSTL$, $bestSolution$
```

Algorithmus 5.4 Tabu Search: Update Load

```
1: for all $edge$ in $route[flow]$ do
2:     $load[edge] \leftarrow load[edge] + length$
3:     $flowsRoutedOnLink[edge] \leftarrow flow$
4: end for
```

5.4 Tabu Search Algorithm

The routing algorithm introduced in this section is based on the heuristic algorithm Tabu search. Tabu search is a local search algorithm, which begins from an independently generated initial solution and searches for a neighboring feasible solution with a better objective function value. Local search algorithm is likely to get stuck in a local optimum. To overcome this limitations and guide the search towards the global optimum, tabu search algorithm uses tabu list. The tabu list prevents cycling to previously visited solutions and guides the search towards unexplored regions. The routing algorithm can be broken down into four segments; Initial solution, neighborhood search, tabu list and termination criteria. We discuss each segment independently to enable clear understanding of its functionality in the algorithm.

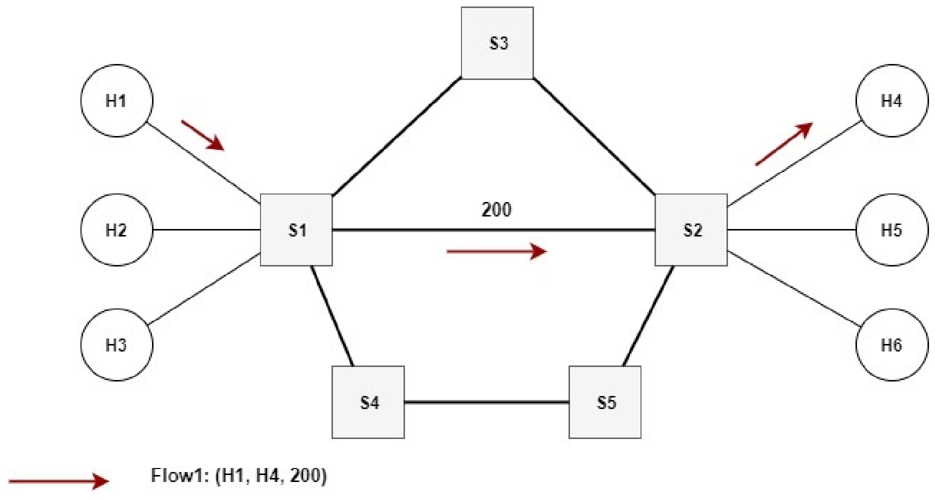


Figure 5.5: Initial Solution: routing Flow1 using Tabu Search Algorithm

5.4.1 Initial Solution

Tabu search algorithm begins by generating an initial solution. Since Tabu search is a local search algorithm, our aim is to begin with an initial solution with minimum (or maximum) possible objective function value. This gives the algorithm an opportunity to discover better solutions faster. The initial solution in our problem entails to routing each flow in the given set of flows over the network. While routing the flows we aim at realizing the following two conditions:

- Finding the shortest path: routing over a longer path increases the number of transmission operation. This impacts the quality of the schedule (flowspan).
- Minimizing the MSTL value: a lower MSTL value implies the load has been distributed over the entire network, thus reducing the number of flows in contention with each other.

To achieve the first condition we use the shortest path first (SPF) algorithm. This ensures the flow is routed over the shortest path in the network. For example, in Fig 5.5 Flow1 is routed over the shortest path over Link$_{(S1,S2)}$. To encourage gradual increase in value of MSTL, the algorithm identifies the link with maximum load ($MaxLoadLink$) and stops considering it temporarily. In Fig 5.5 Link$_{(S1,S2)}$ has the maximum load, it is removed from the topology and Flow2 is then routed over the next available shortest path, over Link$_{(S1,S3)}$ and Link$_{(S3,S2)}$ as shown in Fig 5.6. On completion of routing the $MaxLoadLink$ Link$_{(S1,S2)}$ is added back to the topology.

To route Flow3 in the next iteration the algorithm selects the $MaxLoadLink$, three links with the same load exist as shown in Fig 5.6, a random link is selected and removed

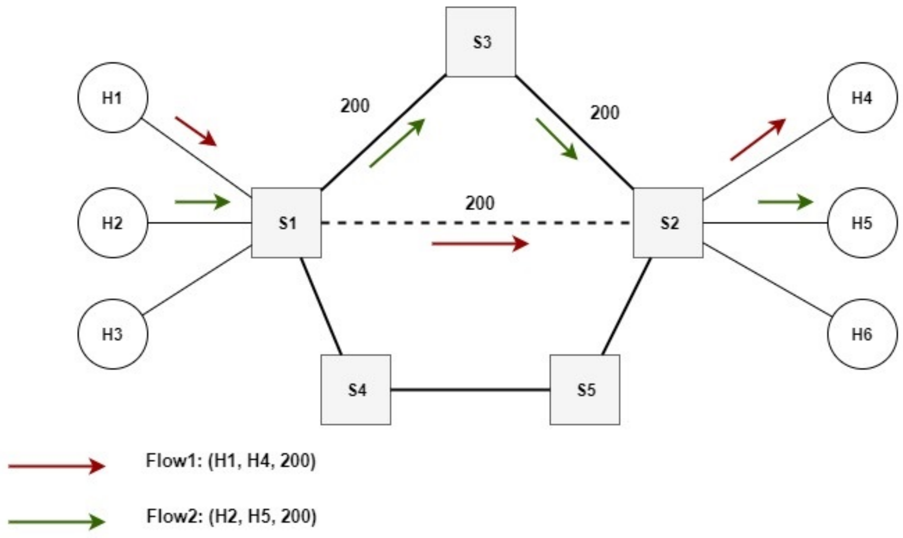


Figure 5.6: Initial Solution: routing Flow2 using Tabu Search Algorithm

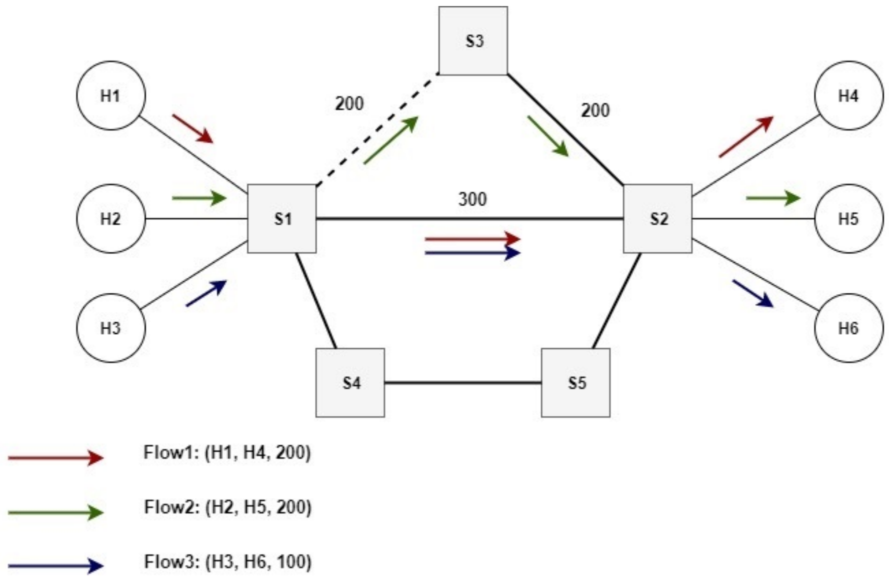


Figure 5.7: Initial Solution: routing Flow3 using Tabu Search Algorithm

from the topology. Assuming Link$_{(S1,S3)}$ is selected as MaxLoadLink, the flow is routed over the shortest path over Link$_{(S1,S2)}$ as shown in Fig 5.7, the resulting MSTL value is of 300 bytes. In comparison, if Shortest Path First (SPF) algorithm was used without removing the $MaxLoadLink$ the resulting MSTL would be 500 bytes over Link$_{(S1,S2)}$. Our algorithm returns paths resulting in low MSTL value, while ensuring that the length of the paths are minimum, similar to the MSTL + Hops methodology.

Algorithmus 5.5 Tabu Search: Selecting Flows

```
1: $MSTL \leftarrow max(load)$
2: $MaxLoadLink \leftarrow edge$ with $MSTL$
3: $LoadHistory \leftarrow (MSTL,\ MaxLoadLink)$
4: $routedFlows \leftarrow flowsRoutedOnLink[MaxLoadLink]$
5: sort $routedFlows$ by $length$
6: return $routedFlows, LoadHistory$
```

Algorithmus 5.6 Tabu Search: Re-Route Flow

```
1: remove $MaxLoadLink$ from $topology$
2: $route[flow] \leftarrow$ DIJKSTRAPATH($topology$, $src$, $dst$, $weight$)
3: add $MaxLoadLink$ to $topology$
4: TABU SEARCH: UPDATE LOAD
```

The algorithm maintains two data structures; $load$ to store the cumulative data bytes transmitted over each edge in the network and $flowsRoutedOnLink$ to store information about each flow transmitted over individual edges in the network as shown in Alg 5.4.

There are other possibilities for generating of an initial solution. This includes using standard routing algorithms like shortest path first (SPF) or equal cost multi path (ECMP) to route the flows in the given flow set. The MSTL of the initial solution generated by this approach would be very high. This violates the condition of minimizing the MSTL value. Another approach is to use algorithms like weighted SPF or weighted ECMP for routing. As shown in previous sections, they optimize the MSTL value by routing over longer paths. This is violates the condition of finding the shortest path possible for each flow. Due to violation of the conditions neither of the above mentioned approach is feasible to generate an initial solution.

5.4.2 Neighborhood Search

After generating the first feasible solution the local search algorithm begins its search for the neighboring solutions with a better objective function value. A neighboring solution to the current solution entails a set of routes for flows in the given flow set such that the $MaxLoadLink$ is different from that of the current solution. The objective function is the minimization of the MSTL value.

The first step is to identify flows to re-route. As shown in Alg 5.5 the edge with the maximum load is identified ($MaxLoadLink$). The data structure $flowsRoutedOnLink$ is used to extract information about the flows routed over the $MaxLoadLink$. These flows are sorted according to their data length and re-routed. The algorithm begins

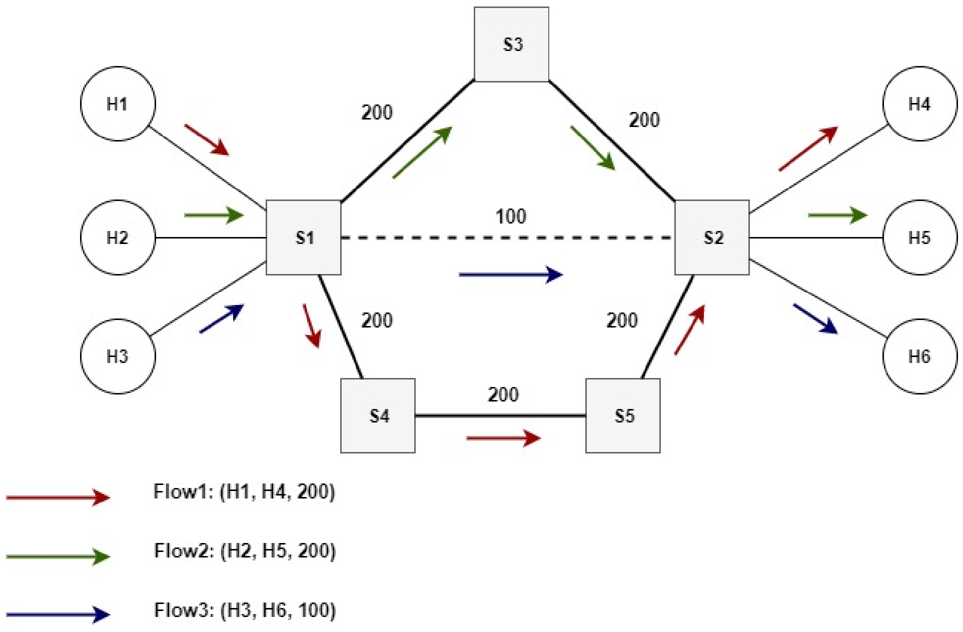


Figure 5.8: Neighborhood Search: re-routing flows on $MaxLoadLink$ to reduce MSTL (Tabu Search Algorithm)

re-routing with the flow with the highest data length because it will have the maximum impact on MSTL value.

The re-routing of flows is done using Dijkstra algorithm on a weighted graph. The $MaxLoadLink$ is temporarily removed from the topology to refrain routing of flows over the link. If flows are allowed to be routed over the $MaxLoadLink$ the MSTL value might not decrease. On the contrary it might contribute to an increase in MSTL value in some cases. To avoid this scenario the $MaxLoadLink$ is not taken into consideration while re-routing. The Dijkstra algorithm on a weighted graph ensures distribution of the data load over the entire network by routing the flows over routes with lower cumulative load. This occasionally results in routing of the flow over a longer route. The $MaxLoadLink$ is added back to the topology after routing and the data structures $load$ and $flowsRoutedOnLink$ are updated.

The aim of the algorithm is to re-route flows of the $MaxLoadLink$ such that the cumulative data load over the link is reduced by distributing it across the network, thus reducing the overall MSTL value. This is shown using the following example, as shown in Fig 5.7 Link$_{(S1,S2)}$ is the $MaxLoadLink$. The flows routed on this link are Flow1 and Flow3, of which Flow1 has the highest data length. The algorithm begins re-routing with Flow1. As shown in Fig 5.8, the Link$_{(S1,S2)}$ is removed from the topology and Flow1 is re-routed over Link$_{(S1,S4)}$, Link$_{(S4,S5)}$ and Link$_{(S5,S2)}$ because this is the route with the lowest cumulative data load. Flow1 is routed over a relatively longer path, but the MSTL value has reduced from 300 bytes to 200 bytes. Thus, the algorithm is successful in finding a neighboring solution where the $MaxLoadLink$ has changed and the MSTL value is lowered.

5.4.3 Tabu List

The search for neighbor solution using local search algorithm might result in getting stuck at local optimum. To encourage the algorithm to search for the global optimum the algorithm should not be allowed to visit already visited solutions and should be guided towards unexplored regions. This is achieved using Tabu List. Tabu as the name suggest is prohibiting the algorithm from cycling back to an already visited solution. The Tabu list stores the previously visited solutions or moves leading to the an already visited solution. During execution the algorithm refrains from executing the moves stored in the list. Thus, preventing it from cycling back in the same solution space. The Tabu list is a FIFO queue of fixed size. When the queue is full the insertion of a new element results in removal of the first element inserted to the queue.

Storing an entire solution in the Tabu list and comparing the current solution with the elements of the list can be very expensive. Therefore, we must determine the parameters that can be stored instead. The possible parameters for the routing problem are flows and links. In this section we discuss the implementation of Tabu list to enable a guided search for the optimal solution.

Tabu List: Flow Tabu

In the Alg 5.7 the Tabu list stores the flows that have been re-routed. Therefore, the Tabu list is referred to as $FlowTabu$. Before re-routing a flow the algorithm checks if the flow is present in $FlowTabu$. If present, the flow has been recently re-routed and re-routing of this flow might lead it to an already visited solution. To avoid this the algorithm refrains from re-routing the flows stored in the list. If the flow is not present in the $FlowTabu$ the algorithm re-routes the flow and adds it to the list.

The algorithm terminates the for-loop when the current $MaxLoadLink$ is no longer the edge with the highest cumulative data load transmitting over it. The algorithm evaluates if the current solution is the best solution found using Alg 5.8. If the MSTL of the current solution is lower than the lowest MSTL ($lowestMSTL$) found so far, the data structure $bestSolution$ is updated with the current solution ($route$) and the $lowestMSTL$ is assigned the value of the MSTL value of the current solution.

Algorithmus 5.7 Tabu Search: Tabu Flow

```
1: $tabuLen := len(F) * listSize$
2: $FlowTabu \leftarrow deque(tabuLen)$
3: $LoadHistory \leftarrow [\,]$
4: $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: INITIAL SOLUTION
5: while True do
6:     $routedFlows, LoadHistory \leftarrow$ TABU SEARCH: SELECTING FLOWS
7:     for all $flow$ in $routedFlows$ do
8:         if $flow$ in $FlowTabu$ then
9:             continue
10:        else
11:            $FlowTabu \leftarrow flow$
12:            TABU SEARCH: RE-ROUTE FLOW
13:            $NewMaxLoadLink \leftarrow edge$ with $max(load)$
14:            if $NewMaxLoadLink == MaxLoadLink$ then
15:                continue
16:            end if
17:        end if
18:    end for
19:    $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: BEST SOLUTION UPDATE($lowestMSTL$)
20:    TABU SEARCH: TERMINATION CRITERIA($LoadHistory$)
21: end while
22: return $bestSolution$
```

Algorithmus 5.8 Tabu Search: Best Solution Update

```
1: $newMSTL \leftarrow max(load)$
2: if $newMSTL < lowestMSTL$ then
3:     $bestSolution \leftarrow route$
4:     $lowestMSTL \leftarrow newMSTL$
5: end if
6: return $lowestMSTL, bestSolution$
```

Algorithmus 5.9 Tabu Search: Tabu Link

```
1: $tabuLen := len(E) * listSize$
2: $LinkTabu \leftarrow deque(tabuLen)$
3: $LoadHistory \leftarrow [\,]$
4: $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: INITIAL SOLUTION
5: while True do
6:     $routedFlows, LoadHistory \leftarrow$ TABU SEARCH: SELECTING FLOWS
7:     for all $flow$ in $routedFlows$ do
8:         remove $edges$ in $LinkTabu$ from $topology$
9:         TABU SEARCH: RE-ROUTE FLOW
10:        $NewMaxLoadLink \leftarrow edge$ with $max(load)$
11:        if $NewMaxLoadLink == MaxLoadLink$ then
12:            continue
13:        else
14:            add $edges$ in $LinkTabu$ to $topology$
15:            $LinkTabu \leftarrow NewMaxLoadLink$
16:        end if
17:    end for
18:    $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: BEST SOLUTION UPDATE($lowestMSTL$)
19:    TABU SEARCH: TERMINATION CRITERIA($LoadHistory$)
20: end while
21: return $bestSolution$
```

An important parameter to evaluate the impact of $FlowTabu$ in search for an optimal solution is the length of the Tabu list. The length of the list is a linear function of the number of flows routed in the network. In chapter 6 the evaluations establishing the relation between the length of the Tabu list and the MSTL value are presented. Also discussed are the evaluations of Tabu Search + $FlowTabu$ algorithm.

Tabu List: Link Tabu

In the Alg 5.9 instead of restricting the flows that can be re-routed, the algorithm restricts the links over which routing is possible. The tabu list stores the links whose selective flows have been re-routed. Therefore, it is referred to as $LinkTabu$. Before re-routing the flows, all the links present in the $LinkTabu$ are removed from the topology. Routing of flows over these links might lead to an already visited solution. To avoid this the algorithm refrains from routing over the links present in $LinkTabu$.

The algorithm terminates the for loop when the current $MaxLoadLink$ is no longer the edge with the highest cumulative data load transmitting over it and it adds this edge

Algorithmus 5.10 Tabu Search: Tabu Link Flow

```
1: $flowTabuLen := len(F) * flowListSize$
2: $linkTabuLen := len(E) * linkListSize$
3: $FlowTabu \leftarrow deque(flowTabuLen)$
4: $LinkTabu \leftarrow deque(linkTabuLen)$
5: $LoadHistory \leftarrow [\,]$
6: $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: INITIAL SOLUTION
7: while True do
8:     $routedFlows, LoadHistory \leftarrow$ TABU SEARCH: SELECTING FLOWS
9:     for all $flow$ in $routedFlows$ do
10:        if $flow$ in $FlowTabu$ then
11:            continue
12:        else
13:            $FlowTabu \leftarrow flow$
14:            remove $edges$ in $LinkTabu$ from $topology$
15:            TABU SEARCH: RE-ROUTE FLOW
16:            $NewMaxLoadLink \leftarrow edge$ with $max(load)$
17:            if $NewMaxLoadLink == MaxLoadLink$ then
18:                continue
19:            else
20:                add $edges$ in $LinkTabu$ to $topology$
21:                $LinkTabu \leftarrow NewMaxLoadLink$
22:            end if
23:        end if
24:    end for
25:    $lowestMSTL, bestSolution \leftarrow$ TABU SEARCH: BEST SOLUTION UPDATE($lowestMSTL$)
26:    TABU SEARCH: TERMINATION CRITERIA($LoadHistory$)
27: end while
28: return $bestSolution$
```

to $LinkTabu$. The algorithm evaluates if the current solution is the best solution found using Alg 5.8.

An important parameter to evaluate the impact of $LinkTabu$ in search for an optimal solution is the length of the tabu list. The length of the list is a linear function of the number of edges in the network. In chapter 6 the evaluations establishing the relation between the length of the tabu list and the MSTL value are presented. Also discussed are the evaluations of Tabu Search + $LinkTabu$ algorithm.

Algorithmus 5.11 Tabu Search: Termination Criteria

```
1: $tuples \leftarrow LoadHistory\ last\ two\ consecutive\ elements$
2: if occurrence of $tuples$ in $LoadHistory > 2$ then
3:     break while loop
4: end if
```

Tabu List: Link and Flow Tabu

In the Alg 5.10 we restrict the flows that can be re-routed as well as the links over which routing is possible. The algorithm has two tabu lists: $FlowTabu$ and $LinkTabu$. $FlowTabu$ stores the flows that have been re-routed and $LinkTabu$ stores the links whose flows have been re-routed.

The flow to be re-routed is checked for in the $FlowTabu$. If not present, first all the links present in the $LinkTabu$ are removed from the topology and then the flow is re-routed. This flow is then added to $FlowTabu$. Routing of flows in $FlowTabu$ or routing over links in $LinkTabu$ might lead to an already visited solution.

The algorithm terminates the for loop when the current $MaxLoadLink$ is no longer the edge with the highest cumulative data load transmitting over it and it adds this edge to $LinkTabu$. The algorithm evaluates if the current solution is the best solution found using Alg 5.8.

In chapter 6 the evaluations establishing the relation between the length of $LinkTabu$ and $FlowTabu$, and the MSTL value are presented. Also discussed are the evaluations of Tabu Search + $LinkTabu$ + $FlowTabu$ algorithm.

5.4.4 Termination Criteria

The search for a feasible neighbor solution is continued until the termination criteria is met. The algorithm may be terminated after a fixed number of iterations or when the objective reaches a specified threshold value. We avoid using either of the above mentioned approaches since they would limit the scope of the search. Instead we propose terminating the iterations when the algorithm observes recurring objective function values.

The data structure $LoadHistory$ stores the MSTL value and $MaxLoadLink$ of the network as a tuple. At the end of each iteration the algorithm evaluates the number of times the last two tuples in $LoadHistory$ have occurred consecutively. If this count is greater than two then the algorithm is oscillating in the same solution space. Hence we terminate the search and return the best solution found so far.

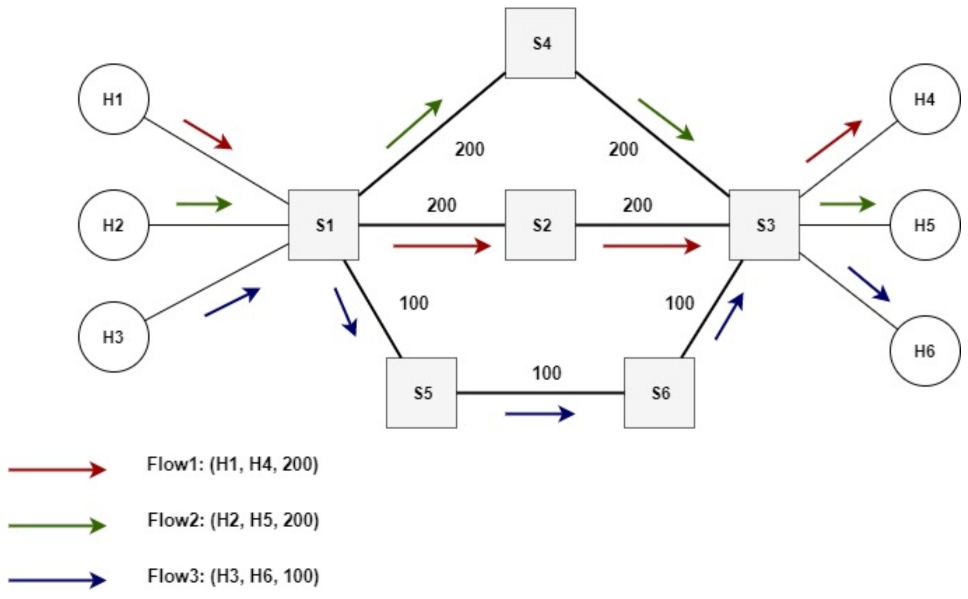


Figure 5.9: Routing of flows in set F_1

5.5 Dynamic Routing Using Tabu Search

The routing and scheduling of time triggered flows discussed so far was for a fixed set of flows. In this section we try to understand the behavior of Tabu Search algorithm in situations where new set of flows are introduced in the network post successful scheduling of the earlier set. This problem can be solved using three different methods. Each method is discussed and analyzed with the help of the following example. Given a topology as shown in Fig 5.9 and two sets of flows. F_1 includes Flow1, Flow2 and Flow3 and F_2 includes Flow4 and Flow5. F_1 has already been scheduled, post which F_2 is introduced into the network for scheduling.

METHOD 1: The first method to solve the scheduling problem when new flows are introduced into the network is to discard the previous transmission schedule, and schedule the sets F_1 and F_2 together. While this is the simplest approach and is more likely to deliver an optimal solution to the problem, it is expensive in terms of time. Also in certain scenarios where transmission of F_1 flows have already begun, this approach is infeasible.

METHOD 2: The second method to schedule the new set of flows is to view it as an independent scheduling problem. The previous flow set scheduled on the network is not taken into consideration. This approach aims at generating an optimal solution only for flows in F_2. The drawback of this approach is that the second schedule cannot begin until the first schedule has been completed. This is due to the fact that the second schedule has been developed in isolation from first schedule, it holds no knowledge

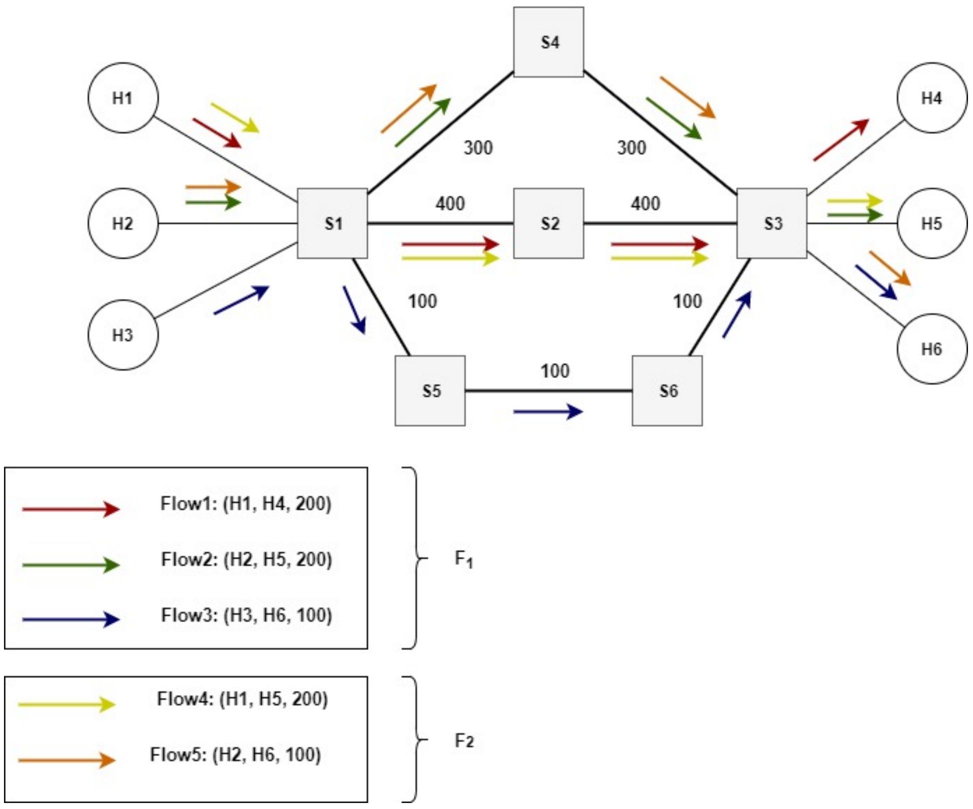


Figure 5.10: Initial Solution using Tabu Search Algorithm

of its transmission operations and thus cannot accommodate its own operations such that it saves on time. Certain scenarios present opportunity to overlap the two schedule in time domain, reducing the completion time of the second schedule. However, this approach is unable to support this merge.

METHOD 3: In third method the solution of the flow set F_1 is taken into consideration while generating a schedule for flow set F_2. It is imperative no changes are made to the schedule of F_1. The aim of this method is to route the flows in F_2 with respect to the already scheduled flows of F_1 such that the new MSTL value is minimal.

The tabu search algorithms first generates an initial solution. It uses the same topology as Fig 5.9 including the load on its edges reflecting the best solution found. The algorithm routes the flows in F_2 using Alg 5.3. The route selected for Flow4 and Flow5 is shown in Fig 5.10 and the resulting MSTL is equal to 400 bytes.

Tabu search begins the search for feasible neighboring solution with lower MSTL value using Alg 5.5. Only the flows in F_2 will be re-routed. A feasible neighboring solution is shown in Fig 5.11 whose resulting MSTL is equal to 300 bytes, lower than the initial solution.

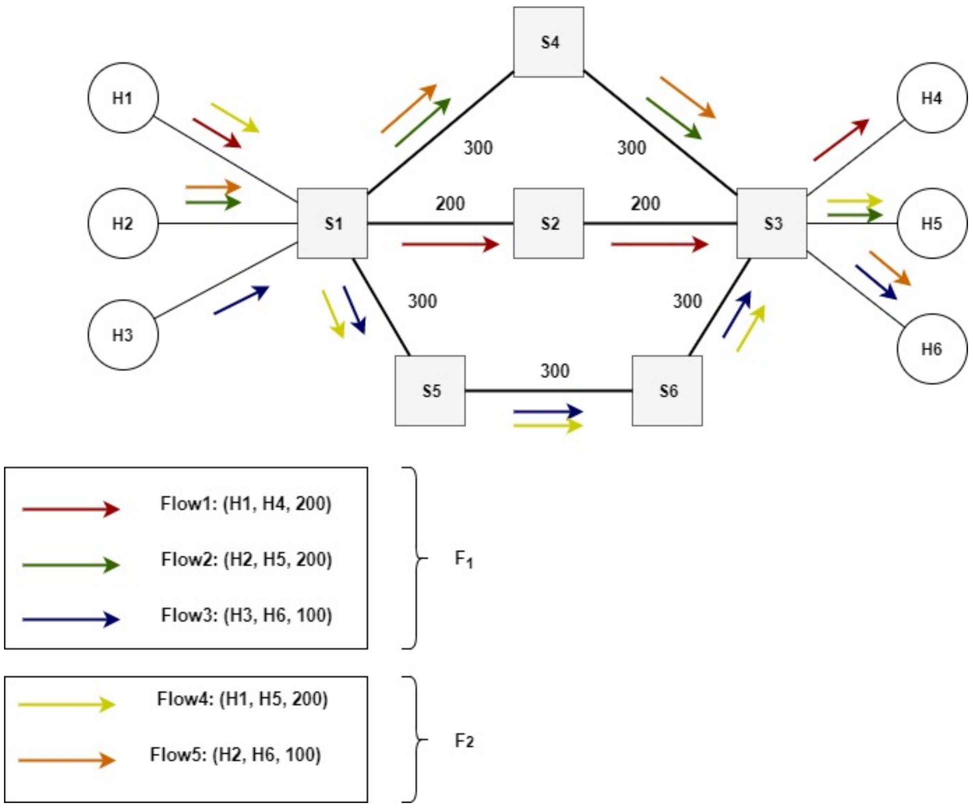


Figure 5.11: Neighboring solution using Tabu Search Algorithm

The algorithm ends when the termination criteria is met. However, in case of dynamic routing another termination criteria can be considered. If the new MSTL value equals the old MSTL value, it is the lowest possible value it can achieve. Therefore, the algorithm can be terminated under such conditions.

The advantage of this method is that it allows us to generate a new schedule without having to re-schedule the previous set of flows as in method 1. Also it does not require to wait for the completion of previous schedule as seen in method 2. However, it is possible the solution generated by this method is not optimal, a better solution might be found if flows from F_1 and F_2 were routed together.

It must be noted that the routing algorithms discussed in this chapter do not consider the order in which the flows are routed. Evaluations have shown that ordering of flows based on data length, source of the flow and destination of the flow and routing has an impact on the MSTL value. However, in this thesis we have not analyzed this aspect, instead we propose it as future work.

6 Results and Evaluations

In this section, we shall present the results of the evaluations done for the routing algorithms introduced in Chapter 5. The following evaluations were performed: the impact of the routing algorithms on the MSTL value of the network, the impact of the reduced MSTL on the quality of the transmission schedules and the scalability of the discussed routing algorithms. The evaluations for the quality of schedule has been done for the two scheduling methods discussed in section 2.2; NW-PSP based Method and SMT based method.

6.1 Impact of Routing Algorithms on MSTL

As discussed in Chapter 5, the routing algorithms used to route time triggered flows influence the MSTL value by distributing the data load over the entire network. In this section we evaluate and analyze the impact of the discussed routing algorithms on the MSTL value with respect to number of flows transmitted in the network and the graph connectivity.

The algorithms evaluated are shortest path first (SPF), weighted shortest path first (wt-SPF), weighted ECMP (wtECMP), ILP based algorithms; Load Aware algorithm (LA) and Load+Hop Aware algorithm (LHA) refer section 2.4 and Tabu search based algorithms. The Tabu search based algorithms include TabuFlow, TabuLink and TabuLinkFlow refer section 5.4.

6.1.1 MSTL vs No. of Flows

Analyzing the MSTL value with respect to the number of flows transmitted through the network helps us assess how well the routing algorithm distributes the data load across the network. A lower MSTL value implies that flows that would otherwise have been routed over the bottleneck link have been routed over different routes to ensure minimum contention between flows on the link. This probably will result in generation of a better quality schedule as discussed in section 2.4.

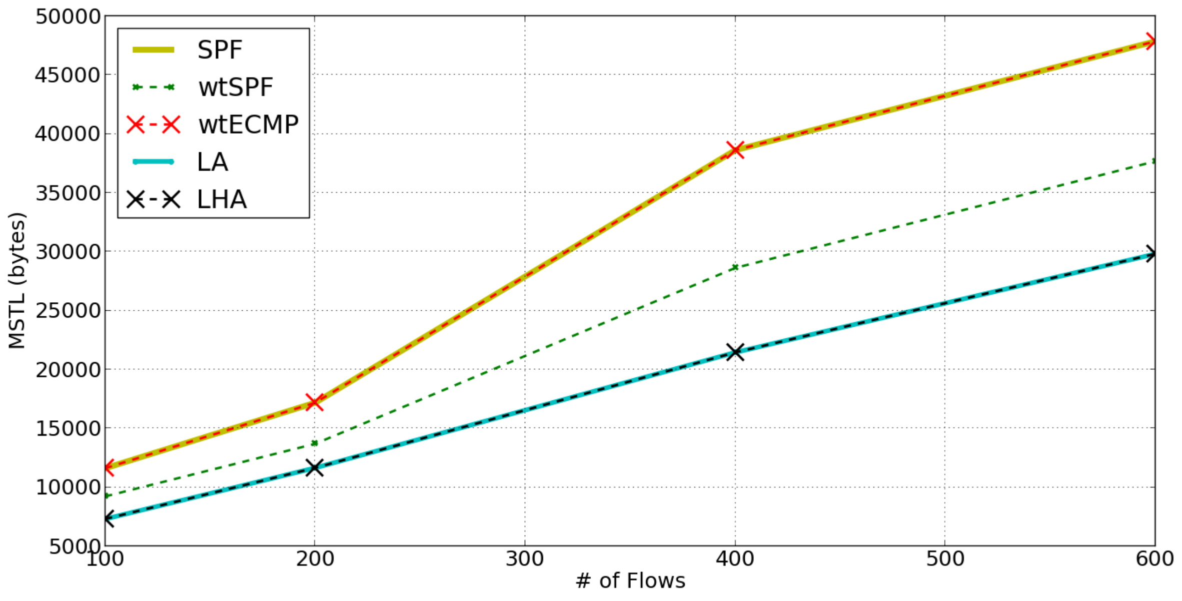
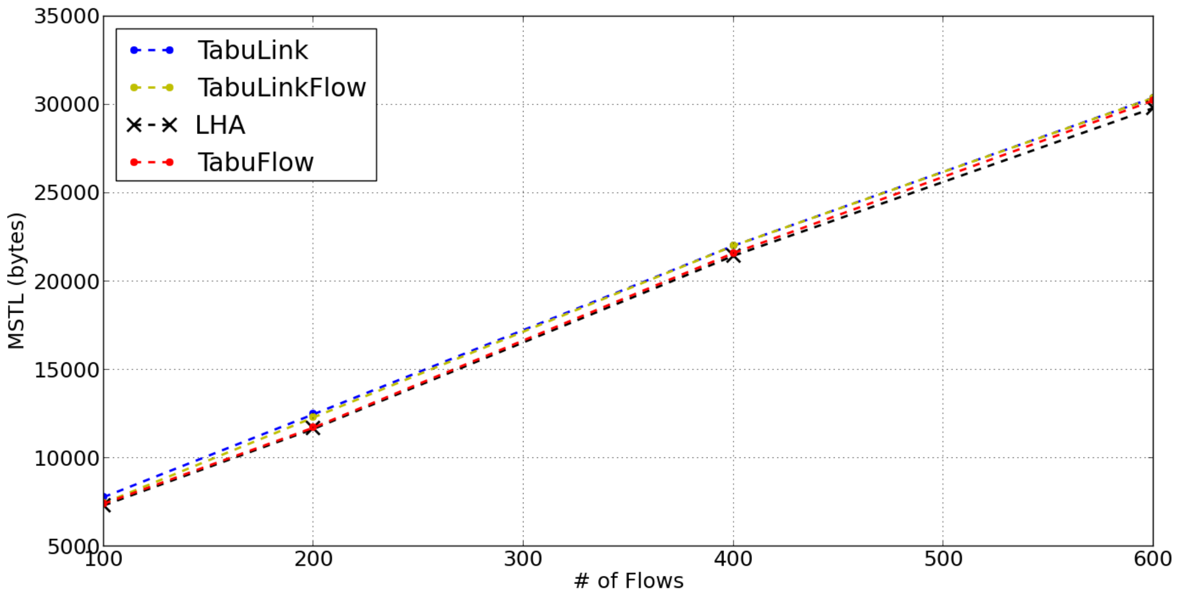


Figure 6.1: MSTL vs Number of Flow

Evaluations of the routing algorithms mentioned above with respect to the resulting MSTL for transmitting a certain number of flows has been shown in Fig 6.1. Shortest path first (SPF) which is a standard routing algorithm returns a very high MSTL value, this implies that the algorithm concentrates the transmission of flows over selected few routes, consequently increasing the contention between these flows. The ILP based algorithms; Load Aware (LA) and Load Hop Aware (LHA) have the lowest MSTL values. They successfully distribute the data load over the network and as a result decrease the contention between flows over a link. The difference in the MSTL values increases with an increase in the data load on the network. This indicates that use of shortest path first (SPF) algorithm would not be feasible in scenarios requiring transmission of large amount of data.

Evaluations of Tabu search based algorithms are potted separately in Fig 6.2. This is done to emphasis the difference in resulting MSTL, since the difference in the values are too low to be displayed clearly in the previous figure. In Fig 6.1 we have plotted the results of TabuFlow algorithm exclusively to demonstrate that it's MSTL values are very close to the values returned by ILP based algorithms. In Fig 6.2 we observe that the difference in MSTL values between Tabu search based algorithms and ILP based algorithms, the Tabu search based algorithms' MSTL are higher by 1.3% to 1.7%. Comparison between the Tabu search based algorithms reveal that TabuFlow algorithm returns better results. However, the difference in MSTL values is not stark. Further evaluations are required to determine which algorithm fares better.



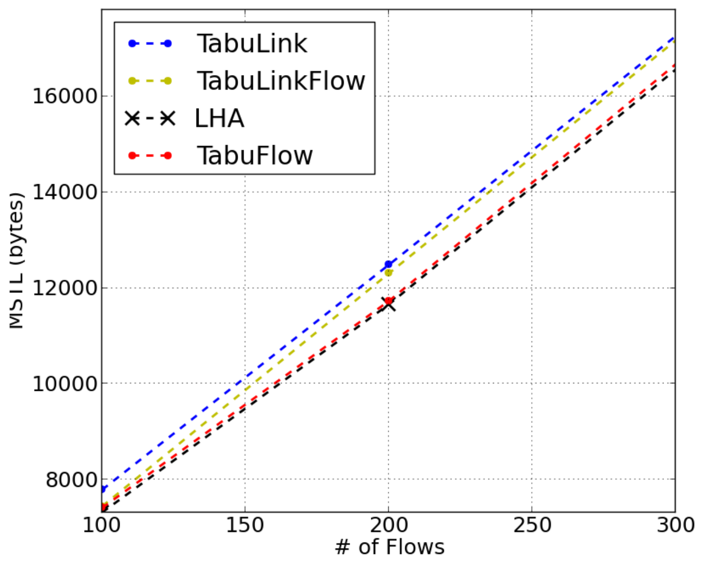


Figure 6.2: MSTL vs Number of Flows (Tabu Search)

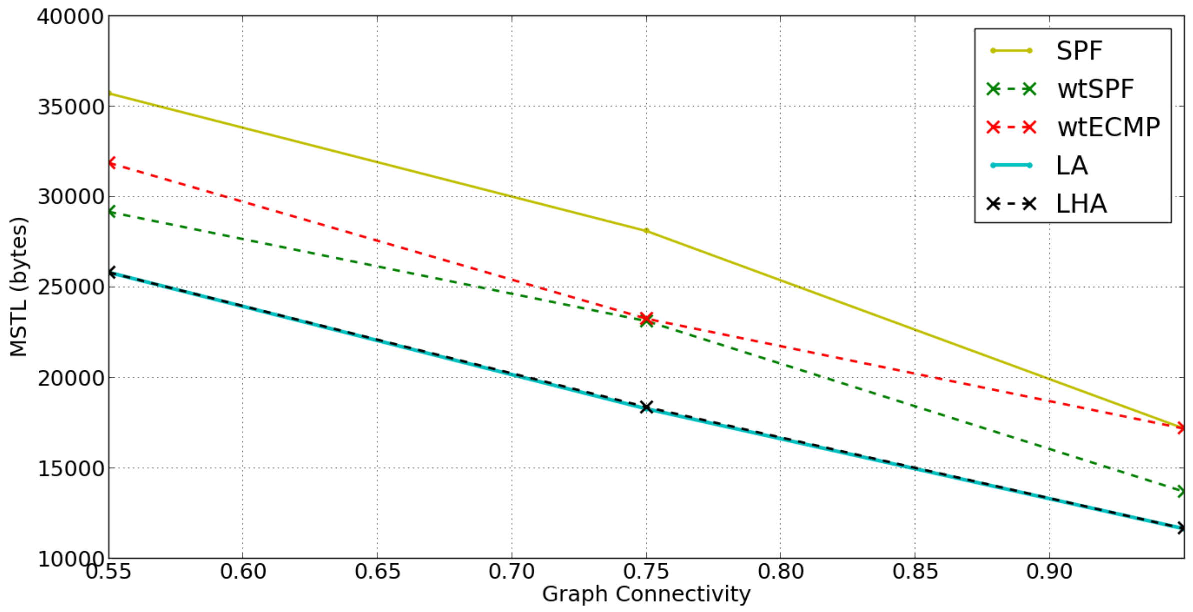


Figure 6.3: MSTL vs Graph Connectivity

6.1.2 MSTL vs Graph Connectivity

For our evaluations we use Erdős–Rényi model to generate the network topology. The parameters taken by the model to generate the topology include; number of nodes (n), probability for edge creation (p), seed for random number generator (seed). We refer parameter probability for edge creation (p) of the network as graph connectivity. It indicates the probability of a connection existing between a pair of nodes. Higher the value of graph connectivity, higher is the number of routes of varying lengths between a pair of nodes over which data can be transmitted. To understand the dependency of the routing algorithms on the number of routes available for transmission, we evaluate the algorithms on networks with varying graph connectivity.

Evaluations of the resulting MSTL on network topologies with varying graph connectivity was done by routing 100 flows of data length varying between 300-1500 bytes. The values have been plotted in Fig 6.3. Shortest path first (SPF) has the highest MSTL across all network topologies. The lowest MSTL values are returned by ILP based algorithms; Load Aware (LA) and Load Hop Aware (LHA). It is observed that the MSTL decreases with increase in graph connectivity, this is due to an increase in number of routes over which data can be transmitted.

As shown in Fig 6.3, the MSTL values of TabuFlow algorithm are comparable to the values returned by ILP based algorithms. However, it must be noted the difference in values decreases with increase in connectivity. This indicates the TabuFlow algorithm's

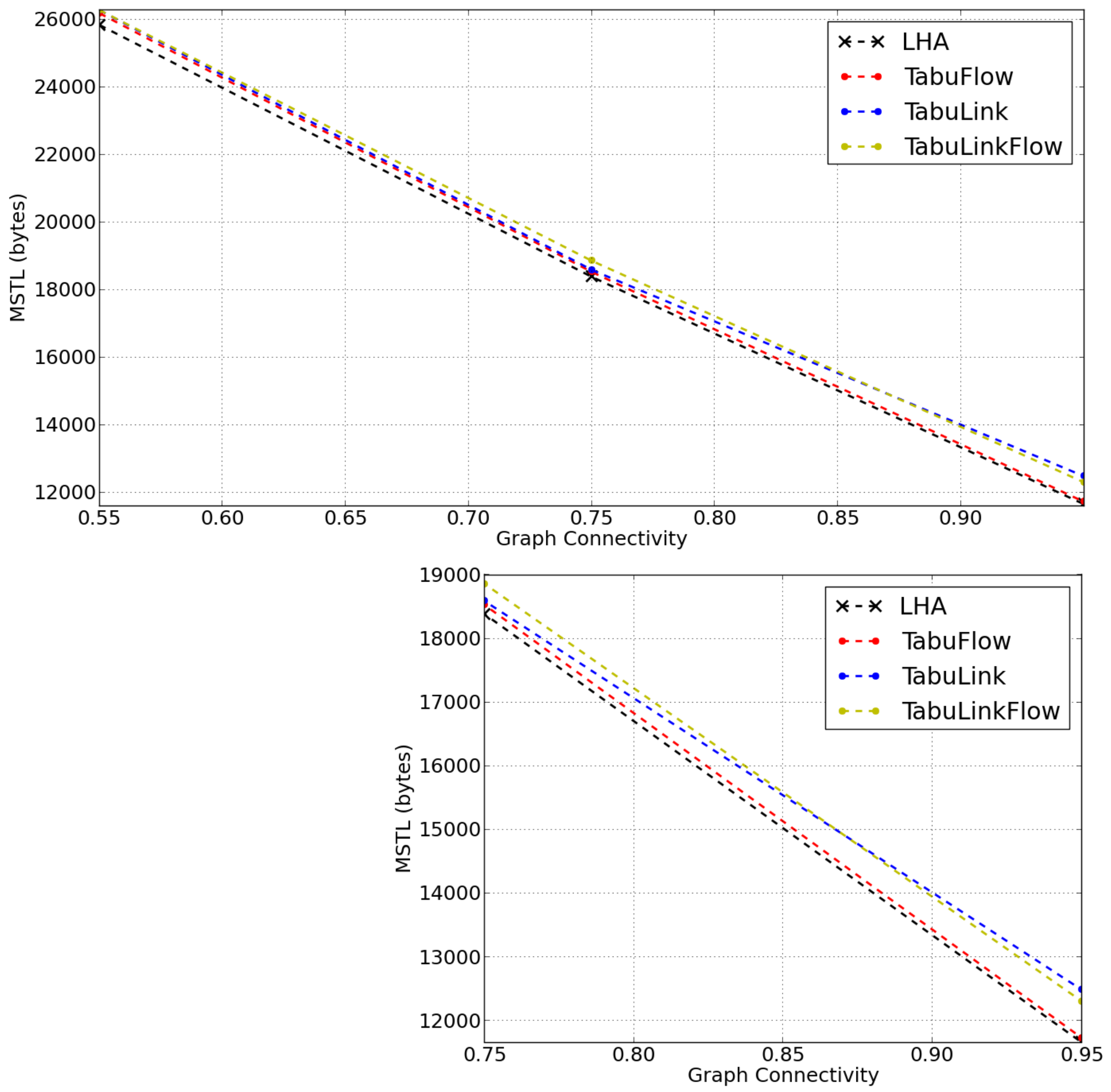


Figure 6.4: MSTL vs Graph Connectivity (Tabu Search)

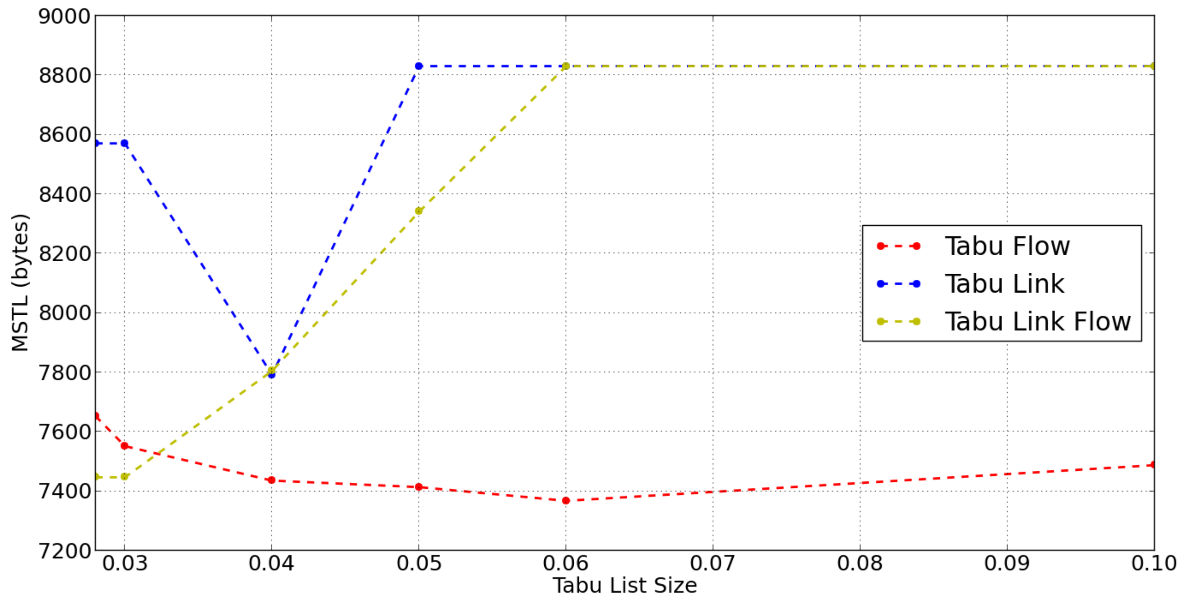


Figure 6.5: MSTL vs Tabu List Size

dependency on the number of available routes to perform as well as ILP based algorithms. In Fig 6.4 the results of Tabu search based algorithms are plotted separately to demonstrate the difference in returned MSTL values. The MSTL value of Tabu search based algorithms is higher by 1.7% to 12% in comparison to ILP based algorithms. It is observed that the TabuFlow algorithm fares better than the others in all the evaluated network topologies. However, the difference is observable only in a mesh network. For a topology with low connectivity the difference is too low to be considered.

6.1.3 Size of Tabu List

The Tabu List is a FIFO queue of fixed sized refer section 5.4.3. In TabuFlow algorithm the list size (Alg 5.6 Line 1) is a function of the number of flows routed in the network. In TabuLink algorithm the list size (Alg 5.8 Line 1) is a function of the number of links in the network. TabuLinkFlow algorithm has two tabu lists corresponding to the lists mentioned above; for flows ($FlowTabu$) and for links ($LinkTabu$). The size of the lists are altered using the variable $listSize$.

Evaluations for the impact of the list size on the MSTL value is shown in Fig 6.5. The TabuFlow algorithm returns best solution when the variable $listSize$ is equal to 0.06. The TabuLink algorithm returns best solution when $listSize$ is 0.03.

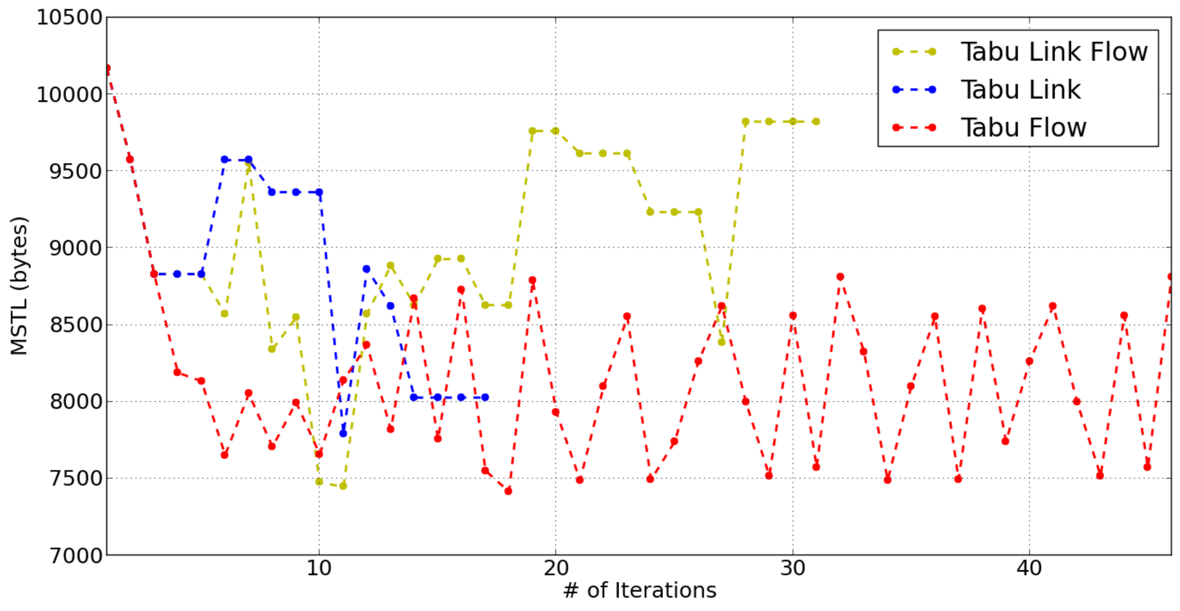


Figure 6.6: MSTL vs Number of Iterations

During previous evaluations we observed that the TabuFlow algorithm returns better solutions than TabuLink algorithm. Therefore, the $listSize$ for $FlowTabu$ in TabuLinkFlow was fixed as 0.06 and evaluations to determine the $listSize$ for $listSize$ were done. As shown in Fig 6.5 the value of 0.04 returns the best solution. These values have been used in further evaluations presented in the thesis.

6.1.4 Termination Criteria

In Section 5.4 we discussed the termination criteria for Tabu search based algorithms. We proposed terminating the algorithm when recurring values of MSTL are observed. To evaluate the termination criteria we routed 100 flows with data length varying between 300-1500 bytes in a mesh network. The resulting MSTL for each iteration of the Tabu search based algorithms are plotted in Fig 6.6.

It was observed that the TabuLink and TabuLinkFlow algorithm terminate after encountering the same MSTL value for four consecutive iterations. However, TabuFlow algorithm continues oscillating between values. The termination for TabuFlow algorithm is done when a set of already visited solutions are encountered more than two times. This behavior is consistent across multiple scenarios with varying number of flows.

6.2 Impact of Reduced MSTL on Quality of Schedule

6.2.1 Flowspan vs MSTL

Analyzing the flowspan with respect to the MSTL helps us assess the impact of load distribution on the flowspan of the schedule. A lower MSTL value implies that the contention between flows on a link is low; therefore a lower flowspan can be expected refer section 2.4.4. We demonstrate this relation of flowspan and MSTL using the example shown in Table 6.1. For the given example 100 flows of data length varying between 300-1500 bytes were scheduled over a mesh network.

Routing Alg.	MSTL (bytes)	Flowspan (Time units)
SPF	11623	96
wt SPF	9234	80
wt ECMP	11623	96
Load Aware	7316	78
Load Hop Aware	7320	70
Tabu Flow	7414	72
Tabu Link	7792	70
Tabu Link Flow	7447	73

Table 6.1: Impact of MSTL on Flowspan

Shortest path first (SPF) algorithm has the highest MSTL value, consequently the flowspan is equally high. Change in MSTL as seen for weighted shortest path first (wtSPF) algorithm lowers the flowspan as well.

The ILP based algorithms; Load Aware (LA) and Load Hop Aware (LHA) have the lowest MSTL and are therefore expected to have the lowest flowspan. However, it is observed that in spite of having a lower MSTL Load Aware (LA) algorithm's flowspan is higher than that of Load Hop Aware (LHA), this is because LHA considers the number of hops in a route while routing. Thus, establishing a direct relation between the number of hops in a route and the flowspan.

The flowspan of tabu search based algorithms are comparable to that of LHA's flowspan. This is due to two factors; low MSTL values and the initial solution generated by tabu

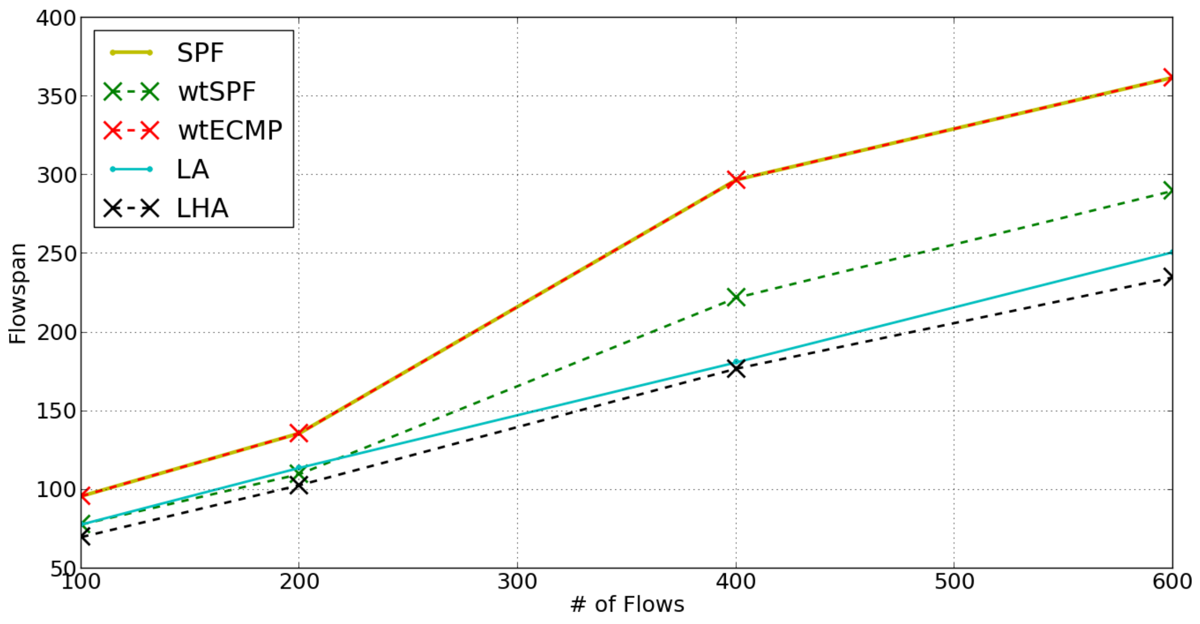


Figure 6.7: Flowspan vs Number of Flows

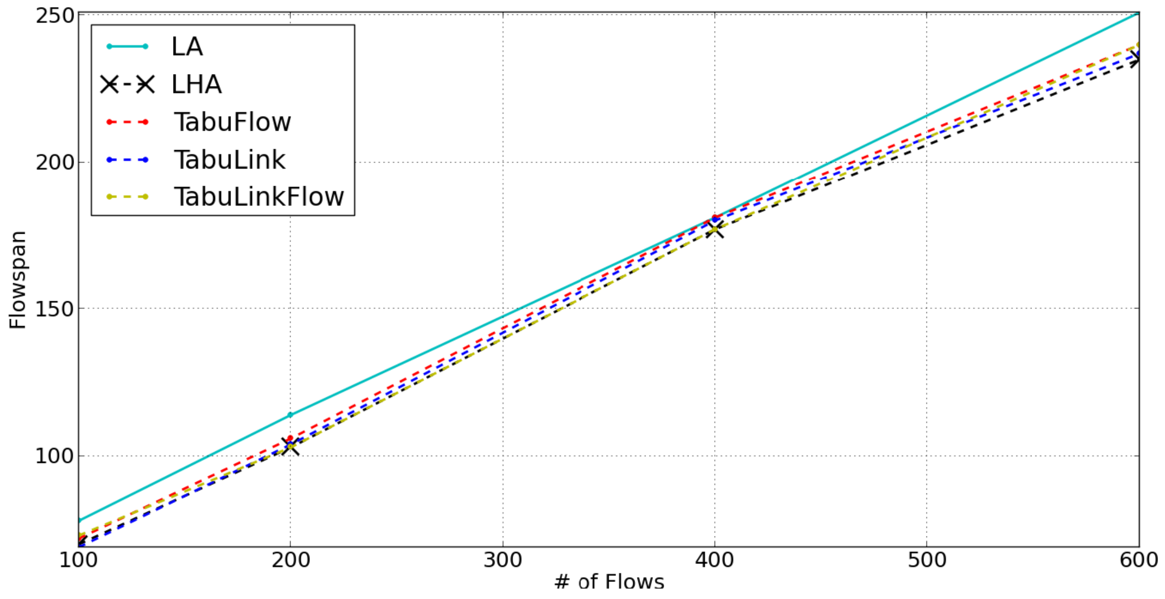
search based algorithms are conscious of number of hops in a route. TabuFlow has a lower MSTL compared to TabuLink, however their flowspan values are almost the same. This indicates that the routes selected while re-routing the flows in TabuFlow algorithm are longer, thus impacting its flowspan value.

6.2.2 Flowspan vs Number of Flows

Analyzing flowspan with respect to the number of flows transmitted through the network helps us assess the impact of increasing data load on the flowspan.

Shortest path first (SPF) algorithm has the highest flowspan. The value increases rapidly with increase in the data load on the network. Load hop aware (LHA) has the lowest flowspan, the increase with increase in data load is gradual.

Evaluations of Tabu search based algorithms are plotted separately in Fig 6.8 to emphasis on the difference in results. They perform better than Load Aware (LA) algorithm. However, the deviation of Tabu search based algorithms from Load Hop Aware (LHA) algorithm is higher by 2.85% to 4.3%. The difference in values between Tabu search based algorithms is too low to evaluate which performs better.



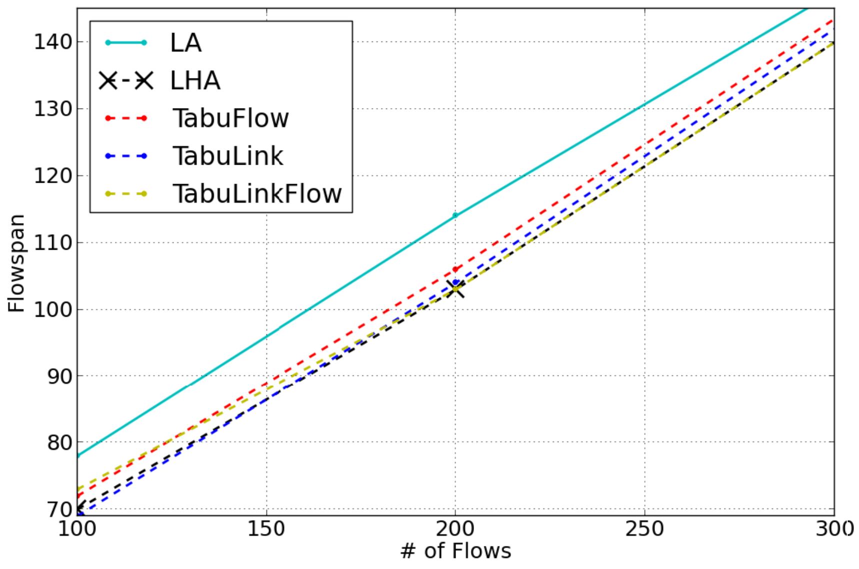


Figure 6.8: Flowspan vs Number of Flows (Tabu)

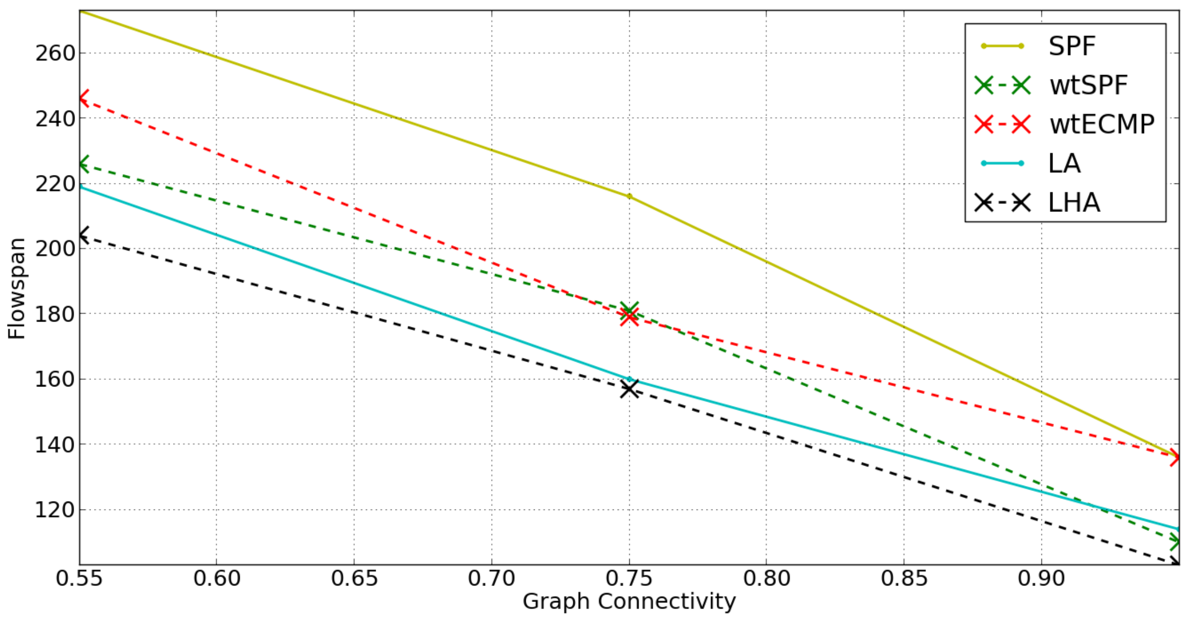


Figure 6.9: Flowspan vs Graph Connectivity

6.2.3 Flowspan vs Graph Connectivity

Evaluation of impact on the flowspan value due to varying graph connectivity is shown in Fig 6.9. The availability of alternate routes for transmission is bound to decrease the load on links, consequently reducing the flowspan. However, the length of these routes also influences the flowspan value. With this evaluation we understand the behavior of the routing algorithms in different scenarios.

Shortest path first (SPF) has the highest flowspan value across different topologies and the ILP based algorithm Load hop aware (LHA) has the lowest flowspan. This is due to the direct relation between MSTL and flowspan refer section 2.4.4. The curves of the two algorithms are same as that in Fig 6.1.

Evaluations of Tabu search based algorithms are plotted separately in Fig 6.10 to emphasis on the difference in results. In most topologies they perform better than Load Aware (LA) algorithm. However, the deviation of Tabu search based algorithms from Load Hop Aware (LHA) algorithm is higher by 0% to 6.4%. The performance of each Tabu search based algorithm varies in different topology, therefore neither can be assessed as a better option.

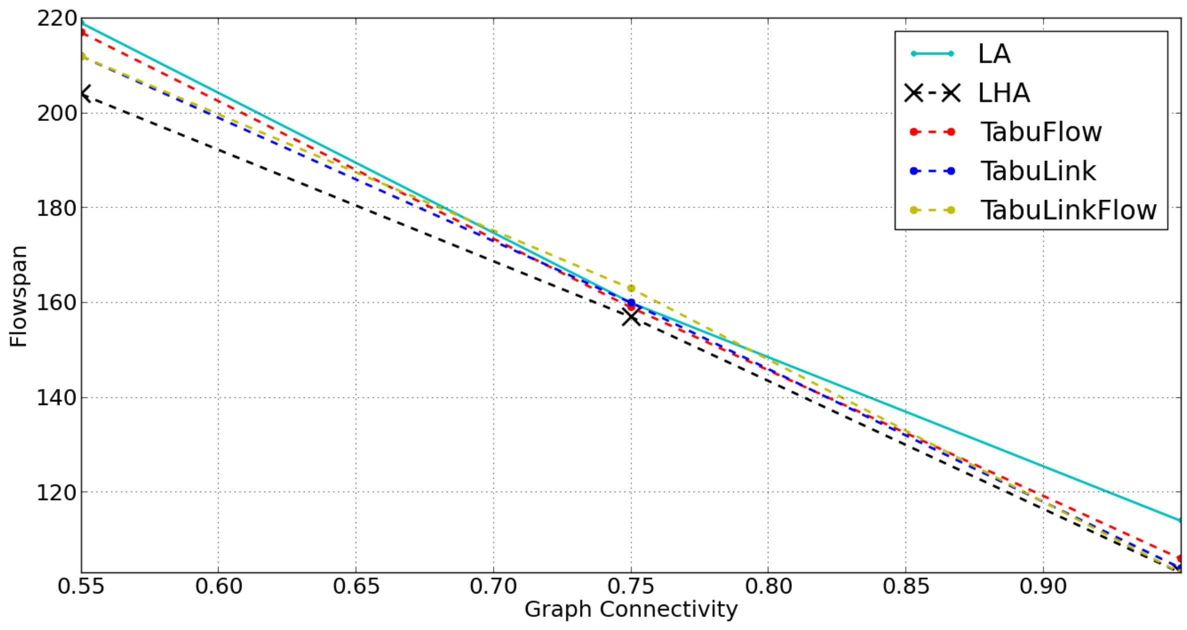


Figure 6.10: Flowspan vs Graph Connectivity (Tabu)

6.2.4 Breaking Point vs MSTL

The quality of a schedule generated by SMT method is analyzed using the metric "Breaking Point" refer section 4.2. In this section we establish the impact of MSTL value on the breaking point of a schedule.

We begin by assessing the MSTL returned by each routing algorithm when 50 flows of data length varying between 300-1500 bytes and end to end latency equal to 100 units of time were scheduled on a mesh network. The hyper-cycle of the schedule was equal to 200 units of time. The results are shown in Table 6.2.

Shortest path first (SPF) algorithm has the highest MSTL value and is not schedulable. This is because the transmission time of some flows exceeds the hyper-cycle refer section 4.2.

The other routing algorithms return routes which are schedulable in the given hyper-cycle. Of the discussed routing algorithms load hop aware (LHA) algorithm returns the lowest MSTL value. The MSTL value of Tabu search based algorithms is 16% to 20% higher than load hop aware (LHA) algorithm.

To assess the impact of these algorithms on the number of flows schedulable in a hyper-cycle of 100 units of time, we tried routing varying number of flows with data length

Routing Alg.	MSTL
SPF	87
wt SPF	31
wt ECMP	35
Load Aware	35
Load Hop Aware	24
Tabu Flow	28
Tabu Link	29
Tabu Link Flow	29

Table 6.2: MSTL for routing algorithms routing 50 flows

Routing Alg.	Breaking Point
SPF	20
wt SPF	80
wt ECMP	70
Load Aware	100
Load Hop Aware	110
Tabu Flow	50
Tabu Link	50
Tabu Link Flow	50

Table 6.3: Breaking Point of routing algorithms

between (300-1500) bytes across a mesh network. The results are presented in Table 6.3.

Shortest path first (SPF) scheduled the least number of flows, this is due to the high MSTL value returned by it. Load hop aware (LHA) algorithm schedule the highest number of flows, due to low MSTL.

Tabu search based routing algorithm in spite of having a low MSTL are unable to schedule more than 50 flows. This is due to routing over longer routes. The tabu search based algorithms do not consider number of hops while re-routing. Routing over longer routes may violate the end to end latency constraint. Even if a single flow's end to end latency constraint is violated a schedule would not be generated.

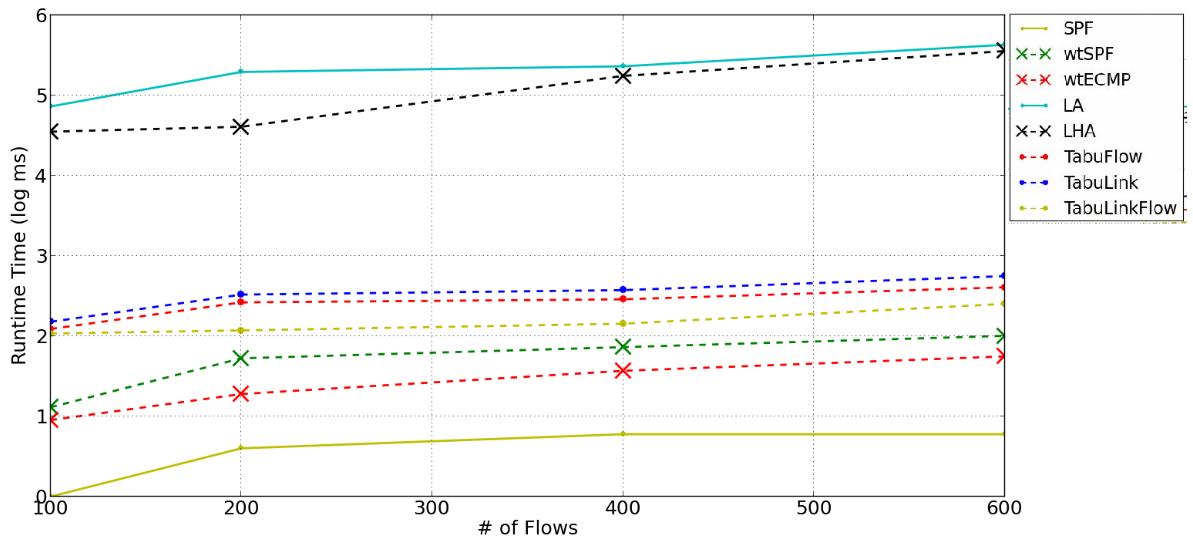


Figure 6.11: Routing Time vs Number of Flow

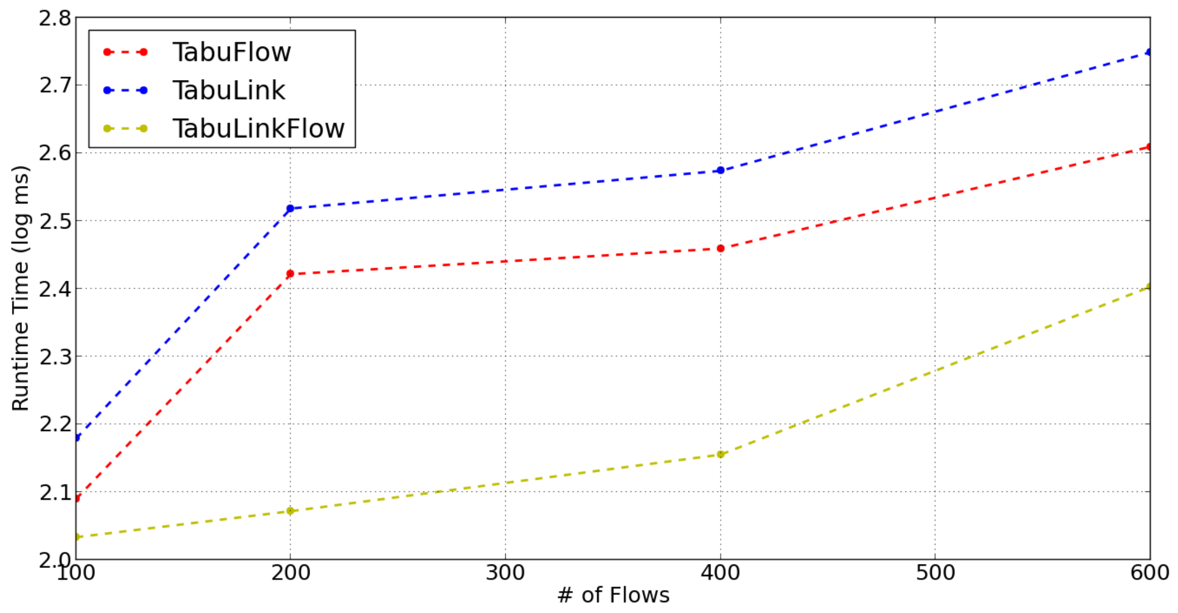


Figure 6.12: Routing Time vs Number of Flows (Tabu Search)

6.3 Scalability of Routing Algorithms

6.3.1 Routing Time vs No. of Flows

Analyzing the scalability of the routing algorithms with respect to the number of flows transmitted through the network helps us assess the runtime time of the algorithms with increase in data load across the network.

The routing time shown in Fig 6.11 is represented in log scale. The load hop aware algorithm's runtime increases exponentially with increase in data load. Therefore, use of the algorithm in scenarios with high data load is not feasible. The runtime of other algorithms in gradual. However, the difference in runtime time between the ILP based algorithms and Tabu search based algorithms is stark. The runtime of ILP based algorithms is higher by 47% to 64% in comparison to Tabu search based algorithms.

Evaluations of runtime of Tabu search based algorithms are plotted separately in Fig 6.12 to emphasis on the difference in results. TabuLinkFlow algorithms has the lowest runtime of the three algorithms and TabuLink the highest. These evaluations help assess which algorithm to select for routing.

6.3.2 Routing Time vs Graph Connectivity

Evaluations of runtime of the routing algorithms across topologies with varying graph connectivity is shown in Fig 6.13.

The results are similar to that of the previous evaluation. ILP based algorithms have the highest runtime. Due which their use in infeasible. The runtime of all algorithms increase with increase in connectivity, due to increase in number of routes to be evaluated for transmission of data. The runtime of ILP based algorithms is higher by 38% and 66% in comparison to the Tabu search based algorithms.

Evaluations of runtime of Tabu search based algorithms are plotted separately in Fig 6.14 to emphasis on the difference in results. TabuLinkFlow algorithms has the lowest runtime of the three algorithms and TabuLink the highest. This behavior is consistent with the previous evaluations shown in Fig 6.12.

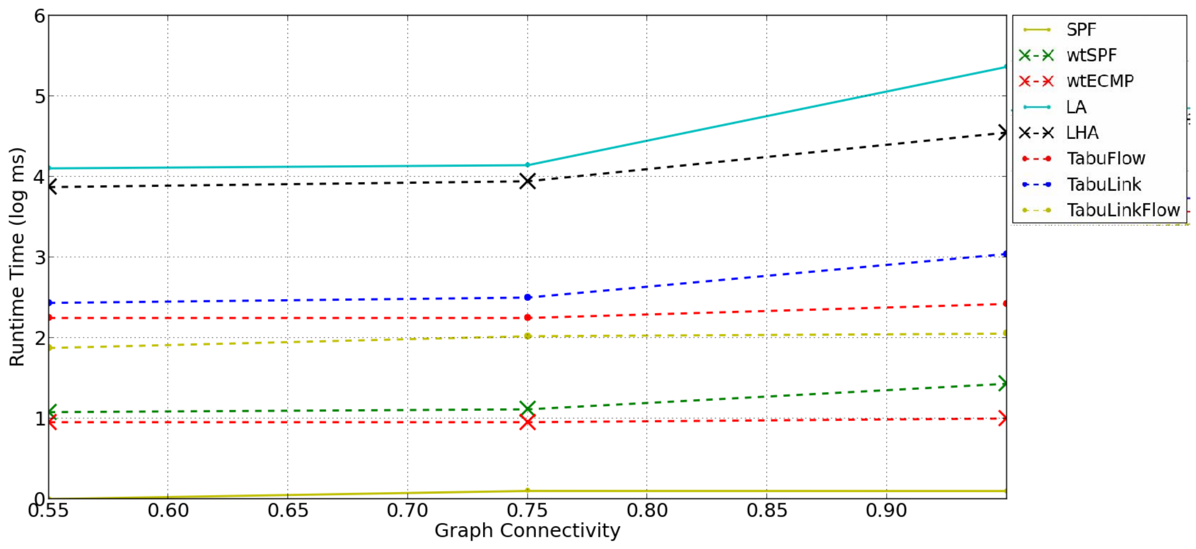


Figure 6.13: Routing Time vs Graph Connectivity

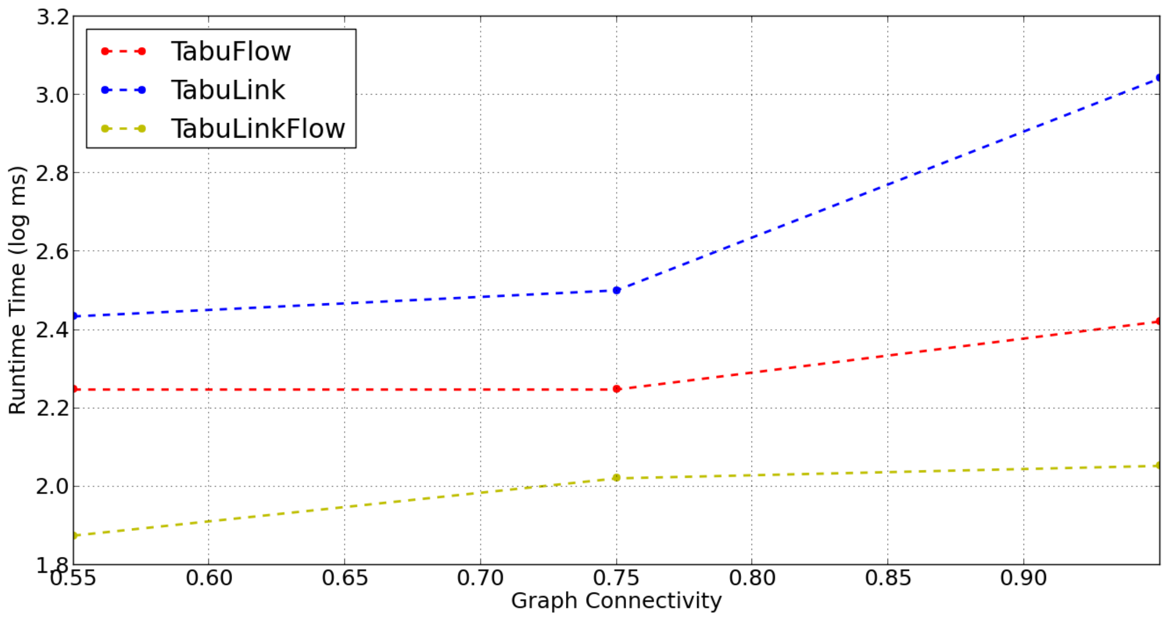


Figure 6.14: Routing Time vs Graph Connectivity (Tabu Search)

7 Conclusion

This thesis presented generic routing algorithms based on heuristic algorithm tabu search to route time triggered traffic in order to optimize the MSTL of the network. The MSTL is an indication of the distribution of data load across the network. It was attempted to benchmark MSTL as a metric that could be used by different scheduling methods to assess the impact of load distribution on the quality of transmission schedule generated by them. However, the primary aim was to prove that routing algorithms impact the quality of a transmission schedule irrespective of the scheduling method employed.

Evaluations done in the thesis have proven that MSTL can be used as a standard metric to assess the impact of load distribution on the quality of transmission schedule generated by different scheduling methods. The NW-PSP method's schedule quality is measured using the metric flowspan. A decrease in MSTL consequently decreased the flowspan of the schedule. This behavior was consistent across multiple scenarios. The SMT based method's schedule quality is measured using the metric Breaking Point, which indicated the number of flows scheduled in a single cycle of transmission schedule. As expected reduction in the MSTL increased the number of flows scheduled in a single cycle.

In this thesis three tabu search based routing algorithms were presented; TabuFlow, TabuLink and TabuLinkFlow. Evaluations of the algorithms indicate that the MSTL value returned by each is at most 1.7% higher than the optimal solution. This behavior is consistent across multiple scenarios; increasing data load on the network and varying connectivity of the topology. However, of the three algorithms the TabuFlow algorithm returns the lowest MSTL values. The scalability analysis of the algorithms show a reduction of 65% in runtime in comparison to the algorithms generating optimal solution. It can be concluded that the tabu search based algorithms are a feasible approach to solving routing problem in larger scenarios.

A drawback of the tabu search based algorithms is that during the search of a feasible neighboring solution they do not consider the number of hops in a route while re-routing the flows. A longer route occasionally violates the end to end latency constraint. The impact of which manifested in the schedule of the SMT based method. The number of flows scheduled in a single cycle was much lower than expected. The algorithms can be improved by making them aware of the number of hops in a route while routing the flows, and additionally ensuring that the end to end constraint is not violated.

There is always room for improvement and enhancement in any scientific work. This thesis being no exception the following suggestions are proposed. In the thesis while routing the given set of flows the order of flows was not considered. Evaluations suggest ordering the flows based on data length, source of the flows and destination of the flows impact the routing and consequently the MSTL. Hence it is suggested to take this parameter into consideration for future work and consider ordering of flows to further optimize the MSTL value.

The presented routing algorithms are based on tabu search algorithm, however other heuristic and meta heuristic algorithms are available such as A* algorithm, ant colony optimization, particle swarm optimization etc. It is suggested to consider these algorithms for developing routing algorithms aimed at optimizing the MSTL of the network.

Bibliography

[1] *802.1AS-2011 - IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. Mar. 2011. DOI: 10.1109/IEEESTD.2011.5741898 (cit. on p. 17).

[2] *802.1Qbv-2015 - IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*. Mar. 2016. DOI: 10.1109/IEEESTD.2016.7572858 (cit. on p. 18).

[3] *802.1Qca-2015 - IEEE Standard for Local and metropolitan area networks– Bridges and Bridged Networks - Amendment 24: Path Control and Reservation*. Mar. 2016. DOI: 10.1109/IEEESTD.2016.7434544 (cit. on p. 17).

[4] *802.3-2015 - IEEE Standard for Ethernet*. TTTech Computertechnik AG, Vienna, Austria, Mar. 2016. DOI: 10.1109/IEEESTD.2016.7428776 (cit. on p. 16).

[5] Cisco. *IS-IS Overview and Basic Configuration*. URL: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_isis/configuration/15-mt/irs-15-mt-book/irs-ovrw-cf.pdf (cit. on p. 31).

[6] S. S. Craciunas, R. S. Oliver, M. Chmelik, W. Steiner. *Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks*. ACM, 2016 (cit. on pp. 18, 27, 46).

[7] M. Dell'Amico, M. Trubian. "Applying tabu search to the job-shop scheduling problem." In: *Annals of Operations research* 41.3 (1993), pp. 231–252 (cit. on p. 34).

[8] Don Jacob. *A Quick Start to Path Computation Element (PCE)*. URL: http://www.packetdesign.com/blog/quick-start-path-computation-element-pce/ (cit. on p. 32).

[9] F. Duerr, N. G. Nayak. *Routing Algorithms for IEEE 802.1Qbv Networks*. 2017 (cit. on pp. 19, 33, 41, 43, 45).

[10] F. Dürr, N. G. Nayak. *No-wait packet scheduling for IEEE time-sensitive networks (TSN)*. ACM, 2016 (cit. on pp. 18, 26, 40, 45).

[11] R.-T. Ethernet. *SERCOS III: Proposal for a Publicly Available Specification for Real-Time Ethernet* (cit. on p. 16).

[12] R. Hummen. *What is TSN? A Look at Its Role in Future Ethernet Networks*. Feb. 2017. URL: http://www.belden.com/blog/industrialethernet/what-is-tsn-a-look-at-its-role-in-future-ethernet-networks.cfm (cit. on p. 15).

[13] "IEEE Standard for Local and metropolitan area networks– Bridges and Bridged Networks - Amendment 24: Path Control and Reservation." In: *IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015)* (Mar. 2016), pp. 1–120. DOI: 10.1109/IEEESTD.2016.7434544 (cit. on p. 30).

[14] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic." In: *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q— as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q—/Cor 1-2015)* (Mar. 2016), pp. 1–57. DOI: 10.1109/IEEESTD.2016.7572858 (cit. on pp. 22–24).

[15] D. Jansen, H. Buttner. "Real-time Ethernet: the EtherCAT solution." In: *Computing and Control Engineering* 15.1 (2004), pp. 16–21 (cit. on p. 16).

[16] N. Kokash. *An introduction to heuristic algorithms*. 2005 (cit. on p. 34).

[17] *Time-Sensitive Networking: A Technical Introduction*. 2017 (cit. on p. 16).

[18] E. Tovar, F. Vasques. *Real-time fieldbus communications using Profibus networks*. 1999 (cit. on p. 16).

[19] Wikipedia. *IEEE 802.1aq*. [Online; accessed 7-Oct-20017]. URL: https://en.wikipedia.org/wiki/IEEE_802.1aq (cit. on p. 32).

[20] W. Xia, Y. Wen, C. H. Foh, D. Niyato, H. Xie. "A survey on software-defined networking." In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 27–51 (cit. on p. 29).

All links were last followed on October 22, 2017.

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies

place, date, signature