UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Advanced Techniques for Multicast Service Provision in Core Transport Networks

Autor:
Gonzalo Fernández Del Carpio

Director:
Dr. David Larrabeiti López

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Leganés, octubre 2012

TESIS DOCTORAL

ADVANCED TECHNIQUES FOR MULTICAST SERVICE PROVISION IN CORE TRANSPORT NETWORKS

Autor: Gonzalo Fernández Del Carpio

Director: Dr. David Larrabeiti López

Firma del Tribunal Calificador:

Firma

Presidente: Dr. José Alberto Hernández Gutiérrez

Vocal: Dr. Xavier Masip Bruin

Secretario: Dr. Pedro Reviriego Vasallo

Calificación:

Leganés, 1 de octubre de 2012

"Sorprendernos por algo es el primer paso de la mente hacia el descubrimiento"

Louis Pasteur, 1822-1895

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Although the network-based multicast service is the optimal way to support of a large variety of popular applications such as high-definition television (HDTV), video-on-demand (VoD), virtual private LAN service (VPLS), grid computing, optical storage area networks (O-SAN), video conferencing, e-learning, massive multiplayer online role-playing games (MMORPG), networked virtual reality, etc., there are a number of technological and operational reasons that prevents a wider deployment. This PhD work addresses this problem in the context of core transport network, by proposing and analyzing new cost-effective and scalable techniques to support multicast both at the Optical layer and at the Network layer (MPLS-IP networks).

In the Optical layer, in particular in Wavelength Division Multiplexing (WDM) Optical Circuit Switched networks, current multicast-capable OXC node designs are of a great complexity and have high attenuation levels, mainly because of the required signal splitting operation plus the traversal of a complex switching stage. This makes multi-point support rarely included in commercial OXC nodes. Inspired in previous works in the literature, we propose a novel architecture that combines the best of splitting and tap-and-continue (TaC), called 2-STC (2-split-tap-and-continue) in the framework of integrated optics. A 2-STC OXC node is a flexible design capable of tapping and splitting over up to two outgoing links in order to obtain lower end-to-end latency than in TaC and an improved power budget distribution over split-and-delivery (SaD) designs. Another advantage of this architecture is its simplicity and the reduced number of components required, scaling well even for implementations of the node with many input/output ports. Extensive simulations show that the binary split (2-split) is quite enough for most real-life core network topologies scenarios, since the average node degree is usually between 3 and 4. A variant of this design, called 2-STCg, for making the node capable of optical traffic grooming (i.e. accommodation of low-speed demands into wavelength-links) is also presented.

At the Network layer, one of the main reasons that hinder multicast deployment is the high amount of forwarding state information required in core routers, especially when a large number of medium/small-sized multicast demands arrive to the core network, because the state data that needs to be kept at intermediate core routers grows proportionally to the number of multicast demands. In this scenario, we study the aggregation of multicast demands into shared distribution trees, providing a set of techniques to observe the trade-off between bandwidth and state information. This study is made in the context of MPLS VPN-based networks, with the aggregation of multicast VPNs in different real network scenarios and using novel heuristics

for aggregation. Still, the main problem of aggregation is the high percentage of wasted bandwidth that depends mainly on the amount of shared trees used.

On the other hand, recent works have brought back Bloom filters as an alternative for multicast forwarding. In this approach the packet header contains a Bloom filter that is evaluated at each hop for matching with the corresponding outgoing link ID. Although this approach is claimed to be stateless, it presents serious drawbacks due to false positives, namely important forwarding anomalies (duplicated flows, packet storms and loops) and the header overhead. In order to solve these drawbacks we propose D-MPSS (Depth-Wise Multi-Protocol Stateless Switching). This technique makes use of a stack of Bloom filters instead of a single one for all the path/tree, each one including only the links of a given depth of the tree. Analytical studies and simulations show that our approach reduces the forwarding anomalies present in similar state-of-the-art techniques, achieving in most network scenarios a forwarding efficiency (useful traffic) greater than 95%.

Finally, we study the possibility of using tree aggregation and Bloom filters together, and propose a set of techniques grouped as H-ABF techniques (hybrid aggregation - Bloom filter-based forwarding), which improve D-MPSS and other previously proposed techniques, practically eliminating the forwarding loops and increasing the forwarding efficiency up to more than 99% in most network scenarios.

Resumen

Aunque el servicio de multidifusión (multicast) basado en redes es la mejor manera de dar soporte a una gran variedad de aplicaciones populares como la televisión de alta definición (HDTV), el video bajo demanda (VoD), el servicio de LAN privadas virtuales (VPLS), la computación grid, las redes de área de almacenamiento óptico (O-SAN), la videoconferencia, la educación a distancia, los juegos masivos de rol en línea de múltiples jugadores (MMORPG), la realidad virtual en red, etc., hay varias razones tecnológicas y operacionales que le impiden un mayor despliegue. Esta tesis doctoral aborda este problema en el contexto de las redes troncales de transporte, proponiendo y analizando técnicas de bajo coste y escalables para dar soporte al multicast tanto para la capa óptica como para la capa de red (redes MPLS-IP).

En la capa óptica, en particular en las redes ópticas conmutadas por circuitos con multiplexación de longitud de onda (WDM), los diseños de nodos OXC con capacidades multicast muestran una gran complejidad y altos niveles de atenuación, principalmente debido a la necesaria operación de división de la señal, además del paso de ella a través de una compleja fase de conmutación. Esto hace que el soporte multi-punto sea raramente incluido en los nodos OXC comerciales. Inspirados en trabajos previos de la literatura, proponemos una novedosa arquitectura que combina lo mejor de dividir (splitting) y tap-y-continuar (TaC), llamado 2-STC (2-split-tap-and-continue) en el marco de trabajo de la óptica integrada. Un nodo OXC 2-STC es un diseño flexible capaz de hacer tapping (tomar una pequeña muestra de la señal) y dividir la señal hacia un máximo de dos enlaces de salida, con el fin de obtener una menor latencia terminal-a-terminal que en TaC y una mejorada distribución de la disponibilidad de potencia por encima de los diseños split-and-delivery (SaD). Otra ventaja de esta arquitectura es su simplicidad y el número reducido de componentes requerido, escalando bien para las implementaciones del nodo con muchos puertos de entrada/salida. Extensas simulaciones muestran que la división binaria (2-split) es prácticamente suficiente para la mayoría de las topologías de redes de transporte en la vida real, debido a que el grado promedio de los nodos es usualmente 3 y 4. Una variante de este diseño, llamada 2-STCg, para hacer el nodo capaz de realizar grooming (es decir, la capacidad de acomodar demandas de menor velocidad en longitudes de onda - enlaces) de tráfico óptico, es también presentada.

En la capa de red, una de las principales razones que obstaculizan el despliegue del multicast es la gran cantidad de información del estado de reenvío requerida en los enrutadores de la red de transporte, especialmente cuando un gran número de demandas multicast de tamaño mediano/pequeño llegan a la red de transporte, ya que

los datos de estado a ser almacenados en los enrutadores crecen proporcionalmente con el número de demandas multicast. En este escenario, estudiamos la agregación de demandas multicast en árboles de distribución, proporcionando un conjunto de técnicas para observar el equilibrio entre el ancho de banda y la información de estado. Este estudio está hecho en el contexto de las redes basadas en redes privadas virtuales (VPN) MPLS, con la agregación de VPNs multicast en distintos escenarios de redes reales y utilizando nuevos heurísticos para la agregación. Aún así, el principal problema de la agregación es el alto porcentaje de ancho de banda desperdiciado que depende principalmente de la cantidad de árboles compartidos usados.

Por otro lado, trabajos recientes han vuelto a traer a los filtros de Bloom como una alternativa para realizar el reenvío multicast. En esta aproximación la cabecera del paquete contiene un filtro de Bloom que es evaluado en cada salto para emparejarlo con el identificador del enlace de salida correspondiente. Aunque se afirma que esta solución no utiliza información de estado, presenta serias desventajas debido a los falsos positivos, esto es, anomalías de reenvío importantes (flujos duplicados, tormentas de paquetes y bucles) y gasto de ancho de banda por la cabecera de los paquetes. Para poder resolver estos problemas proponemos D-MPSS (Depth-Wise Multi-Protocol Stateless Switching). Esta técnica hace uso de una pila de filtros de Bloom en lugar de uno sólo para todo el camino/árbol, incluyendo cada uno sólo los enlaces de una determinada profundidad del árbol. Estudios analíticos y simulaciones demuestran que nuestra propuesta reduce los anomalías de reenvío presentes en otras técnicas similares del estado del arte, alcanzando en la mayoría de escenarios reales una eficiencia de reenvío (tráfico útil) mayor que 95%.

Finalmente, estudiamos la posibilidad de usar agregación de árboles y filtros de Bloom juntos, y proponemos un conjunto de técnicas agrupadas como técnicas H-ABF (hybrid aggregation - Bloom filter-based forwarding), que mejoran D-MPSS y las otras técnicas propuestas previamente, eliminando prácticamente los bucles e incrementando la eficiencia de reenvío hasta más de un 99% en la mayoría de los escenarios de redes.

1 Motivation of this Ph.D. Work

Data core transport networks not only carry regular Internet traffic, but also traffic of multimedia nature, such as voice trunks or IPTV, and data trunks from cellular networks or Virtual Private Networks (VPN). This is due to the well-known phenomenon of service convergence, which is pursued by current telecommunications operators. Thus, the modern core network engineering is focused on providing effective support to carry a wide variety of services in a cost-efficient way. Most of these services can benefit from the enhancement of backbone networks with native multicast capabilities. High-definition television (HDTV), video-on-demand (VoD), grid computing, optical storage area networks (O-SAN), file-sharing, software updates, RSS dissemination, video conferencing, e-learning, scientific applications requiring bulk data distribution among networked supercomputers, massive multiplayer online role-playing games (MMORPG) with more than 10 million subscribers [Act11] (in which the moves of players have to be spread to their virtual neighbors), and the case of large aggregates of multicast/broadcast/unknown virtual LAN traffic whose broadcast functionality must be emulated by the VPN service provider (named Virtual Private LAN Service, VPLS), are technologies that can take advantage of the multicast communication [ZP05].

Given the importance of multicast support, we shall concentrate the research work here presented in the development of new techniques to improve its implementation in core networks. To this end we shall address the following challenges.

1.1 Challenges

Regarding the optical network layer, nowadays *wavelength division multiplexing* (WDM) networks make use of *opto-electronic* (OEO) conversions at switching fabrics and regenerators. However, the multipoint service can also be cost effectively exploited at the optical layer when very high data rates are involved, and many research efforts have been focused on the reduction of OEO conversions, evolving into all-optical *lightpaths* and *light-trees* [SM99] circuits to realize the transparent optical network concept. Although there have been great advances in this field, state-of-the-art architectures proposed for the making of light-trees [HZ98, AD00b, AD00a, Rou02, PEAL10, PEAH10] present important counterbacks regarding attenuation levels, the large amount of components used and implementation complexity.

On the other hand, in *dense wavelength division multiplexing* (DWDM) networks, the standard wavelength spacity is over 10 Gb/s (tipically 160 wavelengths of 10

Gb/s each one, or 80 wavelengths of 40 Gb/s each one [HL04]). Although multicast traffic demands are supposed to transport multimedia flows (in any form) or VPLS broadcast traffic, they usually need lower rates than the one of a single wavelength. One of the most important features of the traffic demand patterns is their lower bandwidth granularity. Therefore, multicast streams must be multiplexed (groomed) into ligh-trees in order to take full advantage of the wavelength capacity. This process is known as *multicast traffic grooming*, which refers to the techniques used to combine lower-speed components onto available wavelengths [DR02]. In this aspect, most current state-of-the-art architecture nodes [BWA03, CM04, HL04, UMK06, SCS+08, AZS11] perform this operation by converting the signal into the electrical domain, where sub-demands are first multiplexed and then put onto single wavelengths. Further research is required toward *all-optical multicast traffic grooming* (AOMTG).

Regarding the network level, although many popular applications could be more efficiently served by network multicast support –in terms of bandwidth consumption–, there is little global deployment of multicast [RES06]. Traditional IP multicast schemes are scalable for very large multicast groups, but they present serious scalability issues with large numbers of distinct multicast groups [BFI+07]. That is why Internet Service Providers (ISP) are discouraged to enable this service, mainly because of the amount of the large amount forwarding information required at intermediate nodes and potential security vulnerabilities, given the fact that the forwarding state is actually triggered by end-users. This problem is not just circumscribed to the Internet, but to many other types of networks, and it has raised the concerns of operators and standards bodies. A practical low scalability bandwidth-expensive work-around valid for Internet users is *application layer multicast*. However, the rapidly growing popularity of VPLS (Virtual Private LAN Service), where the broadcast/multicast capability of a virtual private LAN needs to be emulated, has recently renewed the interest on efficient multicast support in the core.

Current proposals to mitigate the amount of the forwarding state information consist mainly of two technologies: the aggregation of multicast demands into shared distribution trees (in which they use the forwarding entries of the distribution trees)[RGE99, CC00, TH00, OKY+03, MGM06b, MGM06a, MYLS07, AMBM07, LZY08, FSZ11, ZW11], and the use of Bloom filters [Blo70] to encode the multicast routing tree into packet headers in order to make almost-stateless forwarding mechanisms [RJN+09, JZR+09, SRZ+10, ZJS+10, RMM+11, SRA+11, TRL12, TC12]. However, both approaches might generate importante levels of bandwidth wasted: in the first case because shared trees may deliver packets to non-desired destinations, and in the second case because the Bloom filters efficiency depend on the *false positives rate*, that causes important forwarding anomalies[SRA+11].

1.2 Objectives

The main objectives of this Ph.D. thesis are:

1. Optical network layer objectives:

    a) To propose an all-optical node architecture to solve the main counterbacks briefly described in sec. 1.1: reduction of the optical attenuation, and reduction of the number of components and complexity.

    b) To propose a traffic grooming node architecture with the important feature of preserving the switching grooming operations at the optical level, needing OEO conversions only for the addition of local demands.

2. Network layer objectives:

    a) To study the existing trade-off between the aggregation of multicast demands and the waste of bandwidth, and to propose new aggregation techniques in order to improve the state-bandwidth benefits.

    b) To propose new techniques for the improvement of the current state-of-the-art Bloom filter-based multicast forwarding approaches, focusing mainly in the reduction of forwarding anomalies caused by the false positives.

Organization of this Book

The present document is organized as follows. In the next chapter a state of the art of the current optical architecture nodes for circuit switching and traffic grooming is presented in the first sections, while the last section is dedicated to multicast performance at the network level: the most recent proposals found in the literature regarding the aggregation of multicast demands and Bloom filter-based forwarding approaches. In chapter 3 the novel 2-STC node (2-split-tap-and-continue) is presented for the realization of all-optical multicast circuits (light-trees), and the 2-STCg node as its variant for the enabling of traffic grooming. Chapter 4 presents the network-level techniques for both aggregation and Bloom filter-based multicast forwarding. Finally, the chapter 6 is dedicated to drawing the conclusions and future works.

2 State of the Art

2.1 Multicast in Optical Circuit-Switched Networks

The wavelength division multiplexing (WDM) multicast networks are evolving towards *all-optical* multicast routing, which is made up of several promising techniques designed to take advantage of light-trees (optical trees that avoid OEO conversions along the switching path). Although all-optical elements are not yet a mature technology, different research efforts held in this area during the last decade, are contributing to achieve this objective. In this section we analyze and summarize the most relevant problems regarding this topic, then explain some significant contributions made by several authors.

In order to realize optical multicast, the *lightpath* (an all-optical point-to-point path) concept was generalized in [SM99] to that of *light-tree*, which consists of an optical transparent point-to-multipoint channel originated at a source node that has more than one destination node. A light-tree spans several nodes and fiber links, and it is independent to bit-rate, protocol and format [Rou03], and eliminates the need of per-hop packet forwarding. An excellent survey of algorithms for light-tree construction (a.k.a. light-tree routing) over WDM networks can be found in [ZP05].

The key element of *light-tree routing* is the optical multipoint capability of the optical switches. Several *multicast-capable* node architectures have been proposed. One is the *SaD* (Split and Delivery) architecture [HZ98, AD00b], which is based on fixed or configurable power splitters [Rou02] that require optical amplication. From that initial architecture, several other structures were proposed, either focused on improving power efficiency or on fabrication cost. One of these improvements came through the split-sharing concept e.g. the MOSAD architecture. This concept prevents splitting for non-multicast cross-connections but it has the same power penalty as SaD for multi-point connections going through the shared splitter and, the saving in splitters comes at the price of internal blocking probability. Another approach is the *TaC node* (Tap-and-continue node) [AD00a], which improves power efficiency by tapping a small amount of the signal power to the local node when it is a leaf in the multicast tree. Tapping reduces the need for optical amplification at the cost of suboptimal light-tree construction, which, depending on the network topology, can lead to considerable over-occupation of wavelengths.

2.1.1 Data Plane of Optical Multicast

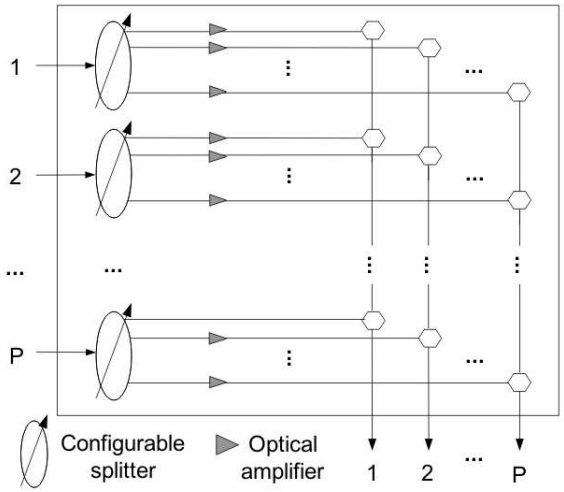
2.1.1.1 SaD-based MC-OXC Node Architectures

The *split-and-delivery switch* (SaD switch) was first proposed in [HZ98] as the main component of the *SaD-based multicast capable node*. This architecture was further modified in [AD00b] in order to reduce its cost and to improve the *power efficiency*. A completely optical multicast capable OXC node (MC-OXC) would be mainly composed of *passive optical light splitters* to enable multicasting, which divide the input power into several outputs without any knowledge about the optical features of the input signal and without changing any property of it except the power. A split operation contributes to power loss; for an ideal device the power of each output is the $(1/n)$-th part of the input signal.

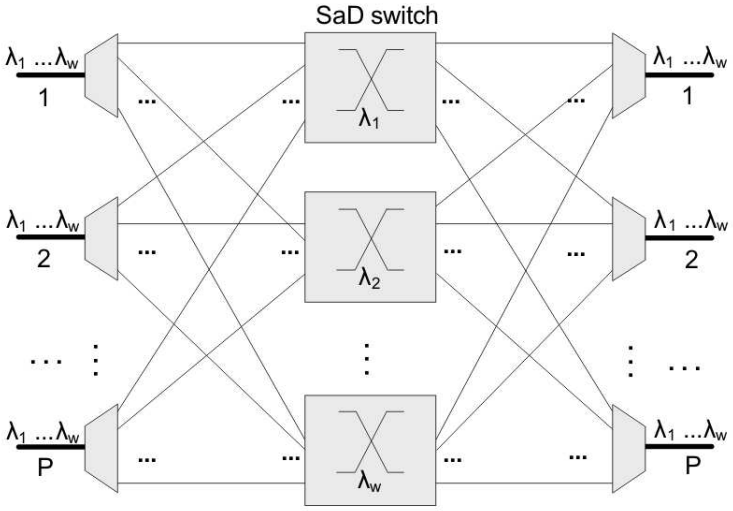
An improvement to the *SaD switch* was later proposed in [Rou02] that employed configurable splitters instead of passive splitters. Fig. 2.1 (a) represents a $P \times P$ SaD switch. These devices can be instructed to split the incoming signal into m outputs ($m \epsilon \{1, .., P\}$), where $m = 1$ corresponds to no splitting and $m = P$ to a broadcast operation. After splitting, each of the m split resulting signals may be switched to the correspondent output by using a P^2 photonic switch matrix made up of 2×1 optical switching elements. As shown in Fig. 2.1 (a), P^2 optical amplifiers are required. They are located after the splitter so that each split signal is optically amplified to compensate the power loss due to splitting. Some known drawbacks of optical amplifiers are their high cost, complex fabrication and amplification of noise levels.

Several other SaD-based node architectures were proposed in [HZ98]. Some of these proposals include built-in wavelength conversion capability i.e. nodes can set up point-to-multipoint connections to different output wavelengths. This capability improves overall blocking performance; however, for the sake of simplicity of comparison, in this work we shall refer only to single-wavelength splitting nodes. Equivalent designs could easily be obtained by placing wavelength converters at the outputs.

The basic and most well-known architecture is that of Fig. 2.1 (b), which is composed of a set of W $P \times P$ SaD switches (with $W = \#wavelengths$). Each input fiber is demultiplexed to extract the individual wavelengths, which are then directed to their corresponding SaD switches, where splitting and space switching operations take place. Finally, P multiplexers combine the W signals onto their corresponding outgoing fibers. Other architectures improve power efficiency for cross-connections that do not require splitting [AD00b] by applying the *splitting-sharing* concept. In this approach, all requests share a single power splitter that is attached to one of the switch ports. Therefore, only one multicast request can be attended at a time. It reduces costs and complexity, but it has poorer performance in terms of blocking probability for multi-point connections. Furthermore, multicast connections suffer the same process as in SaD, and hence require amplification. On the other hand, unicast connections do not have to traverse the splitter, which is a certain advantage.



(a) SaD switch with P inputs and P outputs.



(b) MC-OXC architecture based on SaD switches.

Figure 2.1: SaD switch and SaD-based node architecture [HZ98, AD00b]

2.1.1.2 Tap-and-Continue MC-OXC Node Architecture

With the aim of reducing the cost and improve the power efficiency of MC-OXCs a new architecture called *Tap-and-Continue* (TaC) was proposed in [AD00a]. The purpose of this architecture is doing without splitters to reduce the need for amplifiers in the network. In this approach, when the node is an end-point of the multi-point connection, only a very small fraction (0.5% - 10%) of the incoming power is tapped locally and the rest is switched to an output port. This fraction

is assumed to be enough for the local detector and most of the power is preserved for transmission to other nodes. This architecture contains, per each wavelength, a $(P+1) \cdot P$ *wavelength routing switching* (WRS) module and one shared *tap-and-continue module* (TCM), which taps to the local node. Fig. 2 (a) depicts the node and Fig.2 (b) depicts the TCM. As can be seen in the figure, the selected design is actually *shared tapping*.

The main disadvantage of this architecture comes from the lack of splitting functionality. In TaC the light-tree becomes a linear graph passing through all the nodes in the multi-point connection. This makes it appropriate for ring-like topologies but rather suboptimal in meshed topologies in terms of used wavelengths and average delay. Multicast connections can be realized by using only these TaC OXC nodes, and multicast routing algorithms for networks composed of these nodes have been proposed in [AD00b, ZJM00]. A mixed utilization of splitters (SaD) and tapping (TaC) in different parts of the network has been devised as the only intermediate approach to overcome the cons of TaC.

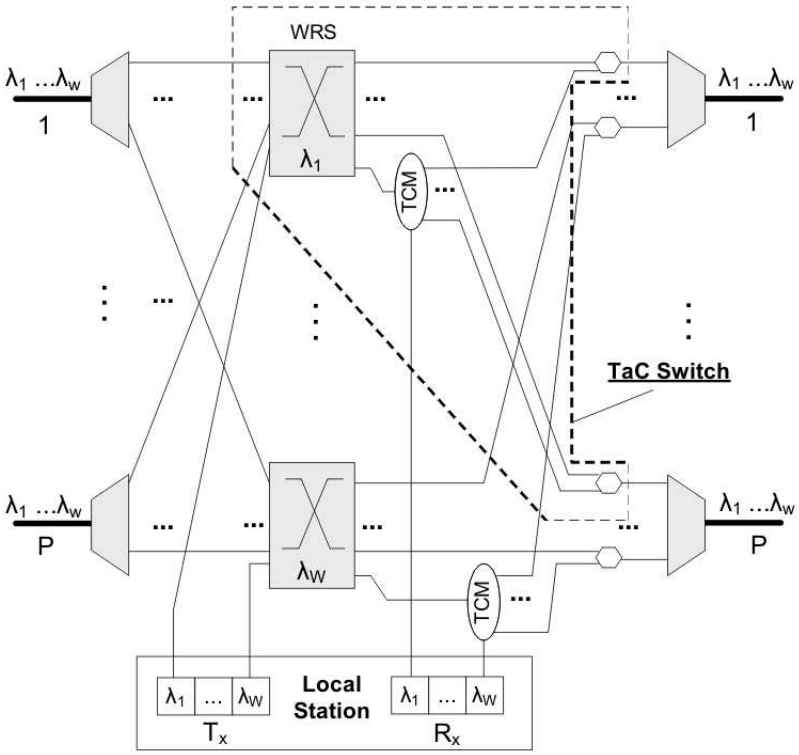
2.1.1.3 Other Optical Multicast Architectures

With the goal of minimizing the overall blocking probability in the network and ensuring that the provisioned multicast connections meet a prescribed bit error rate, the design of a new node architecture was proposed by authors in [PEAL10, PEAH10]. This node is made up of passive splitters and switches, as depicted in Fig. 2.3. *Variable optical attenuators* (VOAs) are present in this architecture and in an actual system they would be used to control the effects of *polarization depended gain/loss* (PDG/PDL) as well as to attenuate the input power to the post-amplifiers. Controllable *semiconductor optical amplifiers* (SOAs) are also introduced as gates to block the power at outputs where the signal is not destined for. All gates are controlled together in an intelligent manner to avoid clashing at the same output port and/or same wavelength of the switch.

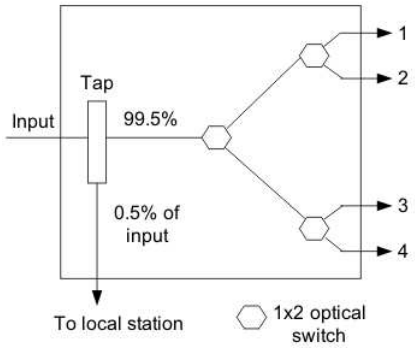
There is a variant of this node, where passive splitters are replaced by active ones. In this case no gates are required, since the splitters divide the signal only to the selected destined outputs. As it is observed, this design requires a large amount of components, leading to greater levels of attenuation and complexity.

2.1.2 Control Plane: MC-RWA and Power Loss Constraints

Regarding the control plane, the major issue to solve is the *multicast routing and wavelength assignment* (MC-RWA) problem. The problem of finding the optimal light-tree is NP-complete and is formalized as the Steiner tree problem [HRW92]. There is much research work for the MC-RWA problem, and proposed solutions deal with MC-RWA for a single multicast request, multiple static multicast requests, and multiple dynamic multicast requests. A wide state of the art can be



(a) A $P \times P$ TaC-OXC node architecture.



(b) A 1×4 TCM.

Figure 2.2: Tap-and-continue node and TaC module (TCM) [AD00b, AD00a]

found in [ZP05]. There are plenty of MC-RWA algorithms for static and dynamic demands, with and without considering wavelength conversion, in the literature [HCT01, WSW05, LPYP07, DJ11, SA11, ZPSZ08, BG12]. Most of the approaches are focused on reducing used wavelengths, max-min average distance from source node to destinations, etc. In our proposal, that will be presented further in chapter 3, we concentrate on power.

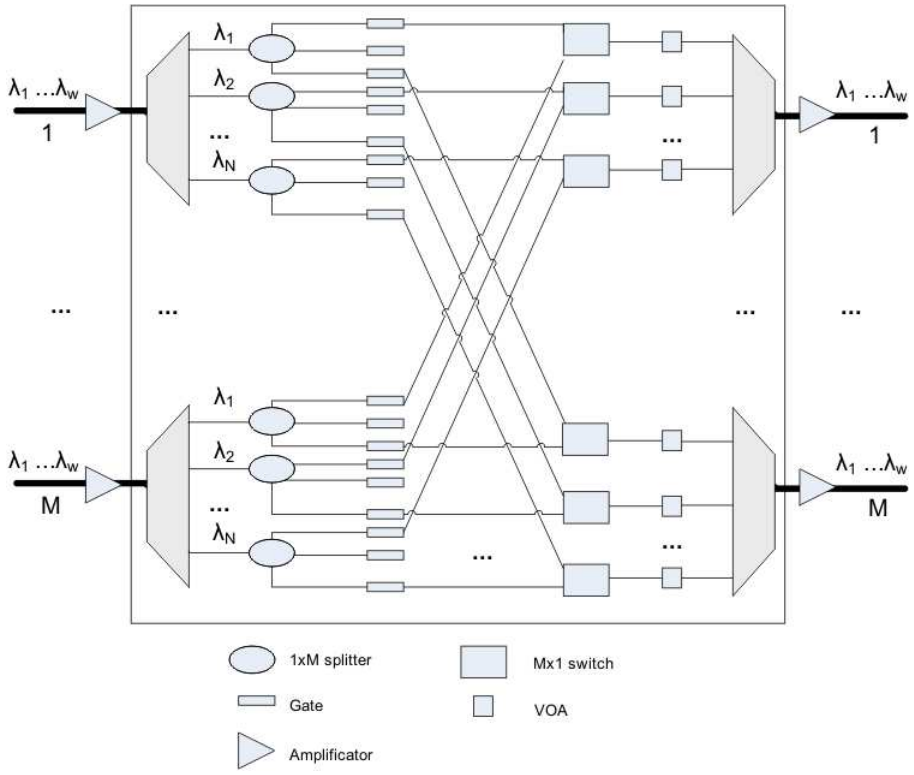


Figure 2.3: Node architecture for light-tree routing proposed in [PEAL10, PEAH10].

The splitting of an optical signal implies a considerable power loss that leads to a bound to the times a signal can be split. Furthermore, since optical amplifiers also amplify noise levels, it can be assumed that there is also an upper bound on optical amplification [AD00b, XR04]. In other words, reducing the amount of amplifiers in the network does not simply obey to monetary cost. There is also the fiber link attenuation. The problem of building light-trees under such optical power constraints [XR04, WWY01] is not the focus of this work. A balanced tree is found to be the optimum setting in SaD. In the case of TaC, the problem is reduced to the *open traveling salesman* problem.

2.1.3 Architecture Nodes for Multicast Traffic Grooming

In general, traffic demands may not occupy all the wavelength capacities of WDM core transport networks, which make *optical traffic grooming* a necessary capability for saving bandwidth. This capability consists of allowing nodes to *groom* sub-demands into wavelengths (i.e. demands that need sub-wavelength speeds). For example, in some developing countries, optical transport networks are not widely deployed mainly because of economical factors and geographical difficulties, and the

wavelength capacities might be under-utilized. On the other hand, multicast sub-wavelength transmissions are usually generated by an important variety of services and, in this sense, several technologies have been proposed to enable the grooming capability.

As we mentioned, in WDM networks the *traffic grooming* capability allows to groom small demands into wavelengths. To perform multicast grooming demands, special node architectures are required. Some of these are described in [Kam06]. To realize multicast, the signal has to be either split or duplicated, and this process may be optical or electrical. A multicast grooming node architecture was proposed in [BWA03], which has three switching levels, the last one allows demultiplexing tributaries on the client side. Authors in [CM04] proposed a mixed architecture, with two SaD units: one optical (o-SaD), and other electrical (e-SaD). The o-SaD unit splits the incoming signal and delivers the result to their destinations, i.e. a simple multicast procedure. The e-SaD has the capability to add, drop and copy tributaries (grooming); all performed on the electrical domain.

Another node architecture (shown in Fig. 2.4) where the duplicate process is held on the electrical domain was proposed in [UMK06]. In this architecture, tributaries that require duplication go to a *line terminal equipment* (LTE), and then to a *digital cross connect* (DXC). This device switches the wavelength to a buffer and duplication hardware that processes the tributaries. Then the tributaries are directed to the DXC and the LTE. Finally the resulting wavelength is directed to the correspondent destination. The figure shows an example in which an input light is duplicated electronically and forwarded to their corresponding outputs.

In [SCS+08, SFD08, SFD09] authors proposed the *stop-and-go light-tree* (S/G lightree) node, for unicast and multicast traffic grooming demands. The node has three parts: a routing unit, a label detection system, and a tributary add-and-drop system. Every sender node prepares the wavelengths with data and special tags. Wavelengths containing different tributaries have optical labels called *traffic tags* (TT). These were previously modulated by FSK (*frequency-shift keying*), meanwhile the real data is modulated by intensity (IM). The FSK labeling technique, which allows fast label recognition, was proposed, widely explained, and demonstrated in [KHF+05, VZC+03]. At the receiver, the optical signal is demultiplexed and switched. Then an n-splitter (n =number of outputs) is followed by an amplification phase. After that, a small percent of the incoming signal is tapped and the TT is analyzed by the label detection system. It consists of a FSK demodulator, a 1×2 fast switch, a bit interpreter, a contention resolution and an idle detection system with delay lines. Once the TT label is demodulated by the FSK demodulator, the bit interpreter determines if the tributary has to be stopped or if it is allowed to pass. This information goes to the 1×2 fast switch to physically realize the stop and go function. If the tributary is allowed to pass to the next node, it is switched to the second *optical switch fabric* (OSW). On the other hand, the tributary is switched to the add-and-drop system, where the tributary is dropped. The idle detection informs to the circuit decider about the new blank space available. It is also possible

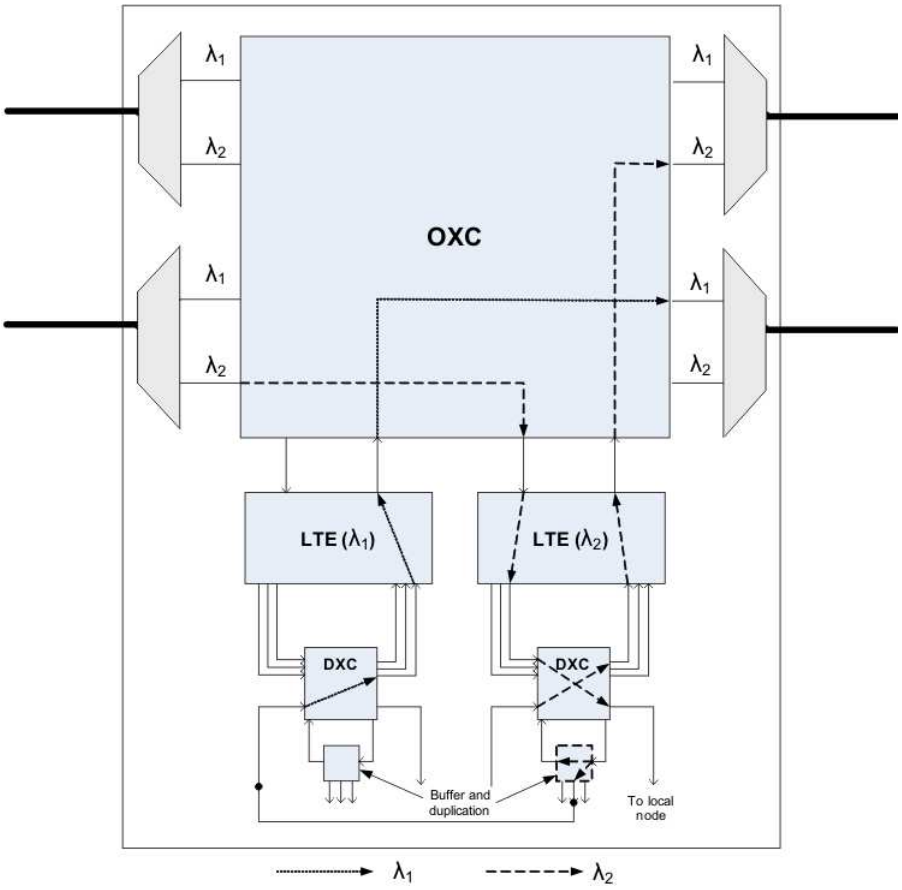


Figure 2.4: Multicast traffic grooming architecture node with LTEs, buffers and duplication hardware [UMK06].

to add new tributaries if there is available bandwidth in the wavelength.

Regarding more recent works, authors in [AZS11] introduced the *architecture on demand* (AoD) concept whereby transparent optical cross-connects (OXC) do not have an associated static architecture but can adapt their architecture to suit to the switching and processing requirements of the input traffic. The proposed AoD-OXC is implemented with a 96×96 3D-MEMS (*micro-electro-mechanical switches*) optical switch fabric functioning as an optical backplane that interconnects architecture-building modules that provide the required services such as arbitrary spectrum switching, time-domain sub-wavelength switching and optical multicasting. This approach provides flexibility and modularity for traffic demands.

2.2 Multicast in Packet-Switched Networks

Although we observe that many present popular applications could be more efficiently served by network-based multicast support, there is little global deployment of multicast [RES06]. Traditional IP multicast schemes are scalable for very large multicast groups, but they have scalability issues with large numbers of distinct multicast groups [BFI+07]. That is why Internet Service Providers (ISP) are discouraged to enable this service, mainly because of the amount of forwarding information required at intermediate nodes and potential security vulnerabilities, given the fact that the forwarding state is actually triggered by end-users.

However, the support of multicast has been one of the most studied problems, since it is the most efficient way –in terms of bandwidth consumption– to implement several important services and applications. This is the case, for example, of massively multiplayer online role-playing games (MMORPG), with more than 10 million subscribers [Act11], in which the moves of players have to be spread to their virtual neighbors. The adoption of Internet TV broadcasting, file-sharing, software updates, RSS dissemination, VoD (Video on Demand), video conferencing, e-learning, and grid computing, are other well-known applications that can take advantage of the point-to-multipoint support. Thus, the research on approaches to solve the drawbacks of multicast is still active.

This interest is not just circumscribed to the Internet, but to many other types of networks. In fact, the rapidly growing popularity of VPLS (Virtual Private LAN Service), where the broadcast/multicast capability of a virtual private LAN needs to be emulated, has raised the concerns of operators and standards bodies.

2.2.1 The BGP/MPLS VPN-based Multicast Networking Scenario

Among the technologies to deliver the VPN service, Multi-Protocol Label Switching (MPLS) is one of the most popular, due to the capability to realize traffic engineering and the compatibility with connection-oriented frame relay and ATM networks, optical networks (with generalized MPLS), and any layer-2 mechanism. One weakness of MPLS VPNs with respect to carrier-grade Ethernet is the multicast feature. Until recently, the standards for MPLS-based Virtual Private Network (VPN) service implementation provided only point-to-point delivery, and multicast could only be deployed by means of customer tunnels. However, the need for multipoint support inside the Service Provider networks soon became clear with the advent of Virtual Private LAN Services (VPLS), where the service provider (SP) must emulate a shared layer-2 broadcast/multicast network [AKF12].

The Internet Engineering Task Force (IETF) has made notable efforts to provide solutions for multicast MPLS VPN communications [AKF12, RR06, RA12, WMKT11]. The support of multicast for L3VPN [RA12] is at present the standard RFC6513 of the IETF, enhancing RFC4364 (BGP/MPLS IP VPNs). The same evolution is

going to follow the work in multicast for VPLS: Internet Draft [AKF12] has just entered the standards track and will soon become RFC. As noted in that work, one of the limitations of the existing VPLS implementation of multicast in RFC4761 [KR07] and RFC4762 [LR07]is that they rely on ingress replication. This means that the ingress Provider Edge (PE) replicates the multicast packet for each egress PE and sends them to the egress PEs using several unicast tunnels. Ingress replication may be an acceptable model when multicast traffic is low or/and the number of replications is small. Otherwise [AKF12] recommends the use of multicast trees to distribute VPLS multicast packets [YKWS+09] for its inherent bandwidth saving. This multipoint service would be alternative or complementary to unknown MAC address unicast packet flooding to egress PEs (before the ingress PE learns the destination MAC address of those unicast packets) that may still use ingress replication.

It is important to note that ingress replication is not intrinsically a disadvantageous feature for VPN Service Providers (SP): VPLS based on ingress replication makes the customer subscribe higher committed information rates because ingress replication requires more inter-site bandwidth. That means higher revenue for the SP. In other words, from the SP's viewpoint, if customers want the plug-and-play features of Ethernet, they will pay more, because it requires much more bandwidth than the unicast-only service. But then, what is the interest in featuring multipoint inside the network? The answer is competition and scalability. On the one hand, SP featuring tree-based multicast can provide cheaper VPLS and better QoS for the same subscribed rate than their ingress-replication-based competitors. On the other hand, ingress replicating VPLS would not scale to thousands of VPN sites. Furthermore, there are emerging large-scale VPN-based applications for which unicast MPLS LSPs is not an option due to their extreme scalability requirements. Most service providers (SP) have deployed private multicast configurations for high-speed multipoint traffic across their networks. It is the case of triple-play providers [SMM06], that deliver TV over IP multicast to their ADSL residential clients, where usually the last hop is delivered over IP unicast from a multimedia relay at the SP point of presence (PoP), or over native IP multicast to the set-top-box. This can easily scale to thousands of channels and users, if implemented with IP/MPLS over point-to-multipoint (P2MP) label switched paths (LSP) sent from a content delivery root to the relays [MYLS07]. Therefore, a careful utilization of multicast LSPs is paramount to build a more efficient and scalable multi-service network.

However, as pointed out in [AKF12, RA12, YKWS+09, MYLS07], the trouble with the tree approach is the trade-off between used bandwidth and forwarding state. The problem is that the core network nodes must keep per-multicast-group per-vpn per-tree information. A first step to mitigate this effect is the introduction of a single tree per VPN, in a similar fashion to the BUS server of LAN Emulation in ATM. In this setting, all sites in a VPN forward their multicast packets to the root of the tree. Note that this is a more radical approach than the Multicast Server (MCS) of RFC2022 for IP over ATM, or the PIM-SM rendezvous point in

the case of IP multicast routing, in the sense that the tree provides a broadcast channel per VPN, not per multicast group. However, even with a single shared tree per VPN, the network still keeps per-vpn forwarding state, which is against the basic design rule of VPN implementations: having VPN-specific information in intermediate nodes does not scale well. All VPN specific information should remain at the PE (Provider Edge) routers instead. Therefore, the standard approach [AKF12, RR06] recommends to use a number of trees to be shared by many VPNs, that conversely, introduces new bandwidth consumption penalty [MYLS07] and a complex traffic grooming optimization problem: how to aggregate multicast groups into shared trees. In [YKWS+09] authors identify the trade-off issue and claim that further work is required to study it.

In a BGP/MPLS VPN network, packets are unicast-routed without any state information about the VPNs kept in core routers. That information is only known by the provider edge (PE) routers, which connect sites directly to the VPN. Client data travels from a PE to another PE through core nodes within tunnels, usually *label switched paths* (LSPs). The associated state information in core routers depends only on the number of PEs, instead of the number of active VPNs. It may be the case that the SP is not interested in building aggregation trees (shared trees) over the backbone, or is not able to do it, as it happens in GMPLS optical core networks. In this event, the ingress PE router could make multiple copies of the packet and, by unicast, send each of them through a tunnel to the correspondent egress PE router.

In the case of multicast, routing for a specific group is optimal if and only if: when a PE router receives a packet from a Costumer Edge (CE) router for the multicast group, it forwards it to all the PE routers connected to CE routers of the group; the packet is not received by any other PE router, only one copy of the packet traverses each link, and the packet goes through the minimum cost multicast tree. The problem is that this optimality requires at least one AT per source and per multicast demand. Therefore, we notice that the state information at every provider router reaches a triple dimension: multicast source, multicast group and VPN. In this way networks scale poorly. Potentially, this would require unlimited amount of state information at the provider routers, because the SP has no control over multicast groups within the VPNs, or over the number of transmitters at each group, or over distribution of receivers.

Let us name MVPN a given VPN transporting multicast traffic, which is made up of multicast packets that a CE sends to the other CEs members (also attached to their respective PEs) of the multicast group. There are two aggregation models, defined by [AKF12] and [RR06], as depicted in Fig. 2.5.

Inclusive aggregation tree

In this model (Fig. 2.5(a)), a distribution tree could attend traffic of one or more MVPNs. The singularity here is that every PE router supporting a site associated to any of those MVPNs becomes a part of the tree. Since trees are unidirectional, the

number of routing forwarding entries (state information) is proportional to $n \times m$; where n is the number of aggregation trees and m is the average of PEs per MVPN. Even if each tree attends a single MVPN, the upper bound of the state information is proportional to the number of VPNs, not to the number of multicast groups inside those VPNs. This model has the inconvenience that, the more MVPNs are aggregated to a tree, the higher probability that some PEs receive useless leaked packets (packets not from their VPNs), incurring in bandwidth waste. This is because the aggregation is made without considering memberships to multicast groups.

Selective aggregation tree

In Fig. 2.5(b), a tree is used to transport traffic of a set of multicast groups from one or more MVPNs. In this case only PE routers from MVPNs are included. In other words, there is no aggregation unless another group had the same members than other, in which case a single tree would aggregate more than one multicast group. For very high bandwidth-consuming groups, the selective aggregation model should be preferred.

In Fig. 2.5(a) we can see that the aggregation tree AT1 would deliver pointless traffic into PEs that do not belong to the same group (g1, g2, and g3), and also to sites not associated to any group (g01, and g02). On the other hand, in the selective case of Fig. 2.5(b), unwanted traffic would be received only by PEs associated to different groups.

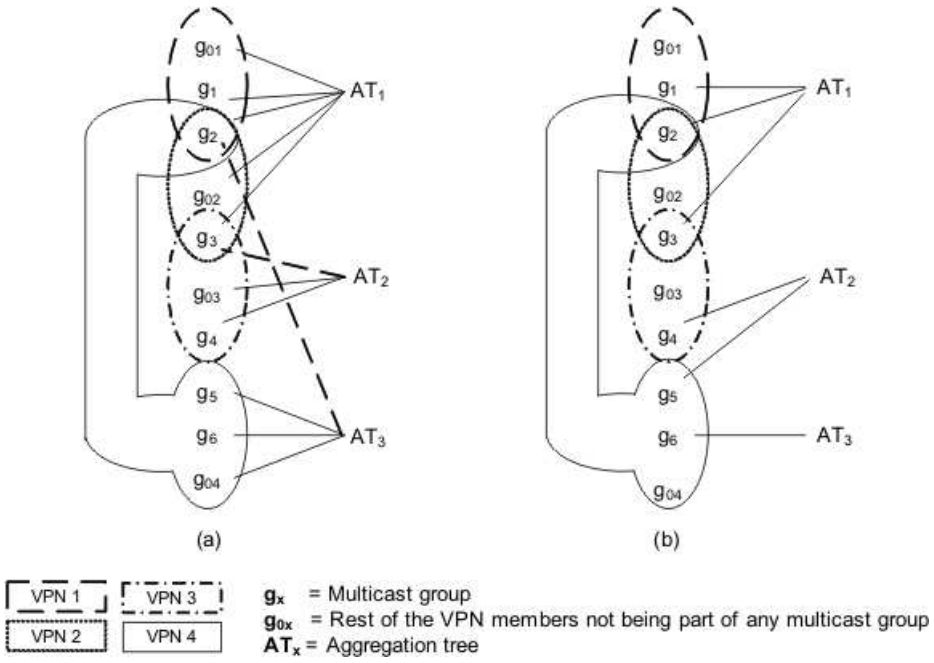


Figure 2.5: Inclusive and selective trees aggregation models: (a) Inclusive aggregation trees, (b) selective aggregation trees

2.2.2 Multicast Aggregation Approaches

We shall use the generic term multicast group to refer to a set of Customer Edge (CE) recipients of a VPN multipoint packet, be it members of an IP multicast group, the destination nodes of a multicast MAC address or the whole set of VPN member sites in the case of VPLS broadcast emulation. In this procedure, multiple multicast groups are forced to share a single multicast distribution tree (MDT) a.k.a. aggregation tree (AT) [RGE99, FCGF01a, FCGF01b].

The idea of aggregation was firstly studied in [RGE99], further exploited by A. Fei et al for IP multicast routing with/without MPLS in [CFGF01, FCGF01a, FCGF01b], and it has been revisited in the specific context of MPLS VPNs in several works [AC05, MYLS07] where different aspects are addressed together with aggregation, including point-to-multipoint (p2mp) tree signaling and frame encapsulation mechanisms on the shared tree.

In [FCGF01a] the authors addressed many implementation aspects of group management and introduce a centralized management entity called tree manager, which is in charge of assigning groups to existing trees or create new ones. They propose that the set of aggregated trees to be established can be determined based on traffic pattern from long-term measurements. They introduced the notion of perfect match (identical trees) and leaky match to denote trees delivering to non-member leaves. When no perfect match is found, a leaky match may be used if it satisfies certain constraint (i.e. bandwidth overhead (sum of leaked links) is within a certain limit). Otherwise the incoming tree is added to the network. In our proposal (presented in sec. 4.1), we shall use a more restrictive approach given the fact that in practice the maximum number of forwarding entries is pre-determined. Thus we enforce the usage of the best-match existing tree unless the group or VPN is very small; in this latter case, we shall construct ad-hoc small trees at the price of extra state information. In [FCGF01a] authors also introduced a number of metrics to measure the saving of IP forwarding state by aggregation that we shall reutilize in our work for the case of MPLS. In [AC05] authors use the same aggregation algorithm and applies the same to the particular case of VPN MPLS implementation based on label stacking. In [CFGF01], authors extended their own work introduced in [FCGF01a, FCGF01b] to support QoS by including measurement-based admission control to aggregated MPLS trees featuring Diffserv. Yet the forwarding state analysis and aggregation method is similar.

An alternative methodology to analyze the behavior and benefits of using shared aggregation trees in MPLS-based VPN networks was proposed in [MYLS07]. In this work, a simple state vs. bandwidth trade-off analysis was presented, showing the benefits of aggregation even if the best tree-matching algorithm is replaced by uniform random tree allocation. As we will explain later, like in this work, we present our results with respect to the aggregation ratio, as this parameter is more general than the absolute group numbers of [FCGF01a] or [AC05] and can be more useful in network planning. Another difference with previous analysis is the fact that the

distribution of VPN sizes was considered uniform over a narrow range of sizes (e.g. 2-10 in the case of [FCGF01a]).

Recently, several research efforts have been directed to finding the optimal construction of the shared trees and the best possible accommodation (aggregation) of the individual multicast demands into them. Research works presented in [ZXWY09, WZYY09, WGZY09, ZW10, ZW11, FSZ11] propose the use of optimization methods mainly based in artificial intelligence techniques. Thus, authors in [WZYY09] and [WGZY09] proposed a genetic algorithm and an immune algorithm for the optimization of aggregated multicast, respectively. In a similar manner, in [ZXWY09, ZW10, ZW11, FSZ11] the *ant colony optimization* (ACO) technique is employed, inspired in the ants behavior (in which *pheromones* traces are left by the ants whenever they have found a feasible solution). Note that these algorithms have the common inconvenience of needing, in most of the cases, large training periods and resources, and furthermore, long times of convergence to give stable results. In this sense, they could be more useful for off-line optimization calculations.

2.2.3 Bloom Filter-Based Forwarding Approaches

As a further step to solve the limitations present in the aggregation approaches, several works found in the literature have been developed by using Bloom filters [Blo70] as the base of their forwarding techniques. Some of the recent efforts is LIPSIN [JZR+09], an architecture for multicast forwarding in publish/subscribe systems, that places a Bloom filter into data packets for the multicast forwarding. In this mechanism, links are identified (instead of nodes) using Bloom filters to encode the forwarding information of source-routed trees into the packet headers. The forwarding decision is simple with very small forwarding tables. The proposed framework allows stateless operations. The main novelty of this method is the "inverting" of the Bloom filter thinking [BM05]: instead of maintaining Bloom filters at the network nodes and checking if incoming packets are included in the sets defined by the filters, Bloom filters are put in the packets and allow the nodes on the path/tree to determine which outgoing links the packet should be forwarded to. In general, we will refer to this model as the native Bloom filter-based forwarding approach.

Based on LIPSIN, the same authors proposed the *multiprotocol stateless switching* (MPSS) [ZJS+10]. This is a unicast/multicast source routing mechanism that consist of encoding the path/tree that every packet follows, in a Bloom filter carried in the packet header. The MPSS architecture [ZJS+10] is claimed to be the marriage of MPLS and the Bloom filter-based forwarding approach. The label switching forwarding mechanism of MPLS is replaced with the Bloom filter-based forwarding mechanism. As in LISPIN, every unidirectional link of the network is identified with a set of m bits representing a Bloom filter element, from which k bits are set to 1 ($k \ll m$). These almost-unique link IDs are randomly generated. It is the

equivalent to say that k hashes have been applied to the Bloom filter. Once the routing path/tree is calculated, in order to build the source-routing Bloom filter, IDs of the links that are part of the path/tree are OR-ed together (Fig. 2.6). The resulting array of bits is the forwarding Bloom filter, known as *zFilter*, which replaces the MPLS label, in order to forwarding decisions be made based on the evaluation of the Bloom filter. Forwarding nodes maintain just a small link ID table made up of its outgoing interfaces. Routers map incoming packets to their outgoing interfaces just by AND-ing the zFilter with link IDs: if the result is the link ID itself, the packet is allowed to pass, if not, it is discarded (Fig. 2.6). Since intermediate nodes only need to keep the outgoing link IDs, this approach is virtually stateless, at the cost of extra packet processing.

The previous methods, as inherent to Bloom filters, are subject to false positives, since a link ID may match the Bloom filter even if it had not been intentionally added to it. This generates forwarding anomalies, such as packet storms, forwarding loops, and duplicate flows [SRA+11]. Thus, recent improvements have been proposed [SRA+11, RMM+11, TC12], that have mitigated them in a probabilistic fashion. Another problem not treated yet is that, in order to reduce the probability of false positives, Bloom filters in large networks or large multicast trees need to be very large, leading to excessive transmission. A deep analysis of the false positives rate (fpr) generated by the Bloom filter-based forwarding techniques and of the forwarding anomalies will be provided in sec. 4.2.1.

In order to reduce the false positives rate of Bloom filters, authors in [JZR+09] introduced *link ID tags* (LITs), as an addition to link IDs, where every unidirectional link is associated with a set of d distinct LITs, which allows to construct different candidate zFilters and to select the one performing the lowest fpr. The forwarding information is then stored in d forwarding tables, each containing the LIT entries of the active link IDs. This probabilistic improvement has the counterback that increases the state information and complexity of the zFilter formation and the matching process.

Authors in [SRA+11] proposed to vary the k and m values for the encoding of multicast trees. If a multicast tree is short, i.e. makes use of a few links, it is not necessary the Bloom filter array to be large. Thus, by following a set of steps and formulations the Bloom filter is reduced. As we will show later in this work, this idea has also been adopted in one of our proposals (D-MPSS), but for changing the size of the Bloom filter of a given depht-tree.

Also in [SRA+11] authors propose to use *bit permutation patterns* for Bloom filters at each hop. In this method the filter is constructed by rearranging it hop-by-hop according to the almost-unique random bit patterns kept at every node. By following this, in the moment of performing the forwarding, in an implicit manner, the packet keeps the history of the links that it has to traverse, reducing the probability of packet storms and forwarding loops. In our proposal that will be presented further, for the sake of comparisons, this method is implemented in simluations (sec. 4.2.3).

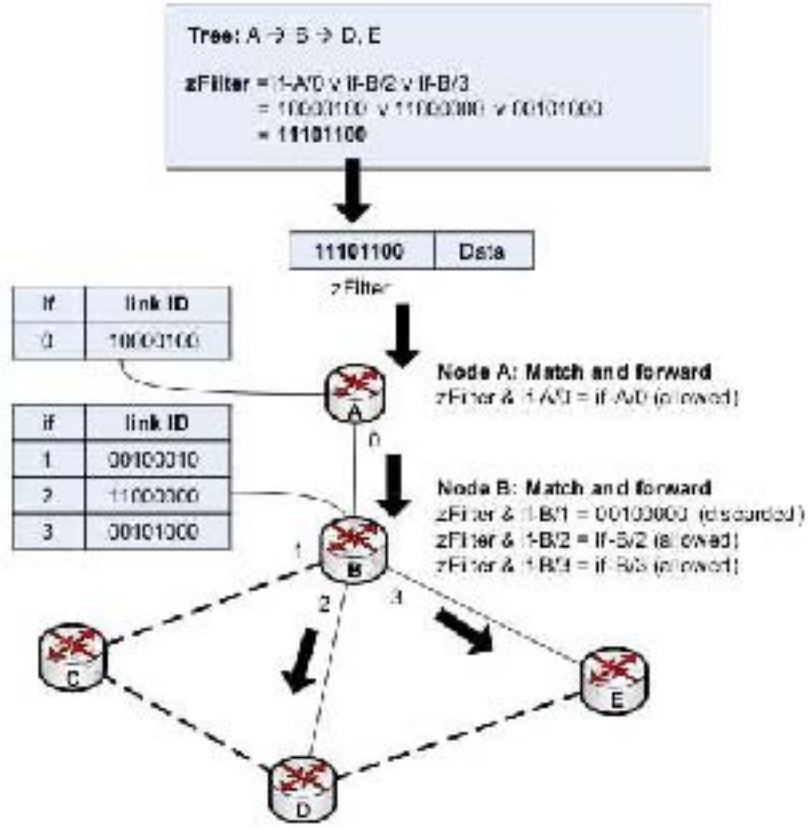


Figure 2.6: The MPSS zFilter formation and forwarding decision

In a similar manner, the recent work developed by C. E. Rothenberg et al. [RMM+11] describes a method to enable false negative free element deletion. This mechanism mark deletable regions within the Bloom filters in order to delete (the opposite of elements insertion) elements from them. This is useful, for example, if some inserted elements need to be processed only once, and this deletion would help to reduce the fill factor of the Bloom filters (the proportion of bits set to one in the array). Note that this method introduces more complexity for the construction of the in-packet Bloom filters and for the packet processing.

Finally, a destination-oriented approach with a great level of complexity that fully eliminates the forwarding loops was recently proposed in [TC12]. In this scheme, the filter carries destination IP addresses, not links of the path/tree, but tree nodes need to maintain state.

3 Tap-and-2Split Node for Optical Multicast in OCS Networks

3.1 Overview

In this section we present a novel cost-effective *multicast-capable optical cross connect* (MC-OXC) node architecture that features both *tap-and-continue* and *tap-and-binary-split* functionalities. This architecture provides an interesting balance between simplicity, power efficiency and overall wavelength consumption with respect to models based on TaC (Tap and Continue), proposed by authors in [AD00a], or SaD (Split-and-Delivery), proposed by authors in [HZ98, AD00b, Rou02]. These and other important references are explained in the state of the art presented previously in sec. 2.1.

The main concept of this architecture is to use a node able to split the signal only up to 2 outgoing links (*binary splitting*). This design decision is taken in order to preserve the power of the optical signal, since the splitting operation for optical multicast divides the signal by the number of outgoing interfaces, and generates high rates of power loss. The *binary splitting* has the important advantage that the node architecture becomes much simpler and uses a many fewer number of components, which is very convenient for the power saving purposes. Since the average degree of the core nodes in referenced optical transport networks is between 2 - 4, the *binary splitting* nodes will not cause an important increase in the number of wavelengths and links used by the light-trees.

The main component of this node is a novel *Tap-and-2-Split Switch* (Ta2S), which is proposed, analyzed an implemented (by simulations) in this section based on integrated optics (namely, MMI taps and MZI switches). We characterize and compare it with other alternatives implemented with the same technology. The study shows that, thanks to the presented Ta2S design, the 2STC node scales better in terms of number of components than the other alternatives. Moreover, it is more power efficient than the SaD design and requires less wavelengths than TaC thanks to the binary split capability. On the other hand, another very important conclusion is that the simulation results reveal that the 2-split condition (*binary splitting*) does not add a significant additional wavelength consumption in usual network topologies with respect to SaD.

As explained in sec. 2.1.3, the traffic grooming capability is also an important characteristic to implement in an OXC node. In this chapter we also propose to extend

the 2-STC architecture for making it capable of groomming sub-wavelength demands into light-trees. The resulting node is called 2-STCg, and it is implemented with *optical ring resonators* (ORR), used for the deletion of optical signals when its required, designed with integrated optics technology.

All the work described in this chapter related to physical implementation of the architecture in *integrated optics* (IO) has been carried out by Dr. Pedro Contreras and Dr. Carmen Vazquez [FVLL09, Lal11] from Displays and Photonics Applications Group at Electronic Technology Department of UC3M, from a fruitful collaboration starting from an original idea at the functional level of this Ph.D. author and his supervisor. In works reported at sec. 3.3 Dr. P. Contreras and Dr. C. Vazquez have worked in integrated optics designs and functionality aspects of the nodes.

3.2 The Novel 2-Split-Tap-Continue Node Architecture

3.2.1 General Node Architecture

We aim to combine the advantages of TaC and SaD in a cost-effective design based on integrated optics. Fig. 3 depicts the general architecture for the *2-split-tap-and-continue node* (2-STC node) firstly sketched by the authors in [FLVL08]. The node has the following capabilities:

Tap-and-continue (TaC): Besides switching, it taps a small fraction of the input power to the local node. Unlike TaC nodes [AD00a] described above, tap is always performed. In TaC nodes, the tap is optional, but it increases the number of switches needed.

Tap-and-2-split (Ta2S): The node should tap and perform 2-split for multiple requests in a strictly non-blocking manner. As described, tap is always performed.

The general architecture is very similar to a SaD-based node [HZ98, AD00b], but instead of SaD switches, a novel module *2-Split-Tap-Continue Module* (2-STCM) is used, which will be explained later. It needs P multiplexers to extract individual wavelengths from each input fiber and P demultiplexers to combine individual wavelengths onto output fibers. A matrix for receiving taps is used (R_x) at the ADM local station, and it could be necessary to have a selector in every 2-STCM because the tap operation is made for every input port.

3.2.2 The 2-Split-Tap-Continue Module (2-STCM)

3.2.2.1 Tap-and-2Split switch (Ta2S switch)

The 2-split operation is the key of our proposal for two reasons:

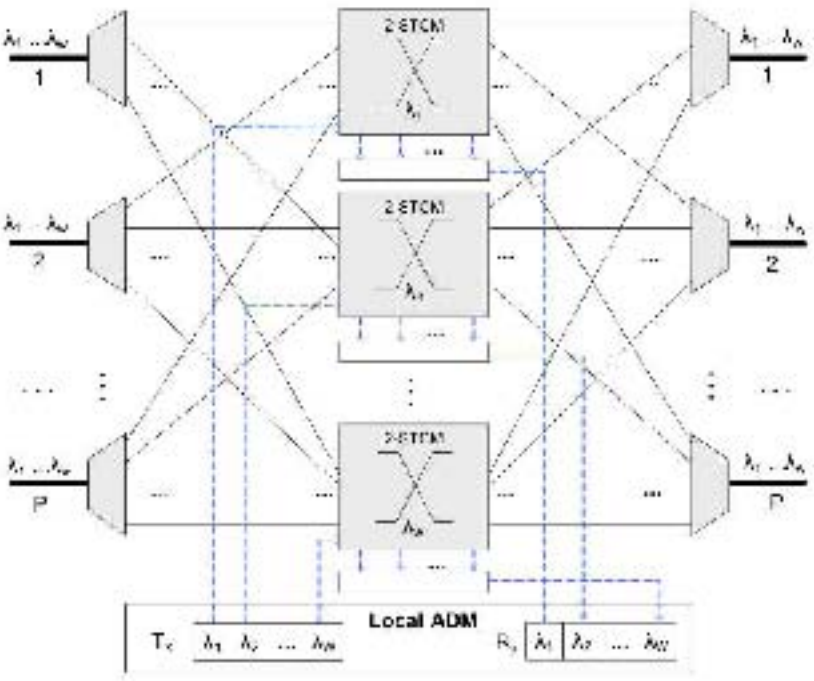


Figure 3.1: General architecture of a $P \times P$ 2-STC MC-OXC node with novel 2-STCM.

1. Realizing 2-split saves power efficiency, since it divides the incoming power only into two outputs. We will show later that, in generic core networks, the 2-split condition is sufficient and achieves a good trade-off between power efficiency and the number of resources used.

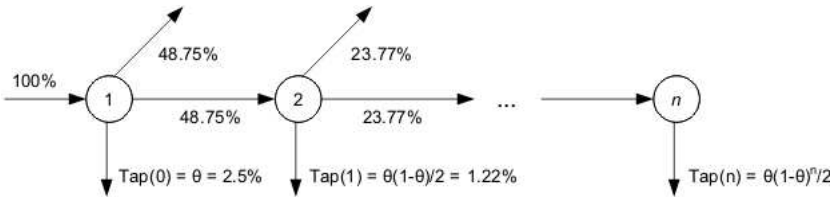
2. A 2-split operation is much easier to perform than an n-split one ($n > 2$), in terms of used components, complexity and fabrication tolerances.

Therefore, we need a device with a single input signal and two output signals, able to switch to any of the outputs (0:100 or 100:0), or to share the optical power to both outputs (50:50); and has also to be able to tap a fixed value of the incoming signal. This novel device is named *Tap-and-2-Split switch* (Ta2S switch) that can be made with existing photonic devices.

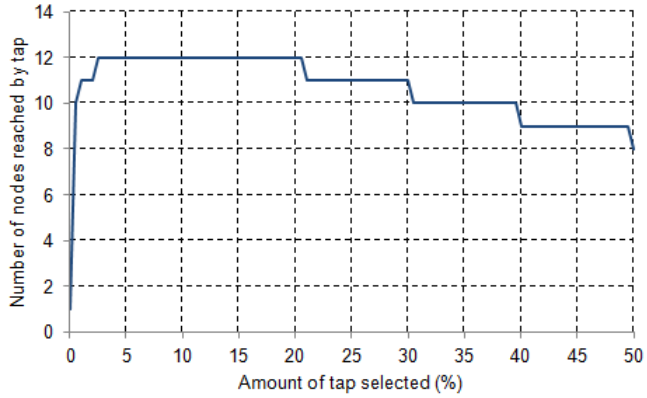
The fixed tap value has been determined as follows. Let us consider a power budget permitting 51 dB of attenuation. That power is at 100% at the root of the light-tree shown in Fig. 3.2 (a). The signal has to traverse n nodes performing tap-and-2-splitting until it can not be detected by the $(n + 1)$-th node. Fiber attenuation on the links, power amplification, power loss generated by nodes are not considered except those caused by the binary splitting (3dB). Different values of fixed tap (at every node) have been tried out in order to find the most adequate one. Results are shown in Fig. 3.2 (b), and it can be seen that the best performance is achieved

when the fixed amount of tap at every node is set between 2.5% - 20.5% (tap can be detected until the 12-th node). For our calculations, a fixed amount of 6% tap will be considered.

An implementation of a Ta2S using a MZI configuration and a fixed tap is shown in Fig. 3.3.



(a) Ratio of input power tapped in a branch made up of n nodes performing tap-and-2-splitting, with a fixed tap of 2.5%.

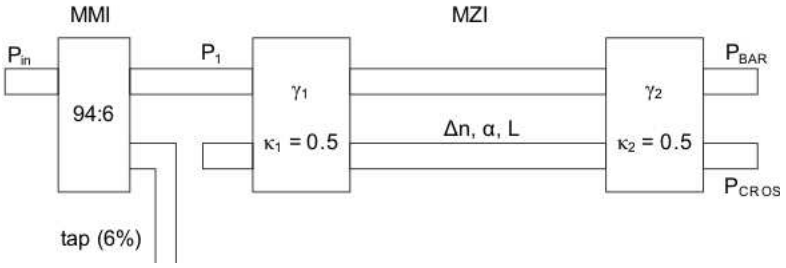


(b) Number of nodes of the branch that receive the tapped signal with less than 51dB of attenuation vs. fixed selected tap rate %

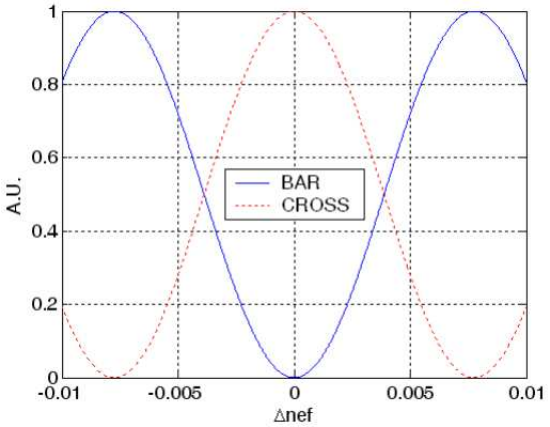
Figure 3.2: Theoretical outreach of tapped power on a branch with tap-2-splitting.

The first stage of the Ta2S switch block diagram is a 94:6 splitter to perform the tap operation. The proposed optical switch is based on a MZI, shown on the right in Fig. 3.3 (a). It is formed by two 3dB couplers, with coupling ratios $\kappa_1 = \kappa_2 = 0.5$ and excess losses γ_1 and γ_2, joint by two waveguides of length L and attenuation α. The switch output is selected by modifying the refractive index difference between both waveguides, Δn.

Thus, when $\Delta n = 0$, at initial setup, all the optical input power is guided to the P_{CROSS} output port. When the refractive index difference is induced, either by applying current or by thermo-optic effect, the optical power at both output ports changes. Output power variations, P_{BAR} and P_{CROSS}, with the refractive index difference changes between the waveguides are shown in Fig. 3.3 (b). Outputs are symmetrical, a maximum in cross output coincides with a minimum in bar output,



(a) Schematic of a Ta2S switch composed of a fixed splitter and an MZI.



(b) Bar and Cross Outputs of MZI vs Δn.

Figure 3.3: Schematic of a Ta2S switch and output power response when Δn modified [Lal11].

and the optical power at each port can be adjusted modifying Δn.

A compact solution can be realized using *Multi Mode Interference* (MMI) splitter for the tap [FLC07] and 3dB MMI couplers [VMHG95] in the MZI. The MMI operation is based on the property of self image of the light propagation in planar waweguides. An MMI coupler consists of two inputs and two outputs attached to a wider section waveguide whose length determines the coupling ratio. The proposed 3dB MMI coupler has input and output waveguides placed at a third part of the MMI width in order to obtain a shorter device, and its length is given by:

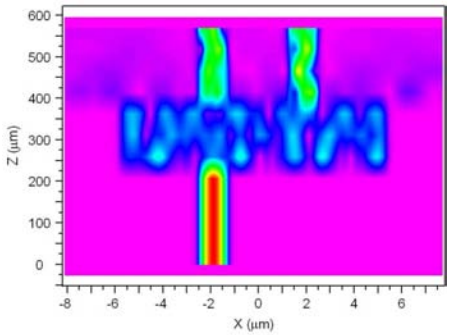
$$L_{MMI}(3dB) = \frac{1}{2}L_\pi = \frac{1}{2}\frac{4 \cdot W_e^2 \cdot n}{3 \cdot \lambda_0} \tag{3.1}$$

Where L_π is the beat length, W_e is the MMI width, λ_0 is the vacuum wavelength and n is the effective refractive index of the waveguide [FLC07, VMHG95].

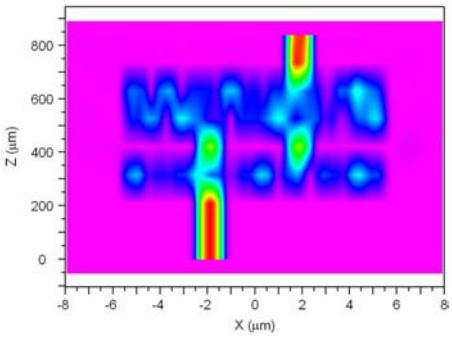
Simulations using the *Beam Propagation Method* (BPM) at $\lambda_0 = 1550nm$ were carried out in order to show the operation principle. The effective refraction index

of the ridge waveguides was 3. The waveguides were $1.3\mu m$ wide, and the 3dB MMI couplers are $11.3\mu m$ wide and $170\mu m$ long. A detail of the optical power distribution in the 3dB MMI is shown in Fig. 3.4 (a). Two simulations of the MZI with $100\mu m$ long waveguides between the two couplers are shown in Fig. 3.4 (b) and (c). The first one shows the initial status, while the second one correspond to $\Delta n = -0.00775$ simulation. The output for both cases has changed.

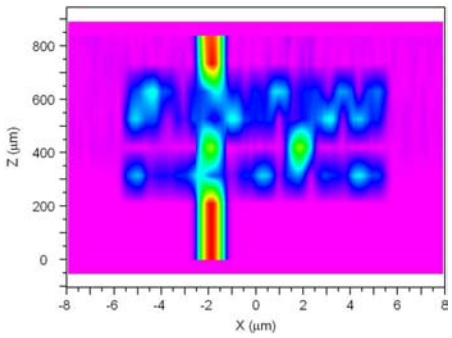
The Ta2S can also be realized using a tunable splitter [LJ01], a 2×2 MMI with symmetric splitting ratios.



(a) Distribution of the optical power in a MMI 3 dB coupler.



(b) Distribution of the optical power in MZI at initial status.



(c) Distribution of the optical power when the MZI is switched.

Figure 3.4: Distribution of the optical power for 3dB MMI and MZI switch[Lal11].

3.2.2.2 Block Diagram and Components

Given that traditional switch fabrics, such as the ones introduced in , in order to make the light pass through the less amount of optical switches in average, we propose the special design depicted in Fig. 3.5 as the *2-STC module* block diagram.

However, for implementation purposes, it can be converted to the one of Fig. 3.6 (a) that shows a 4×4 *2-STC module*. Then, the design for the 8×8 *2-STC module* is presented in Fig. 3.6 (b). The P input signals go through P *Ta2S switches*, which are instructed to perform the required operation. The resulting signals traverse $log_2 P + 1$ phases of switches instructed adequately to guide them to the desired ports.

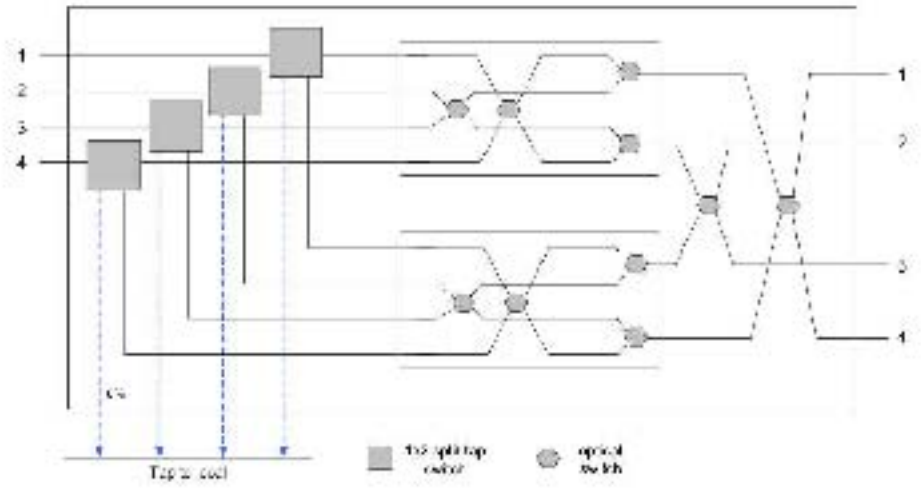
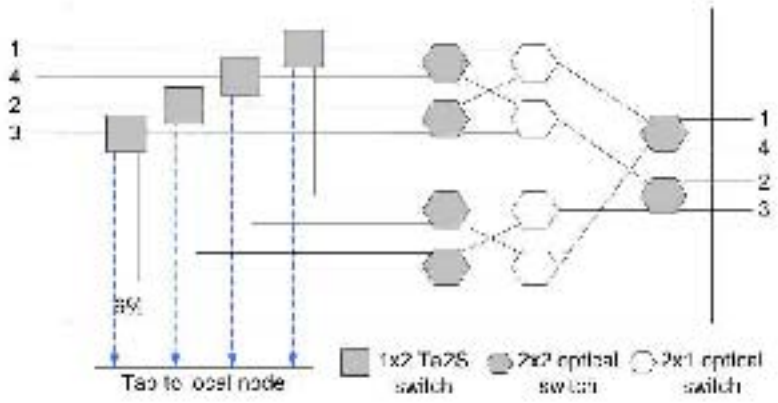


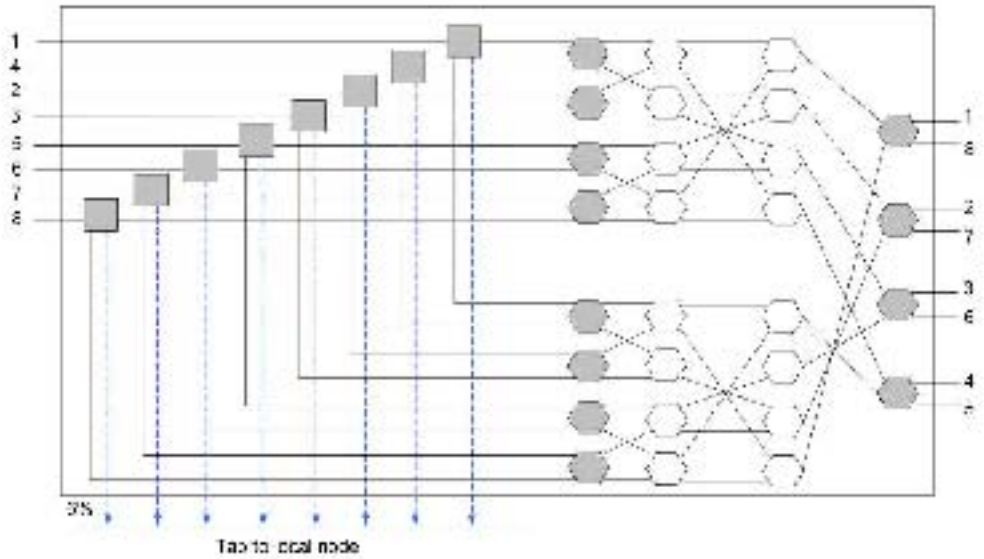
Figure 3.5: Initial design of the 4×4 2-STCM.

Note that the optical switching stage here designed is not similar to a typical $N \times M$ crossbar switch matrix (where $N = 2P$, $M = P$). In such a matrix the best case of a switch operation is given when the optical signal traverses only one switch, the worst case when it has to traverse $N + M - 1$ switches, and an average case, $(N + M)/2$ switches. The interconnection network presented here is designed to use the least possible number of switches. Thus, a signal always traverses a fixed amount of $log_2 P + 1$ switches (number of phases).

For a $P \times P/2$ matrix, if P is a power of 2, each of the first $log_2 P$ phases has P switches and the last one $P/2$. If that is not the case, a $P' \times P'/2$ switch matrix would be given (with $P' = 2^{\lceil log_2 P \rceil}$), and then it should be *pruned* in order to remove the unnecessary switches to get exactly what is needed for P ports. In order to get more compact designs in the future, those switches can also be implemented on the same chip using ring resonators [VVSP05].



(a) The 4×4 2-STCM.



(b) The 8×8 2-STCM

Figure 3.6: Block diagram of the 2-STCM. Input and ports are arranged for easy representation.

3.2.3 Node Architecture Evaluation

3.2.3.1 Number of Components and Power Loss

Let us now analyze the different factors of attenuation in the three basic node architectures under study, based on SaD, TaC and 2-STC switches. For this purpose we shall compare integrated optics implementations of the three models.

3.2.3.2 Power loss of the SaD–based node

The SaD switch is proposed to be made up of P configurable splitters [HZ98]. We propose a configurable splitter by cascading 2×2 MMI's with symmetric splitting ratios. In a single 2×2 MMI with symmetric splitting ratios [LJ01] the output power from one input can be tuned to the other by biasing the refractive index in specific sections of the MMI. According to simulations [LJ01], it has output ratios of 50:50 (i.e. a single split, initial state) and 0.1 dB excess losses with no biasing. But when it is biased to obtain 0:100 or 100:0 split ratios, it actually obtains 1:71 and 71:1 split ratios, mainly because of the attenuation losses caused by biasing. Therefore, for a switched signal simulation losses are around 1.55 dB. As an example, we present a possible implementation of a 1×4 configurable splitter in Fig. 3.7. In general, it would be necessary to align $P - 1$ 2×2 MMI's with symmetric splitting ratio in cascade, arranged in $log_2 P$ phases ($P = \#ports$). Depending on the operation needed, the input power would incur in different losses (see Table Tab. 3.1).

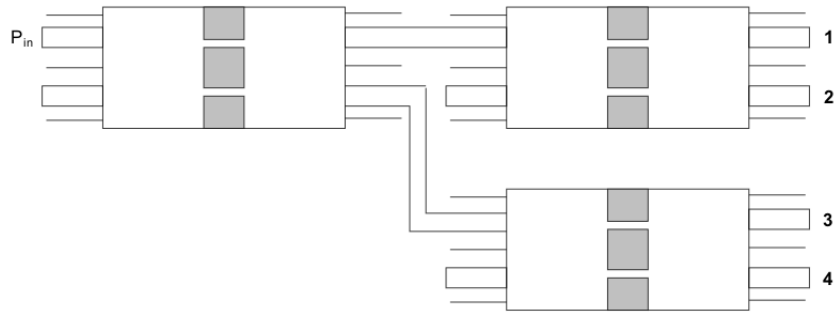


Figure 3.7: A 1×4 configurable splitter using 2 x 2 MMIs with symmetric splitting ratios [Lal11].

Selected output ports	Loss in 1st phase	Loss in 2nd phase	Total loss
1, 2, 3 and 4	0.1	0.1	0.2
1, 2, 3 or 4	1.55	1.55	3.1
1 and 2, or 3 and 4	1.55	0.1	1.65
1 and 3, or 2 and 4	0.1	3.1	3.2
1, 2, 3, or 2, 3, 4	0.1	1.55	1.65

Table 3.1: Power loss (dB) in the configurable splitter of Fig. 3.7.

Loss grows with the number of ports because the number of phases also increase. For instance, for a 1×8 configurable splitter, 3 phases will be needed and losses will be in the order of 4.65 dB for a single switch operation in continue mode. For a 1×16 (or 1×12) device (4 phases), losses will be in the order of 6.2 dB for a single switch; and so on.

Besides the power losses caused by configurable splitters, a signal has to traverse an optical switch matrix (see Fig. 2.1 (a) for reference), which contributes to the overall loss. Therefore, the power loss (in dB) of an input signal in SaD switch is given by:

$$PL_{SaDswitch} = \begin{array}{c}(N_{MMI-bias} \cdot PL_{MMI-bias}) + (\log_2 P - N_{MMI-bias}) \\ \cdot (PL_{MMI-no-bias}) + 10\log_{10} m + N \cdot PL_{sw}\end{array} \quad (dB) \tag{3.2}$$

Where P is the number of ports, $N_{MMI-bias}$ is the number of 2×2 tunable MMI splitters, $P_{LMMI-bias}$ is the power loss of a tunable 2×2 MMI splitter when changing power ratios to 0:100 or 100:0, $P_{LMMI-no-bias}$ is the power loss of a tunable 2×2 MMI splitter when not biased, m is the number of split outputs, N is the amount of optical switches the signal has to traverse at the switch matrix and PL_{sw} is the insertion loss of a 2×1 optical switch.

Finally, the overall loss of an input signal in a SaD-based node (see Fig. 2.1 (b)) is:

$$PL_{SaD-based\,node} = PL_{dm} + PL_{SaD\,switch} + PL_{mx} \quad (dB) \tag{3.3}$$

Where PL_{dm} and PL_{mx} are the power losses caused by demultiplexing and multiplexing, respectively. The expression 3.3 is given only as a reference, but since the power loss for demultiplexing and multiplexing is the same for SaD, TaC and 2-STC architectures, we do not take them into account, because they do not make any difference in the comparison.

3.2.3.3 Power loss of the TaC node

In the case of TaC nodes (see Fig. 2.2), losses are caused mainly by the optical switching elements of the WRS and TCM modules. Notice that when tapping, the signal has to traverse $\log_2 P$ optical switches. Therefore, the loss is:

$$PL_{TaC\,node} = (N_{WRS} + B \cdot \log_2 P + 1) \cdot (PL_{sw}) + B \cdot L_{tap} + PL_{dm} + PL_m \tag{3.4}$$

Where P is the number of ports, PL_{dm}, PL_m and PL_{sw} have the same meaning as in expressions 3.2 and 3.3, N_{WRS} is the number of switches the signal has to traverse in the WRS module, B is a boolean variable (equal to 1 when the tap action is performed, otherwise is 0) and PL_{tap} is the power loss of the fraction tapped to the local node, given in dB.

3.2.3.4 Power loss of the 2-STC node

To analyze the 2-STC node we need to model its main component, the Ta2S switch. A description of the Ta2S switch has been done using Matlab. Block diagram of the proposed MZI switch is shown in Fig. 3.3 (a). The device behavior is determined by the expressions:

$$\left|\frac{P_{BAR}}{P_{IN}}\right| = \Gamma \cdot \left[\begin{array}{c} (1-\kappa_1)\cdot(1-\kappa_2)\cdot e^{-2\cdot\Delta\alpha\cdot L} + \kappa_1\cdot\kappa_2 \\ -2\cdot\sqrt{(1-\kappa_1)\cdot(1-\kappa_2)\cdot\kappa_1\cdot\kappa_2}\cdot e^{-\Delta\alpha\cdot L}\cos(\Delta\beta\cdot L) \end{array} \right] \tag{3.5}$$

$$\left|\frac{P_{CROSS}}{P_{IN}}\right| = \Gamma \cdot \left[\begin{array}{c} (1-\kappa_1)\cdot(1-\kappa_2)\cdot e^{-2\cdot\Delta\alpha\cdot L} + \kappa_1\cdot\kappa_2 \\ -2\cdot\sqrt{(1-\kappa_1)\cdot(1-\kappa_2)\cdot\kappa_1\cdot\kappa_2}\cdot e^{-\Delta\alpha\cdot L}\cos(\Delta\beta\cdot L) \end{array} \right] \tag{3.6}$$

where

$$\Gamma = (1-Tap)\cdot(1-\gamma_{Tap})\cdot(1-\gamma_1)\cdot(1-\gamma_2)\cdot e^{-2\cdot\alpha\cdot L} \tag{3.7}$$

and

$$\Delta\beta = \frac{2\cdot\pi}{\lambda_0}\cdot\Delta n \tag{3.8}$$

P_{IN}, P_{BAR} and P_{CROSS} are optical powers at the input, bar and cross output respectively, Tap is the optical portion of the power needed for the tap operation, γ_{Tap}, γ_1 and γ_2, are the excess losses of each MMI coupler, κ_1 and κ_2 are the coupling ratios, L is the waveguide length, α is the attenuation of the waveguide in $Neppers/m$, $\Delta\alpha$ is the attenuation increment due to the variation of the refraction index in $Neppers/m$, $\Delta\beta$ is the change of the propagation constant due to refractive index change, Δn, and λ_0 is the vacuum wavelength.

When the refractive index is modified, an increment of the attenuation loss is expected. The increment of attenuation simulated is accorded to the equation given in [LJ01] for free-carrier-absorption when current injection is used to change the refractive index.

The necessary Δn for optical switching is obtained by solving eq. 3.5 when the bar output is 1, considering that there is no loss when the MMI is switched is

$$\Delta n = \frac{\lambda_0}{2 \cdot L} \tag{3.9}$$

The obtained refraction index variation necessary for switching a $100\mu m$ length waveguide MZI is $\Delta n = -7.75 \cdot 10^{-3}$. Simulations have been made with a 6% tap, 0.1 dB extinction losses for each MMI, coupling ratios $\kappa_1 = \kappa_2 = 0.5$, $100\mu m$ waveguide length (L), and $1.55\mu m$ vacuum wavelength.

Output powers of the Ta2S switch as a function of refractive index variations at ideal (lossless) and real case are shown in Fig. 3.8 (a). Excess losses about 0.7dB and 0.55dB are obtained for bar and cross outputs, respectively, when they are active. Insertion losses are about 3.7dB when switch is in (50:50) configuration. A fabrication tolerance error of 0.1% in both MMI coupling ratios, κ_1 and κ_2 has also been considered in simulations. A detail of the bar output in active state is shown in Fig. 3.8 (b).

Once the insertion losses for the different states of the Ta2S switch have been modeled, we can give the expression for loss in the complete 2-STCM module (see Fig. 3.6) as:

$$L_{2-STCM} = PL_{Ta2S} + 10 \cdot \log_{10} m + (\log_2 P + 1) \cdot PL_{sw2\times1/2\times2} \tag{3.10}$$

and

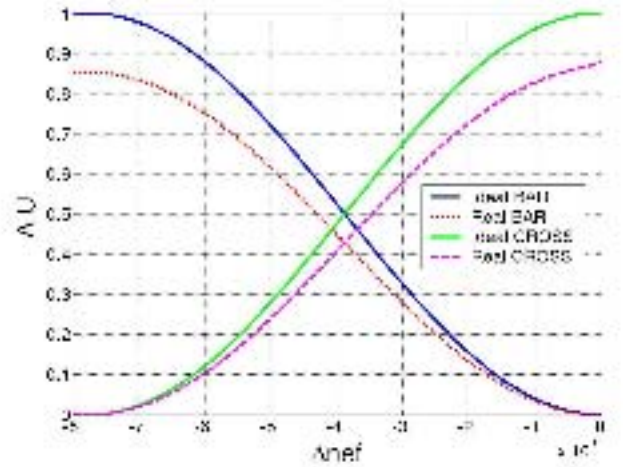
$$PL_{Ta2S} = PL_{tap} + \begin{cases} PL_{cross} \\ PL_{bar} \\ PL_{2-split} \end{cases} \tag{3.11}$$

Where PL_{Ta2S} is the power loss when the signal traverses a Ta2S, P is the number of output ports and m is the split factor (1 or 2), $PL_{sw2\times1/2\times2}$ is the power loss incurred by an optical 1×2 or 2×2 switching element. PL_{cross} and PL_{bar} are the losses in cross or bar states and $PL_{2-split}$ is the excess loss when performing 2-splitting.

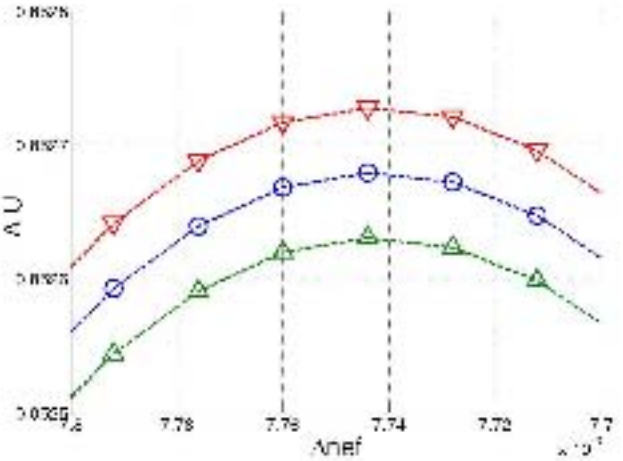
3.2.3.5 Comparison of number of components and power loss

In Tab. 3.2 and in Fig. 3.9 (a) we can see that our proposal scales conveniently with respect to SaD and TaC modules. The number of components (optical switches, Ta2S switches, tunable 2×2 MMI splitters and tap devices) in 2-STCM is much lower than the other proposals, which contributes to improve the power efficiency.

We shall make an overall assessment of power loss in a network by means of a concrete example. Let us assume that 51 dB (see [Rou02]) is the sample power budget, i.e.



(a) Cross and bar outputs of the node vs. refraction index changes.



(b) Difference in bar output power at $\Delta n = -7.75 \cdot 10^{-3}$ when 0.1% error in κ is considered.

Figure 3.8: Simulation results for a MZI optical switch with fixed tap [Lal11].

Elements	SaD	TaC	ns-TaC	2-STCM
Switches $(2 \times 2/2 \times 1)$	P^2	$P^2 + 2P - 1$	P^2	$P \cdot \log_2 P + \frac{P}{2}$ [1]
MZI switches	-	-	-	P
Tunable 2×2 MMI splitters	$P \cdot (P-1)$	-	-	-
Tap devices	-	1	P	P

Table 3.2: Components used in internal switch modules

although ideally amplification can compensate loss, attenuation must not exceed the target power budget for the receiver to work properly. We have calculated the power loss incurred by a single optical signal when continue, tap-and-continue, 2-split, and tap-and-2-split actions are taken in these three modules for best, average and worst cases. For the purpose of comparison, no amplification is considered in any of the nodes.

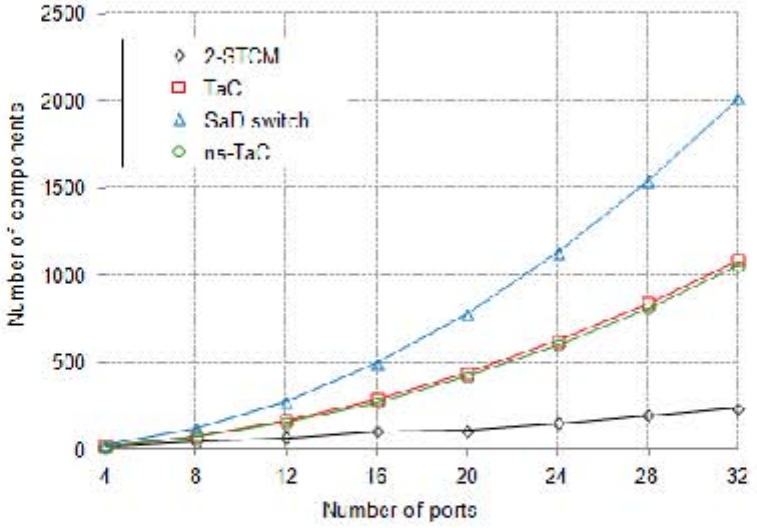
In order to estimate the power loss, we have considered that a single 1×2 or 2×1 optical switch has a power loss of 1.1 dB. TaC, SaD and 2-STCM have been compared for continue and tap-and-continue operations (in the case of SaD, it is a drop-and-continue action), and results are presented in Fig. 3.9 (b) for the average case. Fig. 3.9 (b) also depicts the power loss incurred by a *non-shared-tap TaC* (ns-TaC) node for the same setting. This architecture is introduced in the comparison to have also the cost of tap-sharing stripped off regular TaC [AD00a]. Note that this cost is not negligible as the need to re-switch the continue signal to an output port requires additional switching elements (see Fig. 2.2). This could be avoided by tapping each input as done in 2-STCM. Then, a ns-TaC would have P tap devices but just P^2 2×2 switches.

Fig. 3.9 shows that 2-STCM has a better scalability in terms of number of components and power efficiency for a TaC operation. Although TaC may use fewer components than SaD, it performs the worst in *power efficiency*. We also observed the same trend for the *continue* action. If optical amplification inside the node was considered, the 2-STC node is the most convenient option, because it would use much fewer amplifiers ($2P$) than SaD (P^2) but more than TaC (P). Since best, average, and worst cases are the same for switching in 2-STCM, losses may be known in advance and precise required amplification levels can be set up beforehand. In the SaD switch, power loss grows with the number of ports as $10 \cdot \log(m)$, whereas the loss in 2-STCM grows at a lower rate.

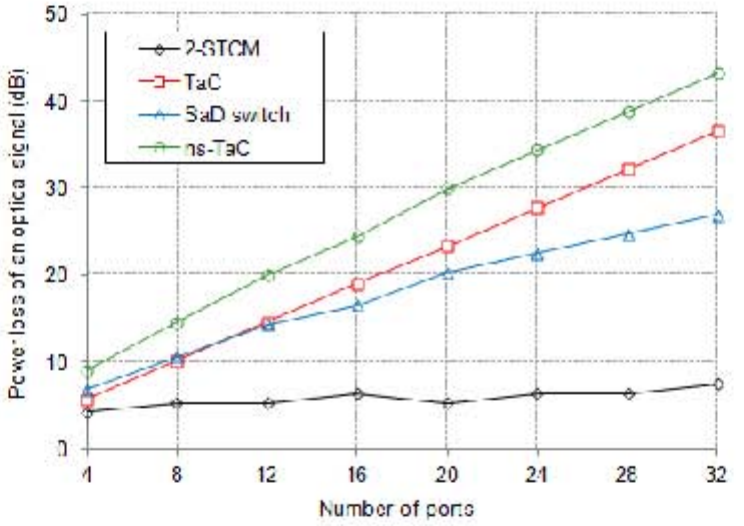
3.2.4 Feasibility in a Network Environment

In the previous subsection, it has been shown that 2-STC nodes present better power efficiency than the other architectures in tap–and-continue mode when implemented with integrated optics as described. However it should be clearly stated that globally TaC is more power-efficient than 2-STC and SaD when the nodes are set in split mode. The insertion loss due to splitting is far from that of tapping. The price that TaC pays is the lack of optimality in the delivery graph i.e. more link wavelengths are taken to reach all target nodes.

The 2-STC node proposed here is inspired on limitting the split fan-out to two branches, and combine it with optional tapping by means of integrated optics. This represents another variant of the MC-RWA problem. In this case, the target is building binary trees as balanced as possible in order to maximize the number of nodes that receive enough power for effective tapping. We claim that this approach



(a) Total number of components in 2-STCM, TaC and SaD modules.



(b) Power loss (in dB) of a single optical signal for 2-STCM, TaC and SaD switch modules when a tap/drop-and-continue action is performed (average case).

Figure 3.9: Comparison of architectures.

shows an ideal trade-off between simplicity of implementation (optimal by tapping in TaC) and transmission costs (optimal by splitting in SaD).

First of all, one major issue to be analyzed is whether the 2-split constraint of 2-STC is enough in practice to satisfy multicast requests in a wavelength-efficient way. Since most core network topologies have an average degree between 3 and 4, binary-split

should perform quite well. Therefore, we evaluate how well 2-STC nodes perform on a reference core network. We took several metrics and compared them under different multicast requests in a network topology made up of 28 nodes: the *COST-266 Optical Transport reference network*[2], shown in Fig. 3.10. Since some nodes of this network have more than 4 output links, we consider that all nodes have 8 ports.

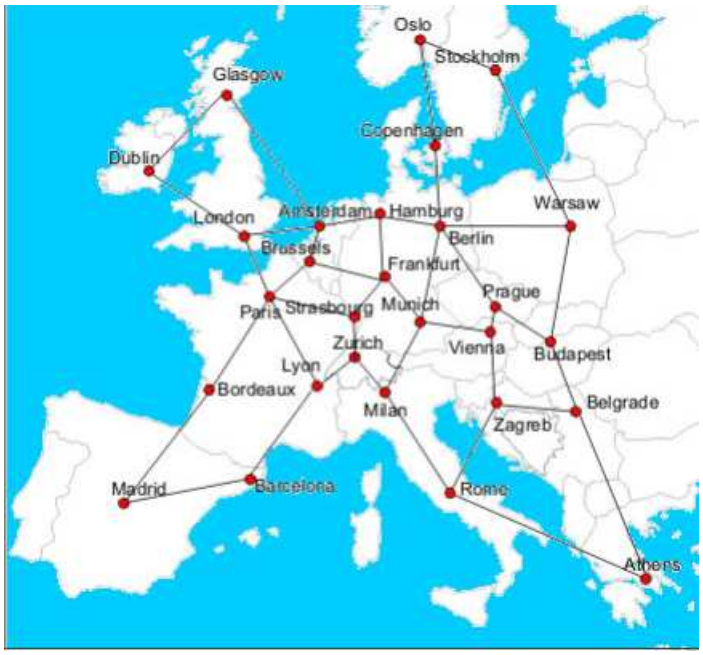


Figure 3.10: COST266 optical transport network

3.2.4.1 LightTree-making algorithms

In order to compare the three architectures under study, we need to implement tree-construction algorithms for each one. The three target problems are known to be NP-complete as shown in the references that follow. These problems are:

1. For SaD: Steiner tree with no degree constraint, i.e. the generic Steiner Tree Problem [HRW92].
2. For 2-STC: Degree-constrained Steiner tree [BV95], in particular the degree constraint is 4 (one input plus three outputs (tap and binary split)).
3. For TaC: A variant of the *open traveling salesman problem* [AD00a] where only a subset of nodes in the network needs to be visited and there is no need to return to the origin.

Since this work is not actually focused on the tree optimization and it is not practical to compute the optimal spanning tree solution even for medium-size networks, we

[2]http://www.ibcn.intec.ugent.be/ INTERNAL/NRS/index.html

shall use existing heuristics where available. Intensive simulations show that very simple heuristics can provide cost-effective solutions not far from the optimal [BV95]. Thus, for SaD we shall simply use the *shortest path tree* given by Dijkstra as a simple and realistic heuristics for tree construction. This is the type of tree that IP multicast routing protocols such as *multicast open shortest path first* (MOSPF) or *protocol independent multicast with dense mode* (PIM-DM) produce.

In [AD00a] the authors describe their target problem as the Multiple-Destination Minimum-Cost Trail (MDMCT) problem for their TaC architecture. Therefore, for TaC we shall use the algorithm proposed in [AD00a], which has been designed with the guarantee that no link is traversed more than once on a given direction. This is claimed by the authors to perform better than other approaches such as next-nearest node first.

Finally, for 2-STC we propose a heuristic algorithm inspired in the informal descriptions of [BV95] that removes the constraint of *no return to a previously visited node* of [BV95]. It should be noted that solving special cases with 2-STC and also with TaC, requires back-tracking. In this sense, the worst case example for both TaC and 2-STC is the star topology. With respect to SaD, broadcasting in an N-node star net requires N links, $N-2$ additional hops with TaC and $(N-3)/2$ additional hops using 2-STC (value for $N = 2^k - 1$, with k integer). In practice, core networks are usually 2-connected for resilience reasons, and backtracking over the same link could be prevented.

The main challenge for an algorithm able to work with 2-STC nodes is to convert a light-tree into a binary light-tree (as detailed in Algorithm 3.1), where every node can not have more than two out links. In the following description of the algorithm the Steiner tree for the multicast session is given as an input. Basically, this algorithm analyses nodes with more than two children and finds possible alternative paths in order to accomplish the 2-split constraint for the parent node. The path is chosen so as to try to get the shortest distance and avoiding other nodes to have more than two children. If more than one solution is obtained it is selected randomly. In the case that it is not possible to obtain any other path, a special round-trip path is established. To achieve this, the links with the shortest distance are chosen and a path of the type *childrenA-parent-childenB* is established.

Once the solution is found, the light-tree has to be recalculated (*recalculateTree* function in the algorithm), i.e. distances, children and links must be reconsidered with the new path-branch found.

3.2.4.2 Simulation methodology

We ran a series of simulation experiments in Matlab 7.0 over the reference topology in Fig. 3.10 [MCL+03], including physical distance between nodes. Each single experiment consisted of a multicast request made up of a random root and a random subset of multicast members. The simulator built the trees for SaD, TaC and

Algorithm 3.1 Algorithm that converts an Steiner tree to a binary tree (i.e. a tree with branches up to 2 at every node).

```
Function BinaryTree (V, D, u, T, d, pred)
begin
    Input: Set of nodes, V
        Set of terminals, D.
        Source node, u
        Steiner tree for session represented in T.
        List of shortest distances from source to every node, d.
        List of predecesors for every node in V, pred
    Output: Binary tree, T2
        Special list of predecesors, sPred
    sPred ← Ø
    nodesToReduceSplit ← {x/x ϵ T ∧ fanout(x) > 2}
    ∀ parent ϵ nodesToReduceSplit DO
        parent ← NextNode(nodesToReduceSplit)
        /* Get the next node nearest to the source */
        optionalLinks ← Ø
        ∀ childNode / IsParent(parent,childNode) DO
            ∀x ϵ V DO
                if e(parent,x) ϵ T AND
                  IsNotParent(x,childNode) AND
                  IsNotParent(childNode,x) AND
                  |{∀ c / IsParent(x,c)}| < 2
                    dist ← d(x) + distance(e(x,childNode))
                    link ← (x, childNode, dist, |{∀ h / IsParent(x,h)}|)
                    optionalLinks ← optionalLinks ⋃ link
                end if
            end ∀
        end ∀
        if optionalLinks← Ø then
            linksA ← ChooseNodesWithFewestChildren(optionalLinks)
            linksB ← ChooseNodesWithShortestDistance(linksA)
            linkChosen ← ChooseRandomly(linksB)
        else
            /* if there is not other possible link to childNode */
            shortestEdge ← e(parent, y) ϵ T / IsTheShortest(getDistance(parent,y))
            pred(childNode) ← y
            sPred(childNode) ← parent
            /* new special link: parent-y-parent-childNode */
        end if
    end ∀
    T2 ← recalculateTree(D, d, u, pred, linkChosen, sPred)
    return T2, sPred
end ∀
```

2-STC for each random request, using the algorithms described previously. The experiments were grouped into series of different densities of receivers in the network.

Thus, four separate experiment series corresponding to random multicast requests spanning 25%, 50%, 75% and 100% receivers in the network were executed. For each series, we measured:

- the average root-to-destinations distance,
- the average root-to-destinations number of links, and
- the average power per link, not having into account link attenuation.

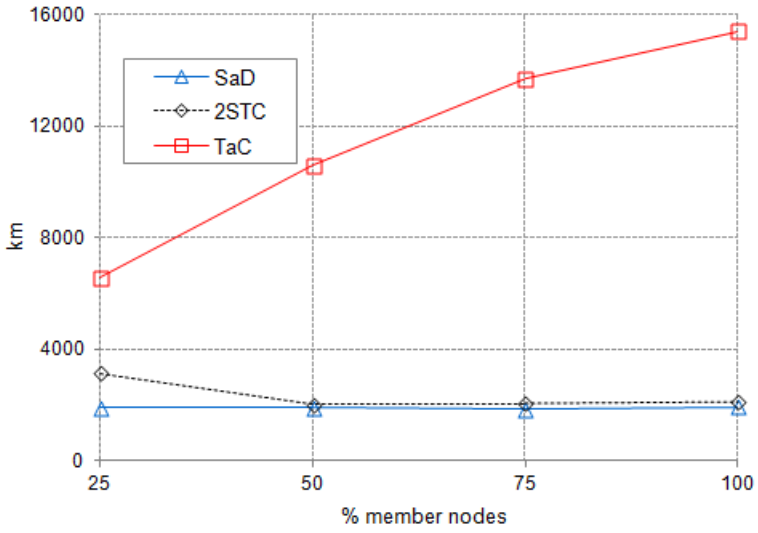
We wish to focus solely on the comparison of loss due by the different node architectures. Again, no amplification is considered for any of the architectures compared. The simulation was stopped when the target average X and confidence interval of 95%, $X \pm \Delta X$, held: $\Delta X/|X| \leq 5\%$. This interval is not represented in the graphics for the sake of clarity.

3.2.4.3 Source-Destination Distance and Link Consumption

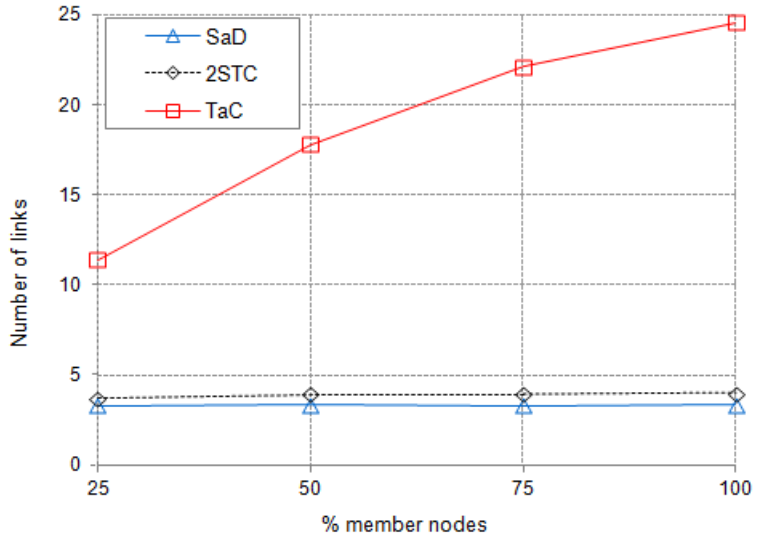
Fig. 3.11 shows the average *source-to-destination distance* and the average *source-to-destination link consumption* resulting from the simulation for TaC, 2-STC and SaD. In a TaC-node network, the resulting light-tree should actually be a single lightpath that traverses the minimum possible number of nodes to pass through all destination nodes, where a TaC operation is performed. Therefore, the distance and absolute link use of TaC is well above 2-STC and SaD. The graph also shows that the more dense the tree is, the longer the average path in number of hops and in distance. Thus TaC seems not to be a good option for broadcasting in a big meshed network in terms of delay and resource consumption. Some improvement to this result could be expected from better heuristics to the open traveling salesman problem, although always performing worse than a tree. The picture also reveals that the degree constraint to multipoint of 2-STC makes almost no difference to SaD in this topology, which was expected to perform better than 2-STC, even though some nodes have 5 links. In other words, the resource consumption is almost the same with binary light-trees. The result confirms previous works [BV95] on experiments with thousands of random topologies, where the authors conclude that the importance of unconstrained multipoint capability (both in terms of % of node support and maximum fan-out) is usually overestimated.

3.2.4.4 Power

A key parameter in the evaluation of the architectures under study is the average power that arrives at each node. As regards tapping vs. splitting, TaC must have much better performance than 2-STC and SaD. This effect is illustrated in Fig. 3.12 (a), where all attenuation due to switching inside the node has been omitted. Even though the path to run by TaC and the number of hops is larger than in the other approaches, the hard loss due to splitting is predominant.



(a) Average root-to-destinations distance.



(b) Average root-to-destinations number of links.

Figure 3.11: Comparison of variables of length.

However, ns-TaC and TaC need a $P \times P$ switching fabric whose attenuation is very relevant (above 4dB in average for a 4×4 switch), given the extra nodes to be traversed w.r.t. SaD and 2-STC (see previous subsection). Furthermore, TaC has an additional module to re-switch the *continue* signal. In contrast, SaD and 2-STC have a more power-efficient mesh of optical switches after splitting. This fact leaves

TaC in a situation quite comparable to the splitting-based approaches.

The results from simulation in Fig. 3.12 show that it is exactly the case in our scenario. Given its required number of hops and attenuation due to switching, TaC gives the lowest average power per link. If tapping is not shared, the ns-TaC case, then the average power is over that of SaD, which implicitly means a lesser need for amplification. 2-STC performs slightly better in this scenario. This improvement is amplified by the fact that 2-STC has a similar link consumption as SaD in practice. Fig. 3.12 (b) shows this effect.

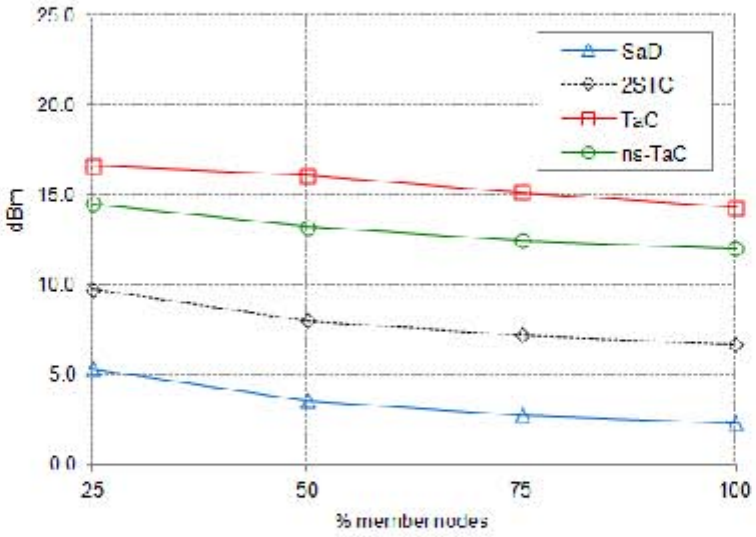
Another reference measure is the available optical power at links, which is depicted in Fig. 3.12. As it is expected, the 2-STC node allows to save more power, followed by the SaD node. This behavior means that, in average, any measure of the optical strength made in any part of the network would give those results regarding the optical strength, in average.

It should be remarked again the importance of a good heuristic for the tree construction in TaC and of the topology. Smaller networks and topologies more favorable to TaC, like ring or multiple-ring-based topologies, are not expected to show such a difference. Regarding SaD vs 2-STC, it is clear that for the reference integrated optics implementation that we have used, 2-STC outperforms SaD in power-efficiency and the extra link consumption of 2-STC seems not to be relevant.

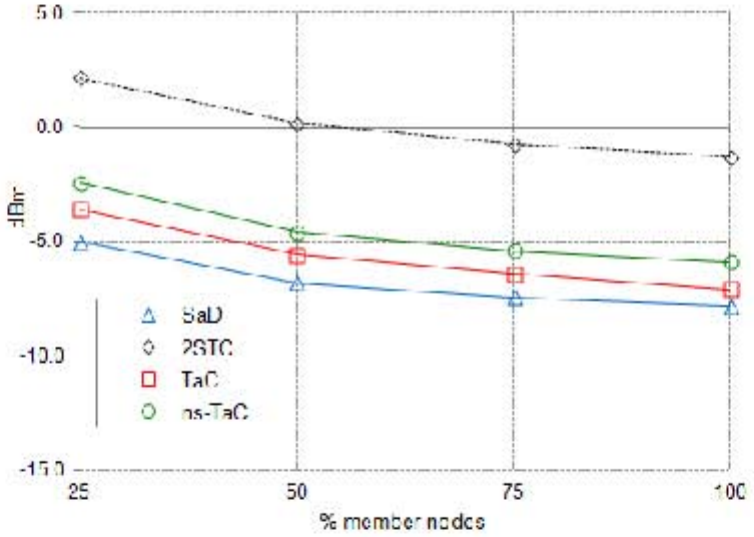
3.3 Extension of 2-STC Node to Support Traffic Grooming

In this section we present the architecture design of the *2-Split-Tap-and-Continue Traffic Grooming* optical multicast node, named from now on as 2-STCg node. This extends the capabilities of the *2-Split-Tap-and-Continue* (2-STC) node, introduced in the previous section: a multicast capable node with a simple structure, constrained to binary-splitting, but efficient regarding the use of power levels and the number of elements used. The extension consists of adding the ability to perform multicast routing tasks featuring *traffic grooming* (i.e. sub-wavelength demands on a single light-tree). In comparison to other state-of-the-art nodes, the resulting architecture allows to reduce the optical power attenuation (in the same proportion the 2-STC node does), bandwidth waste and delay.

The realization of traffic grooming should be made by taking advantage of the delivery light-trees already configured at intermediate core networks. It means that the sub-wavelength demands must be groomed into the light-trees without changing the switching configuration at any node. Let us suppose the example depicted in Fig. 3.14, in which there is a light-tree in a given wavelength already configured at nodes A, B, C, D, E and G, having node A as the source. There are 4 different sub-λ demands that might be groomed into the light-tree, assuming that every sub-demand



(a) Average incoming power at nodes without intern node attenuation.



(b) Average incoming power at nodes with intern node attenuation.

Figure 3.12: Average incoming power at nodes with and without intern node attenuation.

needs only half of the wavelength capacity at each link (i.e. each wavelength-link could fit 2 sub-demands). It is important to note that after the accommodation of demands 1 - 4, the light-tree is fully exploited, but the switching configuration at the optical nodes remains the same. Another important characteristic of this design

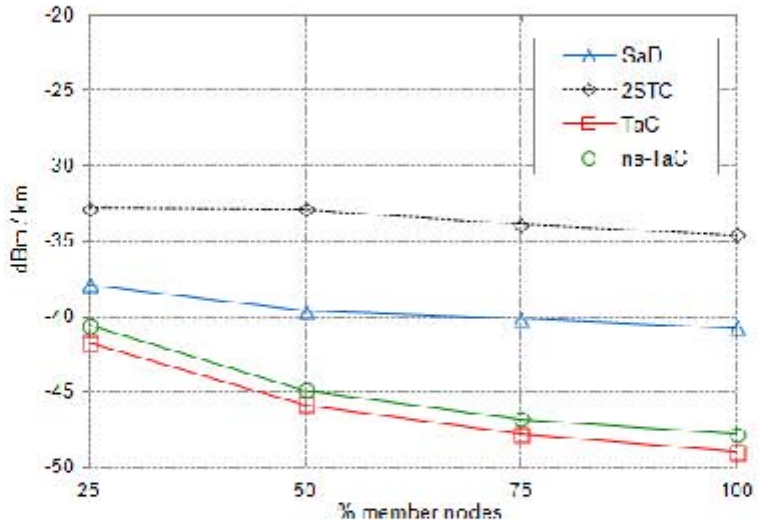


Figure 3.13: Ratio of average incoming power per kilometer (dBm/km).

is that it is not constrained to multipoint requests. The spare capacity of a light-tree can be employed in accomodating point-to-point (p2p) service requests.

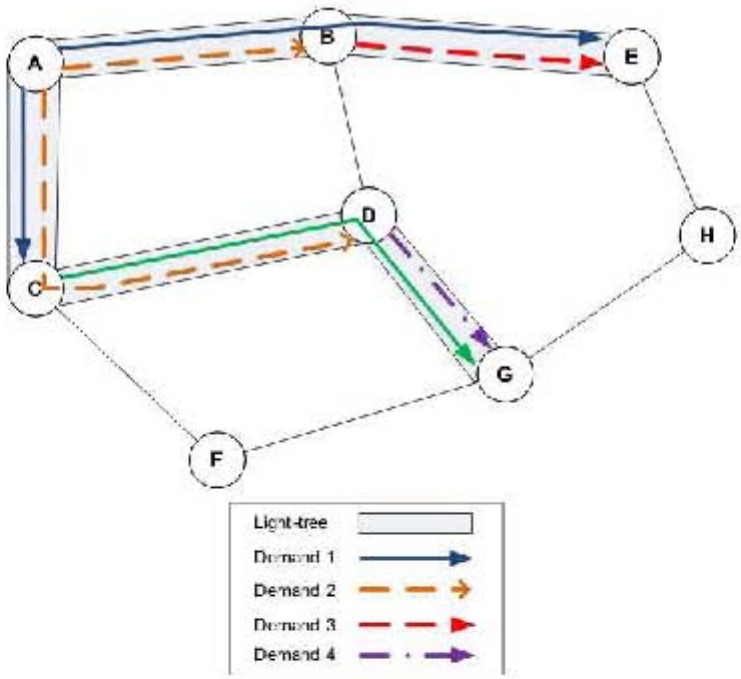


Figure 3.14: Traffic grooming of sub-λ demands into a light-tree.

3.3.1 Design Requirements

This, in addition to the 2-STC legacy features, depicted in Fig. 3.15 (a) and Fig. 3.15 (b) , in order to add the multicast traffic grooming features, the 2-STCg node needs to be able to realize also the following operations:

Only tap: This means that the 6% tap is the only optical signal dropped into the node (Fig. 3.15 (c)). The optical signal has to be deleted in order to not occupying any of the outputs.

Tap and add: This operation is indicated in Fig. 3.15 (d), and consists of freeing (i.e. getting empty) one of the outputs in order to add a local demand.

Tap and 2-add: This consists of deleting the signal for both outputs in order to add a duplicated local demand (Fig. 3.15 (e)).

Unlike any typical optical circuit switched network, we assume that the traffic arrives in *bursts* of variable length, in order to support traffic aggregation at multiple nodes throughout the light-tree. Each packet (burst) is labeled with a header indicating the multicast global destinations to which it is to be delivered.

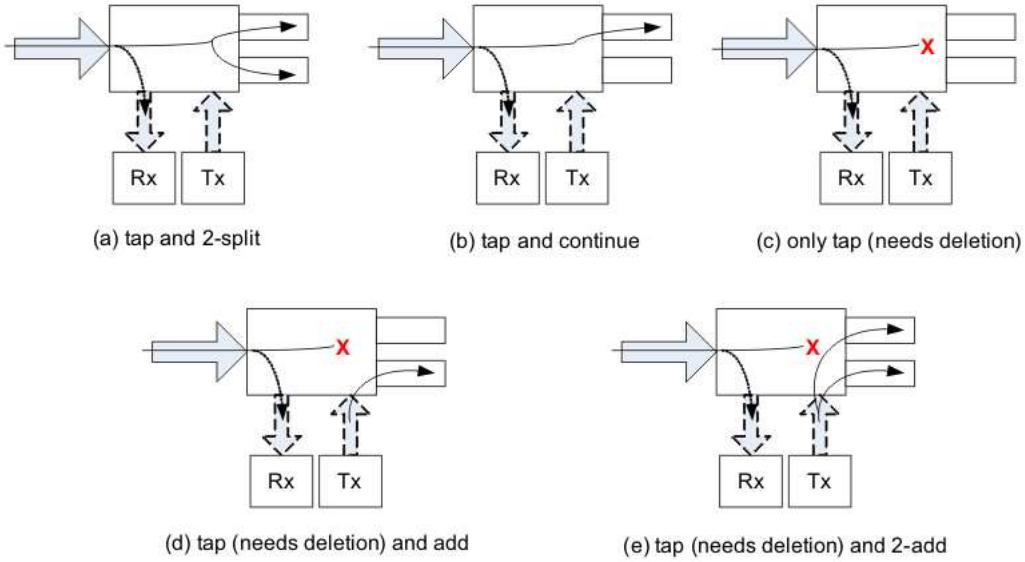


Figure 3.15: Operations required for the 2-STCg node.

3.3.2 Node Architecture

The 2-STCg general architecture proposed here is depicted in Fig. 3.16. As it can be observed, it has the same architecture as its 2-STC predecessor node (Fig. 3.1). At this level, both nodes have the same distribution, 2-STCg only differs from 2-STC on its main module (2-STCgM and 2-STCM, respectively).

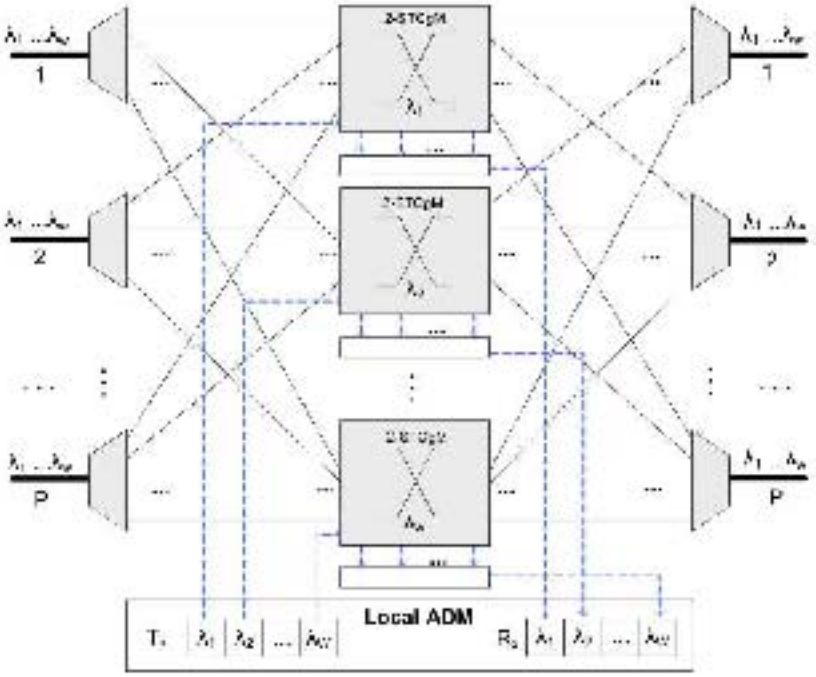


Figure 3.16: General architecture of a $P \times P$ 2-STCg OXC node.

3.3.3 The 2-STCg Module (2-STCgM)

The design of this module, for the case of 4×4 ports, is presented in Fig. 3.17. This component works with a given wavelength, and no conversions are allowed. Note that the 2-STCgM is constructed in a very similar way to 2-STCM detailed in sec. 3.2.2. It is assumed that there are light-trees already configured and it is possible to add traffic only in the available blank transmission spaces, therefore the permanent switching block has to change the switching positions only when the light-tree needs to be reconfigured. That is why it is made up of optical switches that are not intended to be fast.

Let us suppose that an optical signal comes into the port 1. After the arrival, 6% of the optical power is tapped and sent to a photodetector, while the remaining 94% of the optical power is directed to a *fiber delay line* (FDL). After the detection (once in the electrical domain) the label (which is put in the header of the burst) is read

and interpreted, and looked up in a forwarding table. According to the forwarding result, the *controller* sends the corresponding instructions to a set of devices, as depicted. If the forwarding information indicates that the input signal has to be splitted, the controller sends the splitting instruction to the upper 2-split switch. Then, the two resulting signals follow toward their respective1 $\times$ 2 switch - 2 $\times$ 1 couplers (called from now on SC-RRs) made with *optical ring resonators* (ORR). Finally the SC-RRs (also handled by the controler device) let the incoming signal pass to the switching block, which has been configured previously as a part of the light tree.

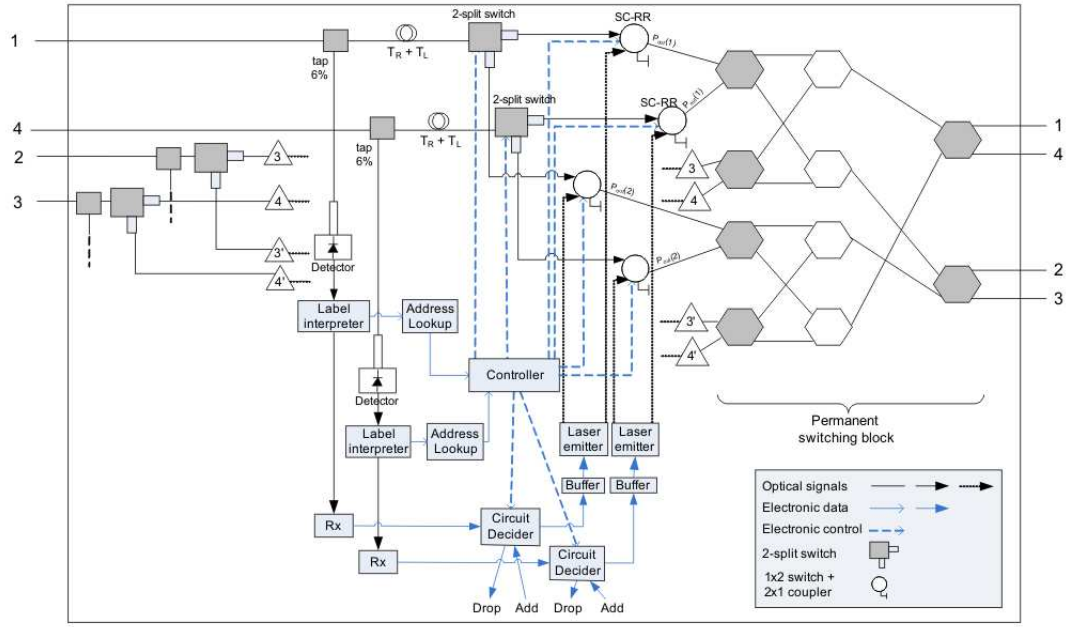


Figure 3.17: Design of the 2-STCg module (2-STCg-M) with 4 $\times$ 4 ports.

The internal *fiber and integrated optics* design of the 2-split switch and the SC-RR (1 $\times$ 2 *switch* + 2 $\times$ 1 *coupler* made with optical ring resonators) could be designed in a similar manner to the Ta2S implementation (3.3a). The 2-split switch could be a MZI able to switch the incoming optical signal to outs P_{BAR}, P_{CROSS} or both. After that, the SC-RR can let the signal pass to P_{out} or delete it, in which case another optical signal could be added $P_{out}(1)$. This design, similar to 2-STCM and Ta2S, should give reasonable attenuation results.

3.3.4 Switching Operation and Discussion

For a better explanation of the grooming mechanism, Fig. 3.18, Fig. 3.19 and Fig. 3.20 are provided. Let us suppose that there is a light-tree configured from node A to nodes C and D, going through node B (which is the local node represented in those

figures). In a first moment burst S1 is served, then S2 and S3 simultaneously, and finally S4 and S5 (also at the same moment). Subdemands are actually sequences of packet bursts that arrive at the input port 1 of the local node (node B). Note that although the design is fully asynchronous, the best performance can be achieved if bursts are sent in a periodic deterministic way.

At the arrival of a packet burst, a tap of the optical signal (6% of the power) is used to read the label (header) of the packet (label interpreter). Once it is recognized and a label lookup is performed, the controller sends the adequate signals to the corresponding device to realize the neccesary operations. The permanent switching block (or stage) is not changed for burst switching, but only the devices before that stage, that are designed to perform fast switching.

Fig. 3.18 represents S1 that is split and then forwarded to output ports 2 and 3. Note that since S1 occupies all the light-tree no other subdemand can be groomed until it finishes its transimission. After that, S2 is switched to output port 2, but since there is an available space for transmiting from the local node to node D, sub-demand S3 is served (Fig. 3.19). Finally, since S4 has to be dropped only to the local node, the S5 packet burst can be duplicated and switched to C and D nodes (Fig. 3.20). Note that for all these operations, some deletions of the optical signal had to be performed.

3.3.5 Bandwidth Savings

Traffic grooming was proposed to increase the bandwidth used, taking full advantage of light-paths and light-trees. It is supposed that when working with nodes capable of traffic grooming, they use fewer wavelengths to cover all the multicast demands. Thus, in this subsection we introduce the simulation results for the comparison of 2-STC (a multicast 2-split node not capablo of grooming) against 2-STCg.

As a preliminar advance (since further simulations are proposed as a future work), we ran a series of simulation experiments in Matlab 7.0 over NSFNET network topology. Physical distance between nodes was considered. Each experiment consisted of a multicast request made up of a random root node and a random subset of multicast member nodes. The simulator built the trees for 2-STCg and 2-STC nodes for each random request, using a 2-split constraint adaptation of the Dijkstra algorithm very similar to Algorithm 3.1. The experiments were grouped into series of different densities of destinations in the network. Therefore, four separate experimental series corresponding to random multicast requests spanning 25%, 50%, 75% and 100% receivers in the network were executed. We measured the average number of wavelengths used by any light-tree. The simulation was stopped when the target average X and confidence interval of 95%, $X \pm \Delta X$, held: $\Delta X/|X| \leq 5\%$. This interval is not represented in Fig. 3.21 for the sake of clarity. As expected, the number of wavelengths occupied using the proposed 2-STCg node (with traffic grooming features) is smaller than in the case of the 2-STC node (with lambda granularity).

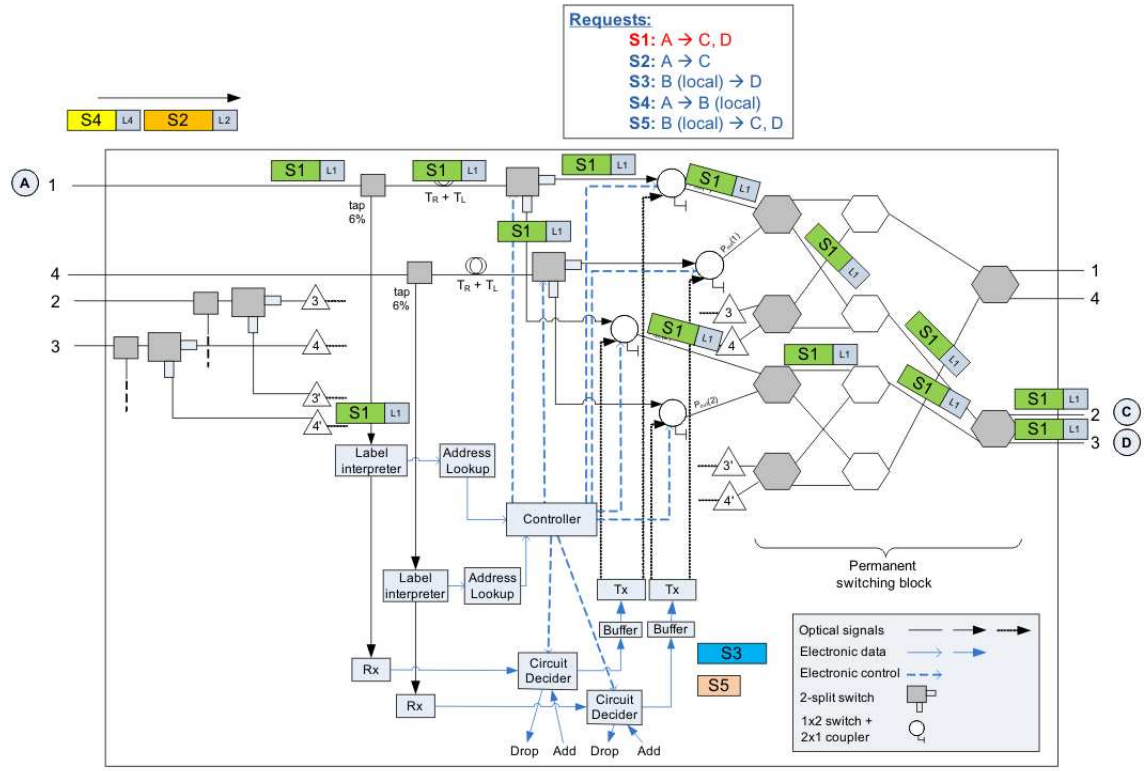


Figure 3.18: Example: S1 burst is switched.

3.4 Conclusion

In this chapter we have proposed the 2-STC, an all-optical multicast capable node with the 2-split constraint for saving the optical signal strength. The architecture is shown to be more simple, uses a fewer number of components and reduces the levels of optical attenuation, in comparison with the other state-of-the-art multicast capable OXC nodes. It has been also shown that the 2-split constraint is quite enough to attend multicast demands in actual network topologies, which have nodes mostly of degree 2, 3 and 4.

2-STCg was proposed as well, introduced as the extension of 2-STC for becoming it capable of traffic grooming. This design keeps the low complexity and the reduced number of components of the original 2-STC design and also allows to save the general number of wavelengths used to serve the multicast demands, taking full advantage of the light-trees, as shown in the simulation results. 2-STC and 2-STCg have been proposed to be implemented with integrated optics technology.

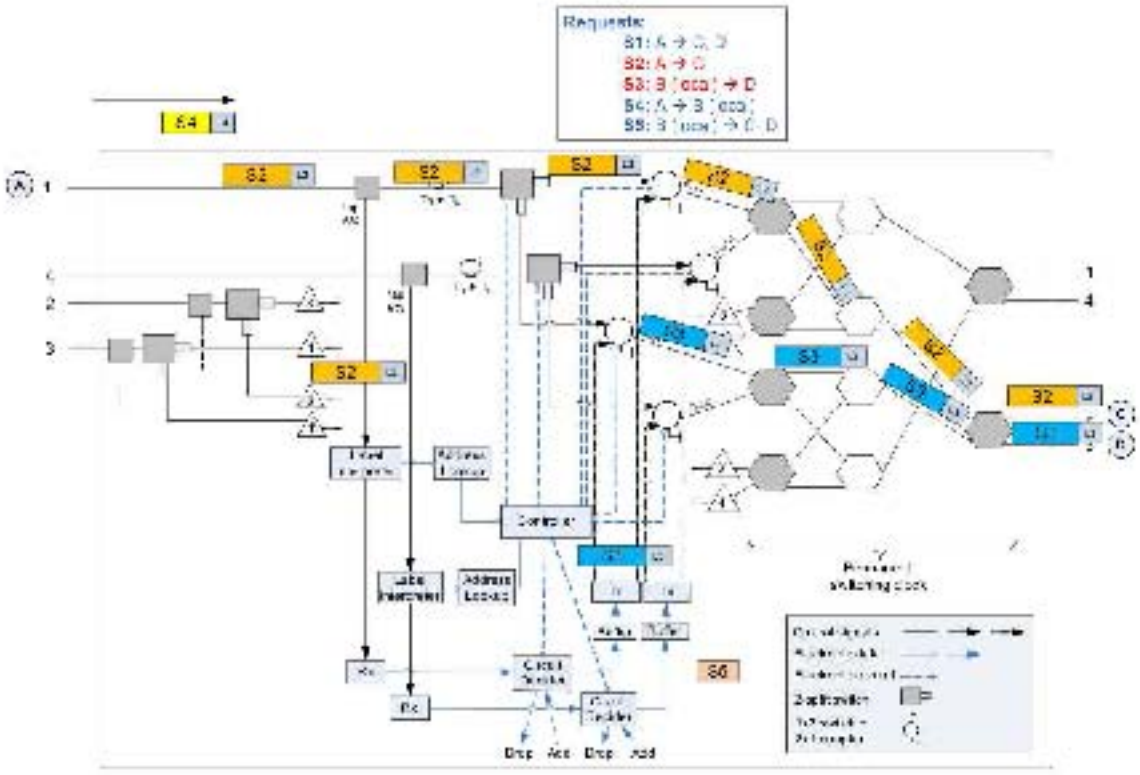


Figure 3.19: Example: S2 and S3 bursts are switched.

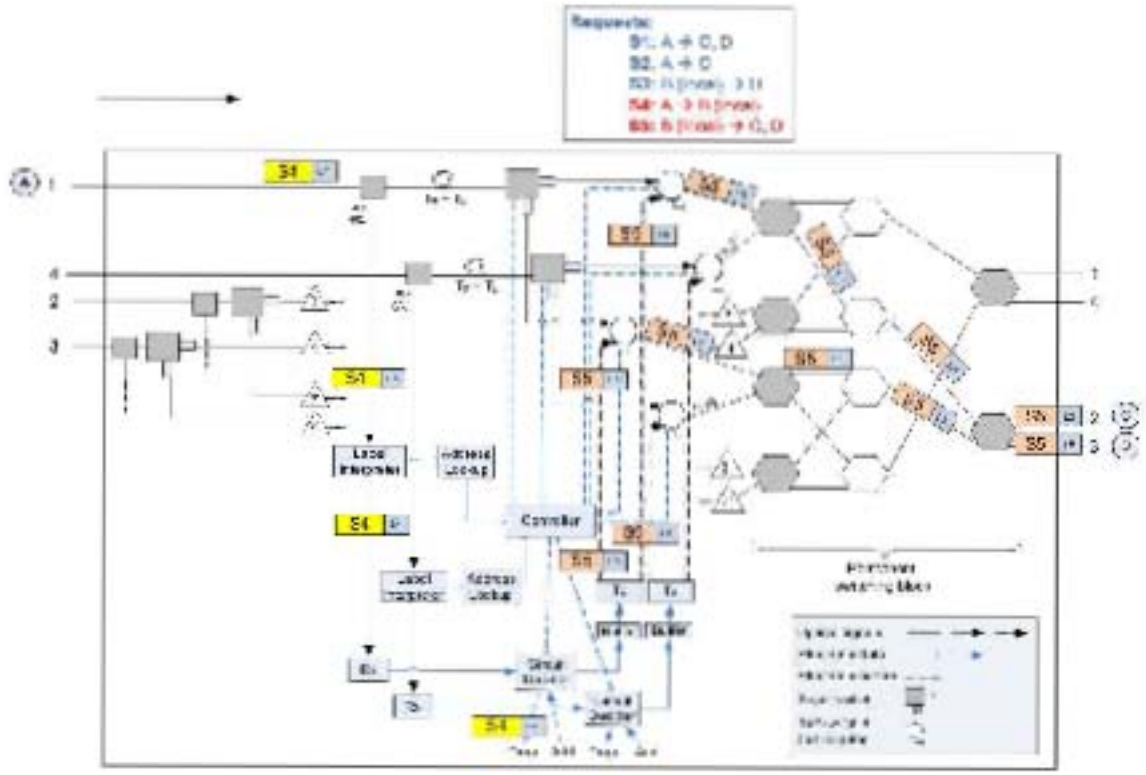


Figure 3.20: Example: S4 and S5 bursts are switched.

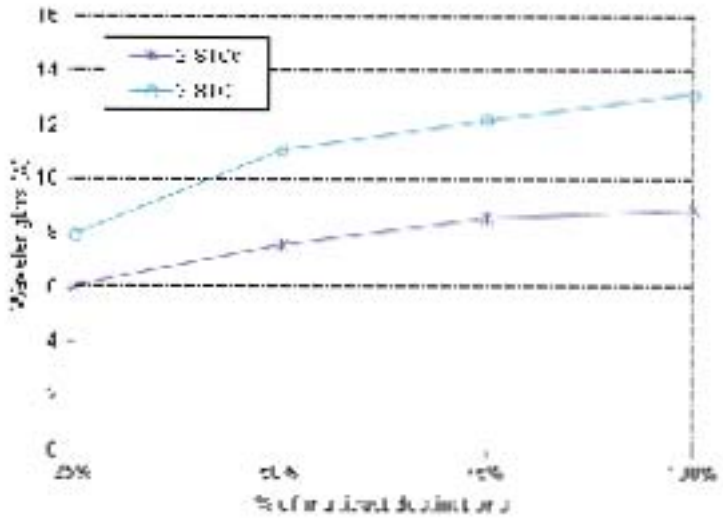


Figure 3.21: Average number of wavelenghts used by a light-tree using 2-STC and 2-STCg nodes.

4 Improving Scalability of Multicast in Packet-Switched Networks

As it was explained in sec. 2.2, one of the most important difficulties for multicast delivery in packet-switched networks is the control of the state information. In this chapter, we propose a set of approaches that improve the efficiency of the state-of-the-art techniques. In sec. 4.1 we study the bandwidth-state trade-off of aggregation trees and propose novel techniques for the construction of shared trees; in sec. 4.2 we analyze and propose a new Bloom filter-based architecture for source-routed multicasting that is almost stateless (i.e. the forwarding state information is bounded up to a really small amount of entries); and finally, in sec. 4.3 we combine both paradigms (aggregation and Bloom filters) in order to improve the preceding technologies.

Network Scenario of the Problem

Although the concepts of the following proposed approaches are applicable to diverse packet-switched scenarios in which the multicast service needs to be enabled, we will use the same network scenario described in [MYLS07, ZJS+10], which is depicted in Fig. 4.1, since this is the use case that has rought the multicst problem back to the research forefront. A BGP MPLS VPN-based network [RR06] of a service provider (SP) is made up of provider routers (P) in the core and provider edge PE routers (PE) interfacing with customer edge (CE) devices (a router or switch located at user premises). For unicast or multicast demands, once the path or tree is calculated, packets (in the case of L3VPN), or Ethernet frames (in the case of L2VPN) are labeled by the ingress PE and sent out over point-to-point or point-to-multipoint label switched paths (LSP) [RA12], and the forwarding of packets is performed by swapping labels at each hop, irrespective of its content. Thus, the VPNs specific information is carried through the network core between the involved PEs, but intermediate nodes do not need to read it. In L3VPN, for each VPN, provider edges have to keep a Virtual Routing and Forwarding table (VRF) with the corresponding VPN routing information.

As explained in sec. 2.2.1, scalable multicast VPN provisioning in this scenario, using the multipoint capability of core nodes, is not feasible. In the unicast case it is made scalable by label stacking: Packets from different VPNs share the forwarding entries in the core; therefore no VPN-specific information needs to be stored in core nodes. However, in the multicast case, label stacking does not solve the problem per-se: Each VPN tree would require a specific non-shared forwarding tree in the core, which

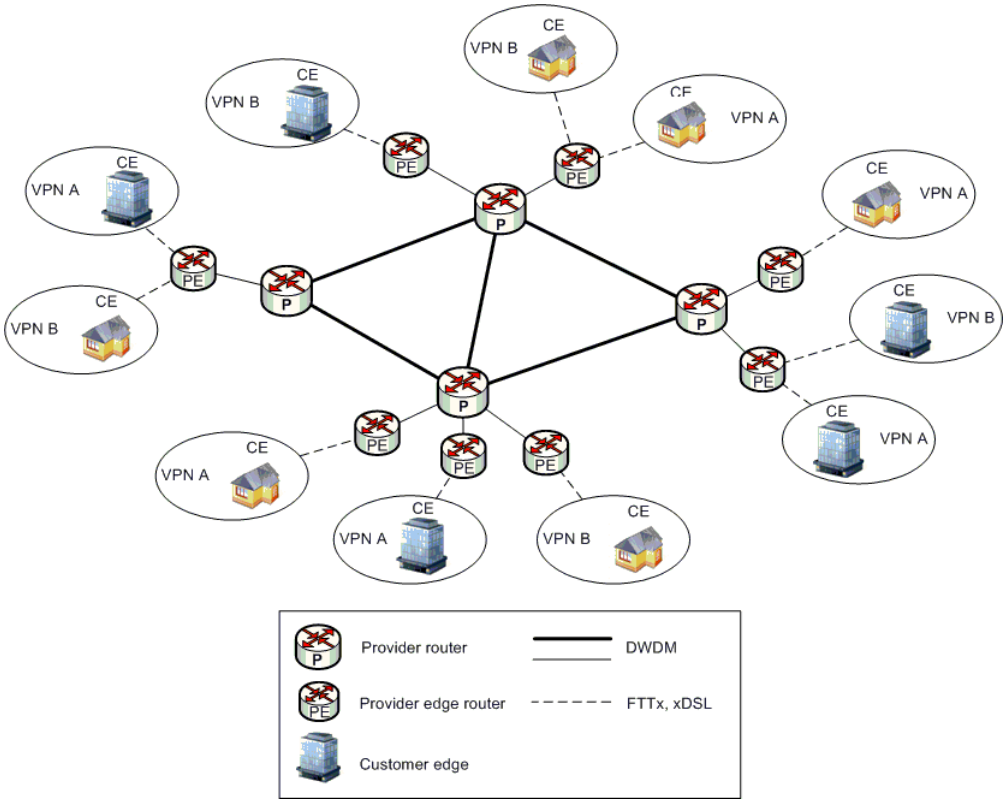


Figure 4.1: An MPLS VPN-based network

is not a valid solution. On the other hand, the currently deployed alternative -ingress replication- does not take advantage of multipoint in the core; so that we focus our research on new alternatives for a bandwidth and state-efficient implmentation of multicast.

4.1 Multicast Aggregation Approaches

As it has been introduced in sec. 2.2.1, the state information of multicast trees might be reduced by using distribution trees, a.k.a. as *shared trees* (STs). At this point, a logical question is raised: what are the quantitative effects of the aggregation? What is more, how can aggregation be measured and modeled to achieve a fair performance trade-off? In this section we present the scenario and some heuristics in order to approach to these issues. The objective of the work presented in this section is to gain insight on thresholds and bounds in which it is convenient to apply multicast aggregation. For this, we consider the *inclusive aggregation tree* model (explained previously in sec. 2.2.1), with the aim to reduce the state data and make a clear *bandwidth vs. state* comparison. In this sense, we also consider a centralized

data control plane, which has the knowledge of the network topology, routing and VPNs.

Regarding the construction of STs, *provider edge nodes* (PEs) will send the multicast traffic to a certain *rendesvouz point* (RP) node, because source-rooted trees provide more reliability and lower delay, but they increase the number of state forwarding entries. Therefore, every ST has its own RP for its own set of *multicast VPNs* (MVPNs). The RP selected is the one that in average causes de lowest delay (in number of hops) to the PEs that take part in the MVPN. This may be achieved by calculating distances (in number of hops) with the Dijkstra algorithm. We discarded optimal solutions to get the optimal RP, because of their NP-completeness condition.

We consider that the requests arrive to the core network dynamically and sequentially, and a number of STs are available for including them on the fly. Therefore, an incoming MVPN is assigned to one of the STs, and if the selected ST is not able to *include* it (some links are missing), the ST needs to *grow* in order to do so. Note that the selection mechanism is very important; if an MVPN is assigned to a ST that will cause much bandwidth waste, i.e. transmission of packets over links that do not lead to MVPN members (overhead packets), the whole network performance is affected.

4.1.1 Aggregation Techniques

For a clear explanation of aggregation approaches proposed here, let us consider the following variables:

u: Number of multicast VPNs

s: Number of shared trees

$V = \{v_1, v_2, ..., v_u\}$: Set of multicast VPNs

$T = \{t_1, t_2, ..., t_s\}$: Set of shared trees

$V(t_j) = \{v_1^j, v_2^j, ..., v_P^j\}$: Subset of multicast VPNs aggregated to t_j

$E(v_i)$: Set of edges of an exclusive tree for attending v_i

$E(t_j)$: Set of edges of t_j

$PE(v_i)$: Subset of PEs that participate in v_i

$PE(t_j)$: Subset of PEs that participate in t_j

$w(v_i, t_j)$: Bandwidth wasted (overhead) after aggregating the traffic of v_i to t_j

Besides, STs are built for aggregating a given group of multicast VPNs, depending on the aggregation ratio (AR), which is defined as:

$$AR = 100\left(1 - \frac{u}{s}\right) \tag{4.1}$$

From Eq. 4.1, $AR \approx 100$ ($AR = 100\%$ only when $u/s = \infty$) means that only one ST is used for all MVPNs, and $AR = 0\%$ means that there is a unique ST assigned to every MVPN.

In the first three following aggregation mechanisms proposed, exclusive STs are built for the first s MVPNs, using heuristics based on the Dijkstra algorithm. So, the first s MVPNs start having their own single ST after the first round. ST's RPs are selected by following the procedure described previously. Aggregation mechanisms differ in how they assign MVPNs to the existing set of STs.

4.1.1.1 Non-Intelligent Aggregation (NIA)

In this mechanism, after the creation of $t_1, t_2, \ldots, t_s$, the remaining $v_{s+1}, v_{s+2}, \ldots, v_u$ are aggregated to the s trees already built by following a round robin mechanism:

$$\textbf{if } (j = i \,\%\, s) \Rightarrow v_i \textbf{ is agregated to } t_j$$

As it can be noted, in this way MVPNs are aggregated without considering the overhead packets generated because of aggregations.

In the case $E(v_i) \subseteq E(t_j)$ (i.e. the set of links of v_j are included in the set of links of t_j), the incoming v_i is aggregated straight into t_j. In the case $E(v_i) \supset E(t_j)$, t_j has to grow for including all the set of links of v_i into the set of links of t_j.

4.1.1.2 Intelligent Aggregation (IA)

Here, $v_{s+1}, v_{s+2}, \ldots, v_u$ are aggregated to the STs looking forward to reducing the waste of bandwidth $w(v_i, t_k)$ and trying to fit v_i into the most similar t_k available, in the following manner:

$$w(v_i, t_k) = min\,(w(v_i, t_1),\, w(v_i, t_2), ...,\, w(v_i, t_N) \tag{4.2}$$

where $w(v_i, t_j) = |V(t_j)| \cdot (E(v_i) \triangle E(t_j))$. Thus, v_i is aggregated to t_k. The idea here is to get the best matching ST into which to aggregate the incoming MVPN.

As in the former technique, if $E(v_i) \subseteq E(t_k)$, the incoming v_i is aggregated straight into t_k, but if $E(v_i) \supset E(t_j)$, t_k has to be extended in order to include $E(v_i)$ into $E(t_k)$.

4.1.1.3 Intelligent Aggregation with Reconfiguration (IA+R)

This technique makes exactly the same aggregation procedure as the IA enhanced with reconfiguration of trees. Considering that $t_1, t_2, \ldots, t_s$ would keep growing up after some aggregation operations, they will probably increase the average RP-PEs

distance, the total number of hops, etc. This may happen since they were built taken into account only the topologies of the s first incoming MVPNs $v_1, v_2, \ldots, v_s$. After aggregating some other MVPNs, the topology of these ATs may need to be reconfigured. Therefore, in order to reshape them, they are reconfigured every y incoming MVPNs, where $y = \lceil log_2 (u/s) \rceil$. This reconfiguration implies to find the best RP and the shortest tree.

It has to be pointed out that y has been determined experimentally during simulations. Given that the number of MVPNs that any t_j should have in average is

$$\frac{u}{s} \approx \frac{\sum |V(t_j)|}{s} \tag{4.3}$$

The logarithmic form of y makes this value significative even for small amounts of the u/s factor.

4.1.1.4 Ad-Hoc Tree Aggregation (HTA)

Although the former aggregation technique seems to be an adequate effort for saving bandwidth, it is possible that a special group of MVPNs would cause a tremendous waste of bandwidth (overhead packes). This is the case in which MVPNs have only a few PEs with Customer Edges (CE) with an MVPN site attached. This phenomenon was observed during the simulations, especially with random samples with a Zipf distribution, where most VPN samples have a few members, and only very few VPN samples have a large amount of members (as we will explain in the next section). Why Zipf distribution for group or VPN sizes? Because actually the number of VPN sites of -e.g. a bank in a city- is expected to be proportional to the number of inhabitants in that city for a fixed per-capita income, and the distribution of population per city in the world is known to tend to a Zipf distribution. It can be easily noted that the main drawback about this is that, the smaller the aggregated MVPNs are, the more leaky (useless) traffic will be delivered to PEs that are not part of those MVPNs.

In order to avoid this, the fourth aggregation technique is proposed, in which small STs are provided for a certain group of small MVPNs (s-MVPNs). As shown in the example of Fig. 4.2, if an s-MVPN with only 3 members was aggregated to a given shared tree, all the packets of this s-MVPN will be forwarded to a great amount of PE nodes uselessly. Therefore, an ad-hoc tree is built for it. Avoiding to aggregate s-MVPNs to large STs will let us reduce the overhead with a small increase of state information.

This method is represented in detail in Algorithm 4.1. The line 2 states that if the number of nodes that would be involved in the s-MVPN v_i is less than a certain

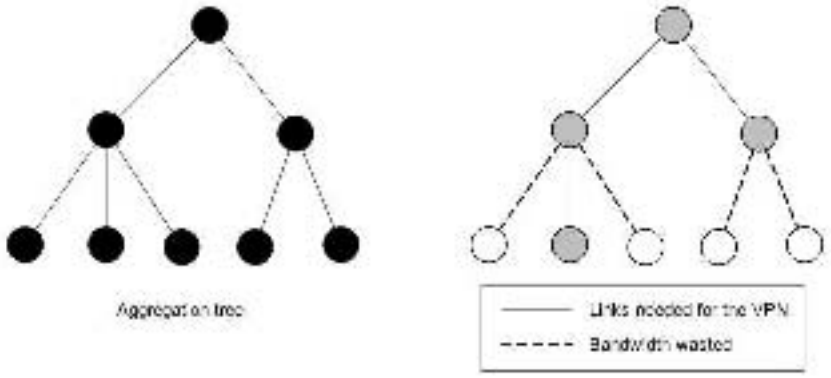


Figure 4.2: An small MVPN (s-MVPN) could waste many links

Algorithm 4.1 Pseudocode of the Ad-hoc Tree Aggregation (HTA)

```
1.  for each v_i ∈ V do
2.      if |E(v_i)| ≤ y then (where y is the upper bound)
3.          if T^s ∈ ∅then (where T^s is the set of small ad-hoc trees)
4.            find w(v_i, t^s_k) = min( w(v_i, t^s_1), ..., (v_i, t^s_q)) (where t^s_k ∈ T^s and q = |T^s|)
5.              if w(v_i, t^s_k) ≤ |v_i|(t^s_k saves bandwidth)
6.                  aggregate v_i to t^s_k
7.              end
8.          else
9.              build t^s_{q+1} ∈ T^s
10.             aggregate v_i to t^s_{q+1}
11.         end
12.     else
13.         apply the IA+R procedure for v_i
13.     end
14. end for each
```

threshold y, then v_i has to be aggregated to an *ad-hoc tree* (lines 3 - 11). In the case that there were previous *ad-hoc trees* built (line 3), v_i is aggregated to the *ad-hoc tree* that generates the lowest bandwidth waste (lines 4 - 7). If there were not any *ad-hoc tree* built previously, an *ad-hoc tree* is built for v_i, and v_i is aggregated to it (lines 8 - 11). Finally, lines 12 - 13 refer to the case in which v_i does not need an *ad-hoc tree*, therefore the IA+R technique is applied.

4.1.2 Simulations and Results

Extending the referred methodology presented in [MYLS07] -used for the analysis of aggregation in a generic topology-, major extensions have been developed in order to achieve more realistic simulations, and to observe the aggregation behavior. Firstly, we constructed a simulation model to work with real network topologies -listed in

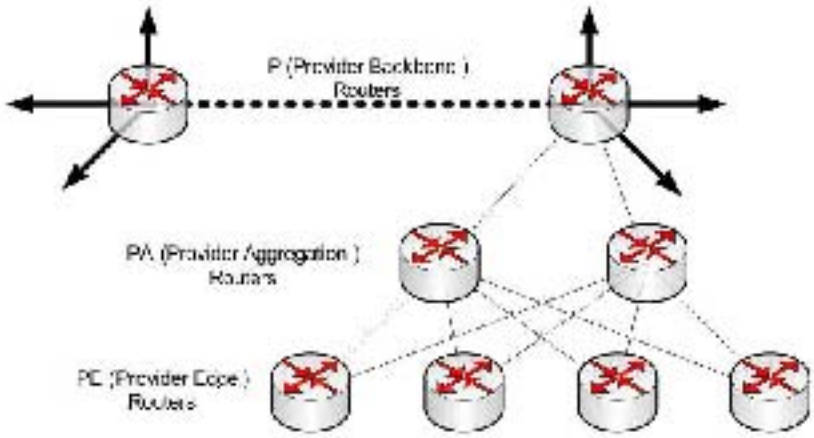


Figure 4.3: Provider aggregation model for each core node, which is actually made up of three levels of routers.

table 1-, with the aim to test the tree aggregation techniques in diverse (i.e. different number of core nodes and links, different topologies, etc.) and real scenarios. To complete the networks, we attached two PA routers (provider aggregation routers) to each core router, and the set of PE routers are connected to those both PA routers (see Fig. 4.3). The number of PEs at each core node has been considered as

$$PEs(x) = \left[\frac{\alpha(x) \cdot P(x)}{\beta}\right] \tag{4.4}$$

where x is the city/region where the core node is located, $\alpha(x)$ the number of Internet users in that city, $P(x)$ the number of inhabitants in x, and β the number of inhabitants of the least populated city from among all the cities that have a core node in the network. Tab. 4.1 summarizes data of the resulting topologies.

We generated random sets of MVPN samples with different distributions in terms of scope (i.e. number of provider backbone involved in the VPN) and density (i.e. number of PEs involved in the VPN) -as described in Tab. 4.2-, in order to observe the implications of these MVPNs in the behavior and efficiency of the aggregation techniques proposed. According to Tab. 4.2, it can be deduced that 12 possible combinations of VPN samples distributions have been generated. From these distributions, Zipf (for core and PE routers distribution) is probably the closest to reality.

Regarding the network topologies specifications, in a lower level, two PA routers (provider aggregation routers) are attached to each provider backbone router, and the set of PE routers attached to those both PA routers. PEs provide Internet and Layer 3 MPLS VPN services from these major locations, as proposed in [GFV05] (Fig. 4.3). PA nodes reduce the number of IGP (*interior gateway protocol*) adjacen-

Name	Core nodes	Core links	PE nodes	Total number of links
Abilene [a]	10	13	209	504
NSFNET [b]	14	20	107	282
KPN (Europe) [c]	39	52	421	1024
Tiscali (World) [d]	45	73	606	1448
COST-266 [e]	39	41	329	796

a. http://abilene.internet2.edu/

b. http://www.nsf.gov/

c. http://www.kpn.com/kpn/show/id=1561743

d. http//www.tiscali.net/

e. http://www.cse.buffalo.edu/~qiao/wobs/obs/papers/gauger_cost02.pdf

Table 4.1: Networks used in simulations

Type of Scope	Distribution of Provider Backbone Routers	Distribution of PE Routers
Fixed [a]	25%, 50%, 75%, 100%	Uniform or Zipf
Variable [b]	Uniform or Zipf	Uniform or Zipf

a. It means that all the VPNs involve x % of the core routers

b. It means that all the VPNs involve a random variable (not fixed) percentage of routers

Table 4.2: Distribution of VPNs among nodes for simulations

cies that have to be maintained by the backbone routers to two, because each core router has to peer only with two PA nodes (in addition to the other core routers in the backbone) instead of with all the PE routers attached to it, whose number can be fairly high. Each PE node may be connected to both PA routers via PoS (Packet over SONET) links.

Considering these scenarios, extensive simulations were run under Matlab 7.1, with 1,000 VPNs samples generated randomly according to the distributions explained before. In the results, snapshots of the different variables and metrics are taken, considering that every VPN has a source CE node already sending packets to the other members of the multicast VPN session. Results measure the impact of one packet per VPN sent to all its members. As pointed out before, five different core network topologies have been simulated (Tab. 4.1), in order to observe the behavior of the techniques implemented; however, for a better reading of this section, we only present the graphical results for the NSFNET reference network.

4.1.2.1 State Information

One of the results of a great interest is that of Fig. 4.4, which depicts the *number of multicast forwarding entries* that the set of core nodes need to keep to attend the VPNs randomly generated. It is important to note that, as the AR increases (towards to a single ST for all the MVPNs), this number is reduced, as expected.

At a first look, it is simple to observe that the use of unicast LSPs for attending multicast demandas is not a practical solution. Although it is better, the same problem is present when building one tree per each VPN ($AR = 0\%$). It is shown that aggregation of multicast VPNs into shared trees does contribute to decrease the alount of state information. Another conclusion is that results may change considerably with different distributions of VPNs: with uniform distribution samples of MVPNs (Fig. 4.4 (a)), the amount of forwarding entries is much higher than with Zipf distribution samples (Fig. 4.4 (b)). This difference is caused only by the size of the resulting STs (since with Zipf the majority of VPNs are small). Finally, we can see that, with the HTA technique, a higher number of *ad-hoc trees* must have been employed in the Zipf scenario for small VPNs (s-MVPNs), and this is why the number of entries are increased in Fig. 4.4 (b) for that technique.

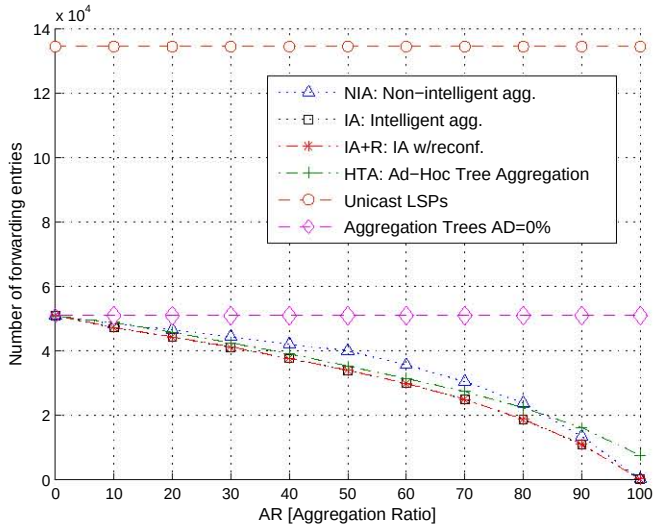
When using shared trees, there are two types of forwarding entries: Those related to the set of STs established in the network, and those of each MVPN attended at PE nodes. The number of forwarding entries of an ST related to the first case is equal to the number of provider backbone routers and PA routers that are part of that ST. These entries are shared for all MVPNs that are part of a given ST. Instead, entries stored at PE routers are specific for VPNs. In [FCGF01a] authors introduce the concept of *reducible* and *non-reducible state information*. When using STs, or even when using traditional IP multicast, terminal nodes necessarily need to keep state information related to the multicast traffic of VPNs, and that is why this state information is called *non-reducible state information*. On the contrary, the number of forwarding entries of distribution trees (shared trees) is variable and will depend on the *aggregation ratio* (AR) employed. The following metric is called *reducible state reduction ratio* ($RSRR$) and is given by

$$RSRR = 1 - \frac{\sum_{t_j \in T} S_{reducible}}{\sum_{v_i \in V} S^{AR=0\%}_{reducible}} \tag{4.5}$$

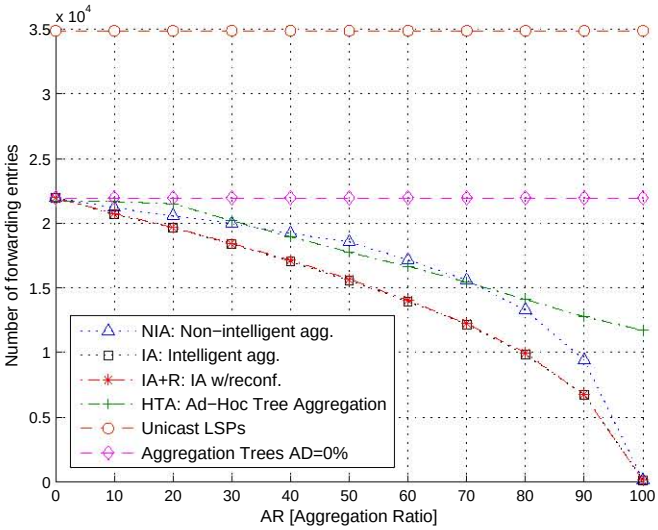
The Eq. 4.5 represents the ratio between the *reducible state* with a given aggregation technique, and the *reducible state* when building one tree per MVPN (when $AR = 0\%$). This is an effective way to monitor the amount of savings on the state information obtained by the different aggregation techniques (Fig. 4.5). As predictable, for higher values of AR (i.e. fewer STs are built), the savings on the state information grow, because with a few STs there is much less state information to maintain. In the case of the Zipf distribution of VPNs, shown in Fig. 4.5 (b), this savings are lower than in the case of uniform VPNs (part a of the figure), because more STs are supposed to be smaller.

4.1.2.2 Bandwidth Consumption and Forwarding Efficiency

The following results stand for the total *bandwidth* for attending the total amount of VPNs using a given aggregation technique. Let us suppose that a multicast VPN



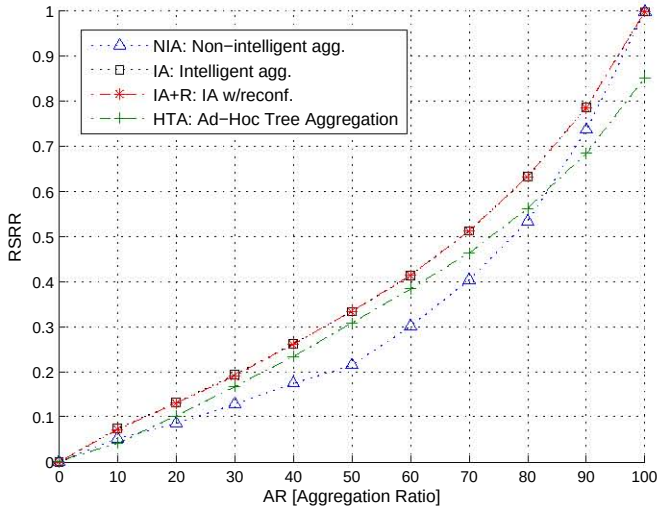
(a) Uniform VPN size distribution



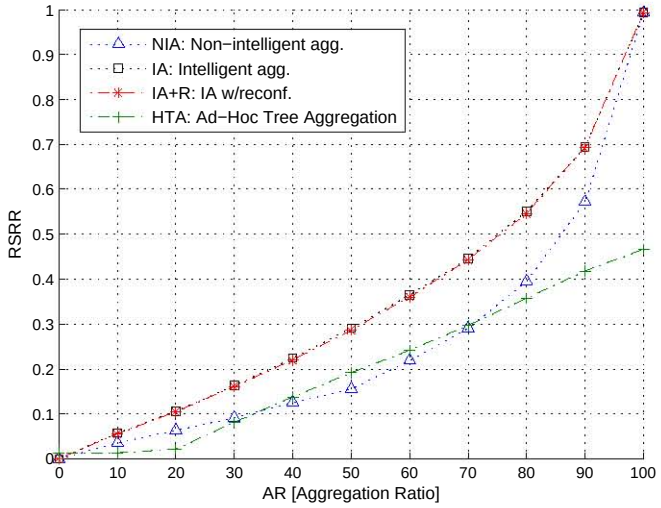
(b) Zipf VPN size distribution

Figure 4.4: Number of forwarding entries (NSFNET network).

packet is sent from a PE node; as long as it goes through the shared tree (assuming that it firstly needs to sent the packet to the RP of the shared tree) to which it is aggregated to, it is duplicated and forwarded through links of the ST. The resulting count of packets at each link is the *bandwidth consumption* generated by that VPN. As it is an aggregation approach, there is a number of links and PEs that receive useless/overhead packets.



(a) Uniform VPN size distribution



(b) Zipf VPN size distribution

Figure 4.5: Reducible state reduction ratio (RSRR) (NSFNET network).

In Fig. 4.6 it is shown that, naturally, NIA is the worst of the aggregation techniques (because it does not consider the bandwidth impact on STs selection), and IA and IA+R techniques perform as expected, with just an insignificant difference between them. HTA has the better performance here, mainly because it saves bandwidth by building ad-hoc shared trees for small VPNs. However, as shown before, HTA sacrifices bandwidth savings instead of state information. The graphic also shows the bandwidth consumption when $AR = 0\%$ (every MVPN has its own distribution

tree).

For the Zipf distribution of VPNs (part (b) of the figure), the use of unicast LSPs means a lower level of bandwidth used than the aggregation techniques NIA, IA and IA+R, when $AR \geq 60\%$. Note that for HTA it only happens when $AR \geq 90\%$, which gives the conclusion that the fact of having more small VPNs (as usual in the Zipf distribution model), it migh not be worth waste much bandwidth by sharing distribution trees. Instead of that, single LSPs, or ad-hoc trees (like in HTA technique) should be used.

As the number of STs decreases, there is a higher overhead penalty. In order to see this behavior in a more adequate manner, we define a metric named *multicast efficiency* (δ) which represents the improvement (when the value is positive) or degradation (when negative) achieved when aggregating traffic into STs against attending the same demands with unicast LSPs:

$$\delta = 1 - \frac{bw_{Agg}}{bw_{LSP}} \tag{4.6}$$

In Fig. 4.7 it can be seen that the *multicast efficiency* is greater than 0 (i.e. aggregation means an improvement with respect to unicast LSPs) when VPN sites are distributed uniformly (part (a) of the figure). For the Zipf distribution, as it was observed also in Fig. 4.6 (b), for a higher AR the results of the *multicast efficiency* means a degradation (there is no improvement with the aggregation).

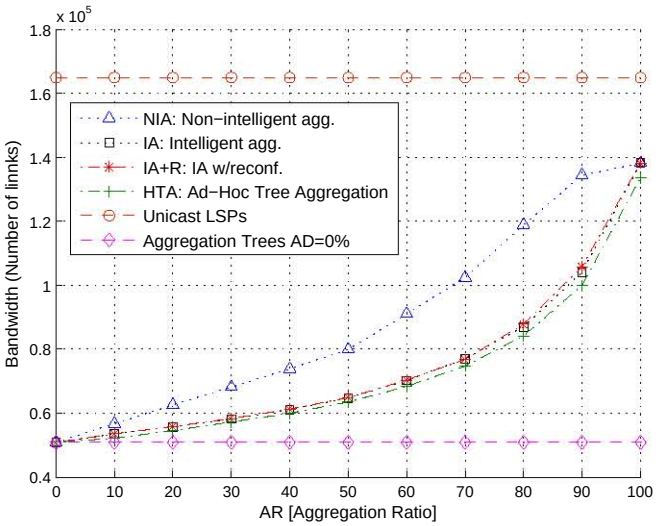
The following metrics is the *average ratio of useless bandwidth* (β), which rates the bandwidth wasted with the use of shared trees against the use of one tree per each VPN ($AR = 0\%$) (Fig. 4.8):

$$\beta = \frac{bw_{Agg}}{bw_{AR=0\%}} - 1 \tag{4.7}$$

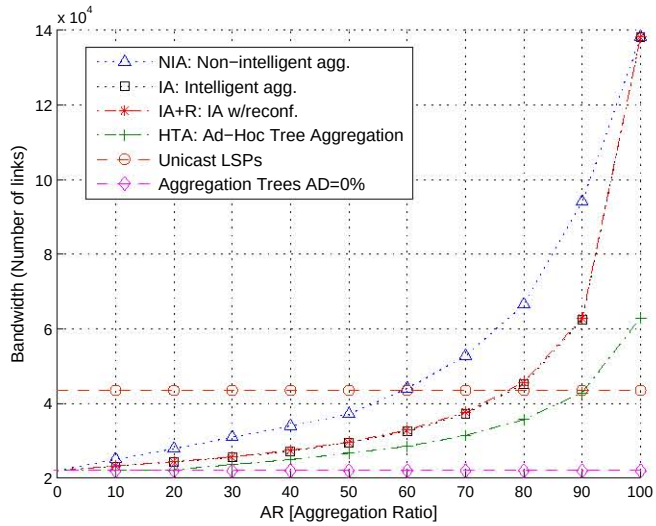
The last metrics to be analyzed is the *forwarding efficiency* (fwe), in order to quantify the performance of any of the methods. This metrics is intended for knowing what percentage of the bandwidth used is useful. For this, we adapt the formulation defined in [JZR+09], obtaining:

$$fwe = \frac{bw_{AR=0\%}}{bw_{Agg}} \tag{4.8}$$

Clearly, in Fig. 4.9, the more STs are employed for aggregation, less *forwarding efficiency* is achieved. Besides, it can be noticed that, in the case of the Zipf size distribution of VPNs (part (b) of the figure), this ratio is lower for most of the techniques, except for HTA, where ad-hoc trees are used for small MVPNs (that should be the most common in that type of distribution).



(a) Uniform VPN size distribution

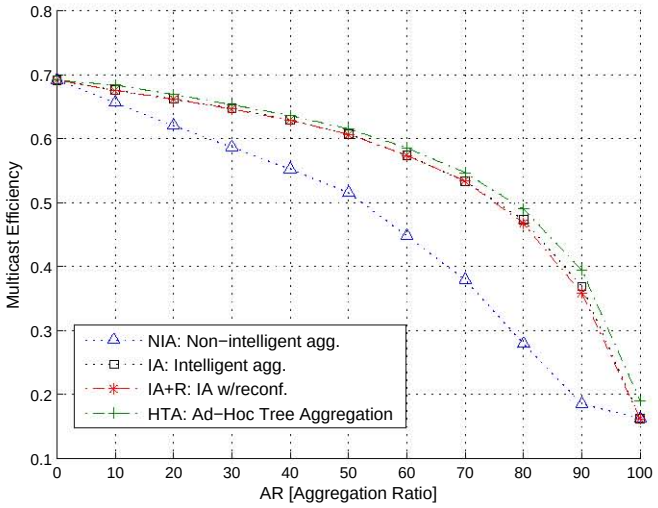


(b) Zipf VPN size distribution

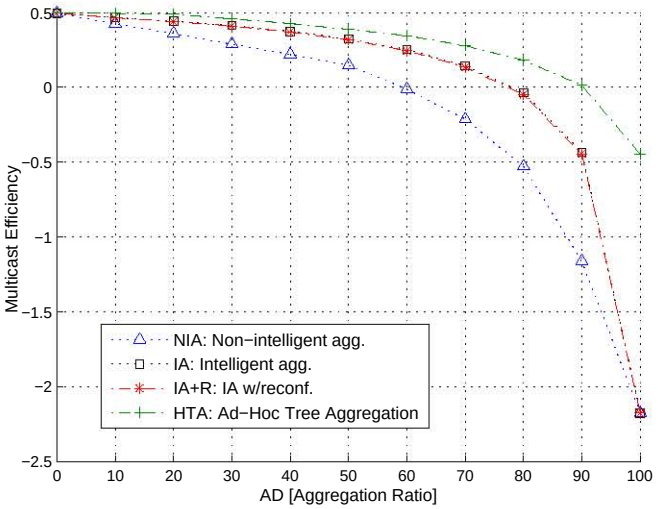
Figure 4.6: Total bandwidth consumption (NSFNET network).

4.1.3 Conclusion: Network Design and Deployment Issues

For network design and deployment purposes, the previous simulation results can be used to determine the range over which aggregation trees could be useful. It is clear, for example, that roughly under $AD = 80\%$, the creation of aggregation trees is useful, with a uniform distribution of VPNs to core nodes and PEs. If we assume that the most common VPN size distribution is Zipf, we can see that LSP unicast



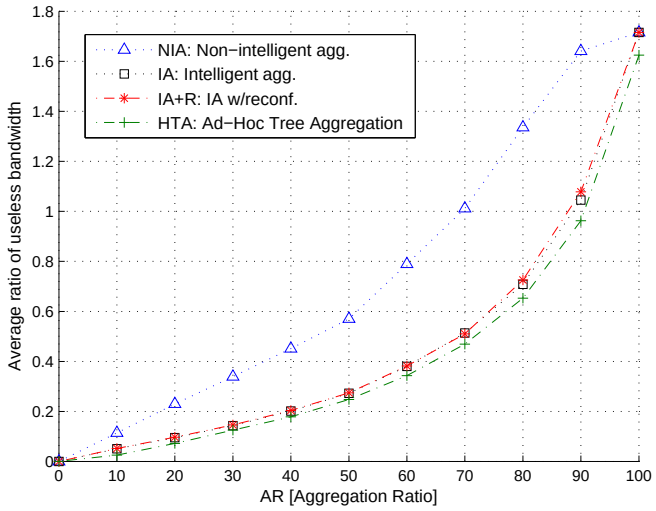
(a) Uniform VPN size distribution



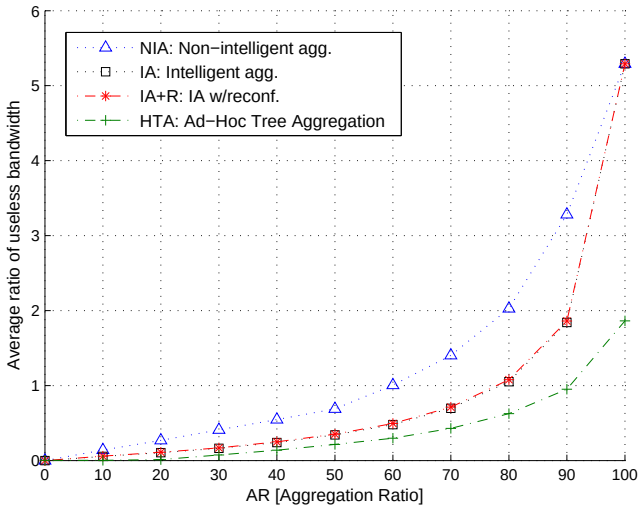
(b) Zipf VPN size distribution

Figure 4.7: Multicast efficiency (NSFNET network).

improves its performance. As shown before, it is specially in this case where the HTA technique could be very useful. By building ad-hoc small trees for small VPNs (situation very common with Zipf), it reduces the bandwidth waste by paying only a few more state forwarding entries. It should also be noticed that no consideration has been made about QoS. Aggregation strategies should take into account impact on current link loads before a new tree is included in an aggregation tree, like in [CFGF01].



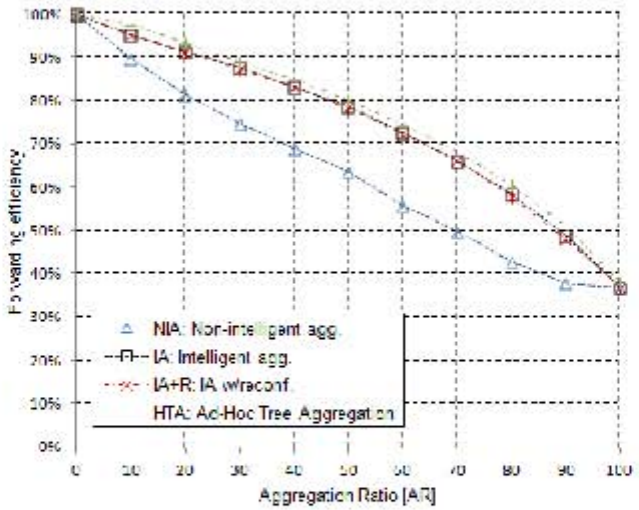
(a) Uniform VPN size distribution



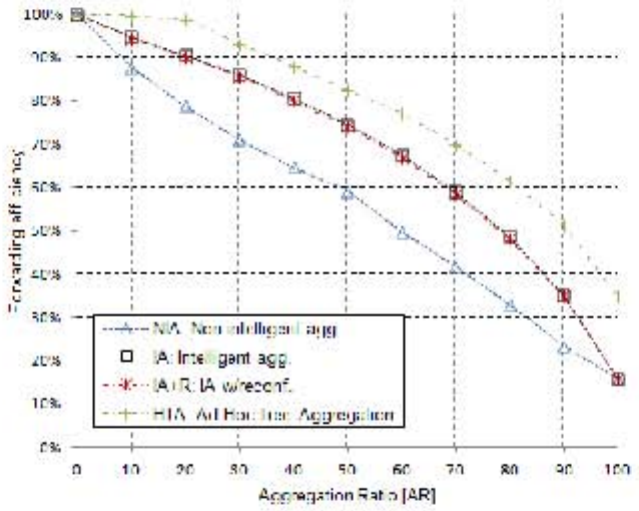
(b) Zipf VPN size distribution

Figure 4.8: Average ratio of useless bandwidth (NSFNET network).

In this section we have introduced the problem of aggregation of multicast demands into super trees, which may reduce the total state information at intermediate nodes but will -depending on the aggregation ratio- generate not negligible levels of bandwidth wasted. We have proposed a set of aggregation algorithms (techniques) and studied the trade-off between bandwidth wasted and state information, which could be useful for the design of further solutions and traffic engineering purposes.



(a) Uniform VPN size distribution



(b) Zipf VPN size distribution

Figure 4.9: Forwarding efficiency (NSFNET network).

4.2 Depth-Wise Multi-Protocol Stateless Switching (D-MPSS)

As referred in sec. 2.2.3, Bloom filters, proposed by B. H. Bloom in [Blo70], have been recently proposed as another alternative for multicast forwarding [RMM+11]. The most important feature of this method is its potential to reduce (and almost eliminate) the amount of state information at core nodes.

One of the most advanced techniques of the Bloom filter-based forwarding family (as explained in sec. 2.2.3) is MPSS, which combines the MPLS switching architecture with the *zFilter* forwarding. This method, in spite of resulting in important bandwidth savings and the practical elimination of the forwarding state information, presents forwarding anomalies (flow duplication, packet storms and forwarding loops) that need to be addressed. Besides, it needs large Bloom filters to encode the muticast trees, which may increase the packet header overhead up to not diminishable rates.

In this section we analyze and discuss the forwarding anomalies present in MPSS. Inspired in this technique, we propose a new method to encode multicast trees and improve the forwarding efficiency of traffic delivery, called Depth-Wise Multi-Protocol Stateless Switching (D-MPSS). The most important improvement is that it gets rid of forwarding anomalies, outperforms the forwarding efficiency of MPSS and reduces the amount of header overhad. This mechanism could be applicable for other types of networks where packet multicast delivery needs to be enabled.

4.2.1 A brief Analysis of MPSS

In order to support the formulations that follow in this section, let us consider the following set of variables.

m: Size of the Bloom filter array

n: Number of elements to be added to the Bloom filter

k: Number of hash functions ($k \geq 1$)

fpr: False positives rate of the Bloom filter

$\bar{\rho}$: Proportion of 0 bits after n elements are inserted

ρ: Fill factor (proportion of 1 bits of a Bloom filter)

d: Number of neighbors of a node (or outgoing links of a node)

v: Branch links of a node that are part of the multicast tree

h: Depth of a tree (maximum distance from root to leave nodes)

i: Level or depth of a node in the tree with respect to the root

N: Number of nodes of the network

oh: Overhead, i.e. number of packets wasted

ho: Heder overhead, i.e. bytes used by the packet headers

4.2.1.1 False Positives Rate and Fill Factor

A Bloom filter [Blo70] is an array $B = \{b_0, ..., b_{m-1}\}$ of m bits that represents a set of n elements $S = \{x_0, x_1, ..., x_{n-1}\}$. At the beginning B is filled with 0s. Each

item in the set S is mapped to B by setting to 1 k positions of B (k is the number of hash functions). A hash function maps the i-th element of S to an integer in the range $[0, m-1]$. After inserting an element x_i into the filter B, the probability that a given bit is not set to 1 by one of the hash functions, is

$$1 - \frac{1}{m} \tag{4.9}$$

Given that there are k hash functions, the probability that a bit is still zero after those k functions is

$$\left(1 - \frac{1}{m}\right)^k \tag{4.10}$$

After the insertion of n elements to the filter the probability for the same case is

$$\overline{\rho} = \left(1 - \frac{1}{m}\right)^{kn} \tag{4.11}$$

This can be approximated to

$$\overline{\rho} = e^{-kn/m} \tag{4.12}$$

Then, the probability that a bit is set to 1, which is equivalent to the *fill factor* (the proportion of 1 bits), is

$$\rho = 1 - \left(1 - \frac{1}{m}\right)^{kn} \tag{4.13}$$

Finally, when an element membership test is done, if all the k array positions in the filter computed by the hash functions are set to 1, it is claimed that the element belongs to the set. The probability of this ocurrence when the element is not part of the set (i.e. *false positive rate*), as presented in [Blo70, BM05], is

$$fpr = \rho^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k \tag{4.14}$$

Although this fpr formulation has been considered in several previous works regarding Bloom filters, Bose et al. [BGK+08] have shown that Eq. 4.14 is actually a lower bound. The problem is that it is based on the asumption that events "bit b_i is

set to 1" and "bit b_j is set to 1" are independent, which is not true. Thus, Eq. 4.14 gives false answers for small values of k, m and n. According to [BGK+08] the real fpr is larger than expected, especially for small values of m, providing the following exact formulation:

$$fpr_{exact} = \frac{1}{m^{k(n+1)}} \sum_{i=1}^{m} i^k i! \left(\begin{array}{c} m \\ i \end{array} \right) \left\{ \begin{array}{c} kn \\ i \end{array} \right\} \tag{4.15}$$

The difference between both formulations is

$$fpr < fpr_{exact} \leq fpr \times \left(1 + O\left(\frac{k}{\rho} \sqrt{\frac{\ln m - k \ln \rho}{m}} \right) \right) \tag{4.16}$$

Unfortunately, the formulation of Eq. 4.15 is impractical to be used in analytical calculations due to its complexity. Given that the error (Eq. 4.16) is negligible when $k \ll m$, which is the case in MPSS and in our proposal, it is possible to use Eq. 4.14 for analytical calculations, and from now on in this work it will be considered as the formulation of fpr.

In general, it can be observed that, for larger values of m, the fpr decreases, but with a larger set of elements to be inserted (n), it increases. That is why Bloom filter applications, like MPSS, require the use of large Bloom filter size (512 bits, 800 bits, etc) in order to reduce the *false positives rate*.

4.2.1.2 Overhead and Forwarding Efficiency

As reviewed in sec. 2.2.3, when a false positive occurs, it is possible that the *overhead* packet gives another false positive in the next hop. This effect is amplified when the packet is forwarded into other sub-trees or the packet comes back to one of the nodes previously visited, in which case a forwarding loop event takes place, causing a packet storm. These anomalies are not desirable at all and increase the wasted bandwidth, which we include under the concept of *overhead* of the method. For both cases, the insertion of a *time-to-live* (TTL) field into the packet header, assigned with the h value (depth of the tree) would limit the existence of packets, alleviating the impact of packet storms and infinite loops.

In order to estimate this overhead, let us consider a regular network. Let d denote the constant degree of all the nodes (their number of neighbors), let v be the number of the actual outgoing interfaces that are part of a given multicast tree, h the depth of the tree (the distance in number of hops from the root node to the leaf nodes), and i the depth of a node in the tree (distance in hops from the root to the node). In Fig. 4.10 it is shown an example of the behavior of overhead propagation in such a regular network. For example, the root node, in the first hop should generate $(d - v - 1) \cdot fpr$ of overhead, in the second hop $(d - v - 1) \cdot fpr \cdot (d - 1) \cdot fpr$, and

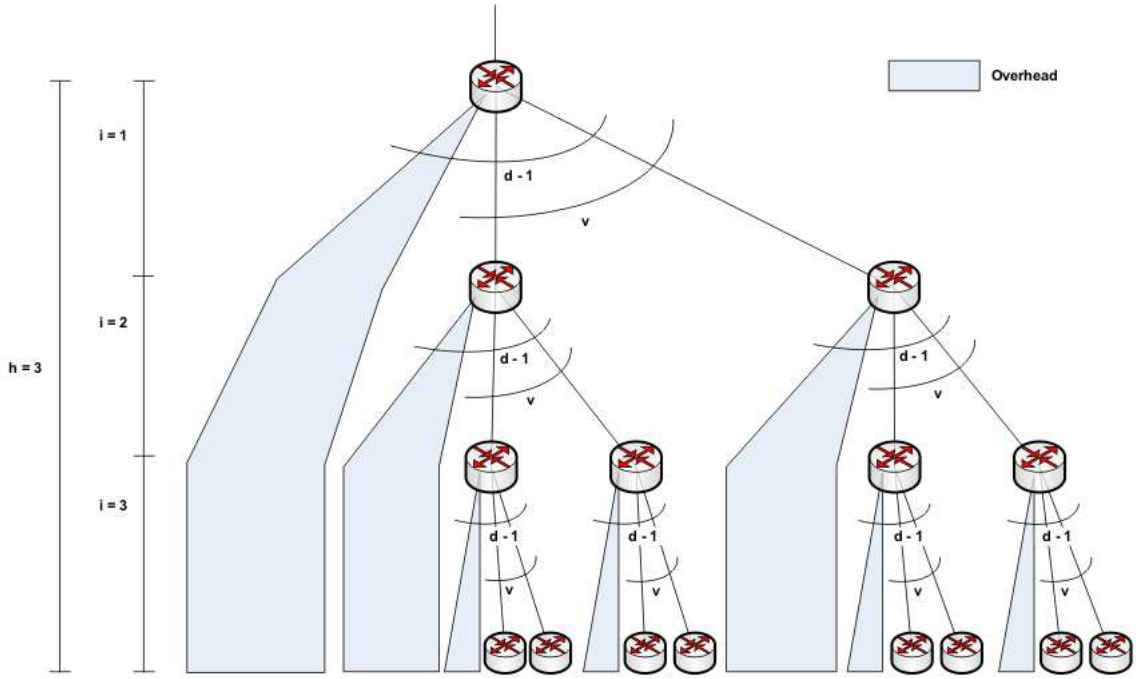


Figure 4.10: Overhead propagation of MPSS in a regular network, with $d \geq 4$, $v = 2$ and $h = 3$.

so on. This partial overhead corresponds to nodes with $i = 1$ (in this case, only the root), and the total overhead is the sum of all partial overheads generated by every node.

The sum of consecutive false positives originated at each node, assuming that the packet's *time-to-live* (TTL) field is set to h, and as derived from the one presented in [SRA+11] for the unicast traffic case, is given by

$$oh(i)_{MPSS} = (d-v-1)\cdot fpr + (d-v-1)\cdot fpr\cdot((d-1)\cdot fpr) + \cdots \\ +(d-v-1)\cdot fpr\cdot((d-1)\cdot fpr)^{h-i} \tag{4.17}$$

$$oh(i)_{MPSS} = \sum_{k=0}^{h-i}(d-v-1)\cdot fpr\cdot((d-1)\cdot fpr)^{k} \tag{4.18}$$

$$oh(i)_{MPSS} = (d-v-1)\cdot fpr\cdot\frac{1-((d-1)\cdot fpr)^{h-i+1}}{1-(d-1)\cdot fpr} \tag{4.19}$$

In equations 4.17, 4.18 and 4.19, $(d-v)$ represents the outgoing links that are not part of the multicast tree. Besides, it is assumed that the evaluation of the outgoing

link that connects to the previous node is excluded from the Bloom filter matching operation, thus obtaining $(d - v - 1)$. Notice that even though false positive packets ocurred after the first hop get back again to the right path of the multicast tree, they are considered as overhead, because thery are actually duplicate packets. The value of $(d - 1)$ represents the number of outgoing links to be evaluated in the next hops (excluding only the evaluation of the previous node).

The total overhead of the MPSS technique, improved with a $TTL = h$ value for bounding the number of hops for all the packets, in this type of network, is given by

$$\begin{aligned} oh_{MPSS} = \quad & oh(1)_{MPSS} + v \cdot oh(2)_{MPSS} \\ & + v^2 \cdot oh(3)_{MPSS} + \cdots + v^{h-1} \cdot oh(h)_{MPSS} \end{aligned} \tag{4.20}$$

$$oh_{MPSS} = \sum_{i=1}^{h} v^{i-1} \cdot oh(i)_{MPSS} \tag{4.21}$$

In order to quantify the performance of any method, we take the formulation of the *forwarding efficiency* (fwe), defined in [JZR+09], which is:

$$fwe = \frac{n}{n + oh} \tag{4.22}$$

The *total overhead* represents the number of extra copies of the packet generated uselessly, and the *forwarding efficiency* is the proportion of *useful packes* and *useless packets*. In general, it can be obviously stated that occurrences of *false positives* are more critical at the top levels of the tree, because they are more likely to cause packet storms. It should be remarked that if a false positive occurs near the leaves, and the packet header is properly signaled with a *time-to-live* (TTL) field on the header indicating the value of h, any packet storm generated would not be so harmful as if no TTL control was set.

4.2.1.3 Header Overhead

It is also important to note that, in order to get a low value of fpr, in MPSS the size of the *zFilter* (the Bloom filter) may need to be large (384, 512, ..., 800 bits), depending on the size of the network. This derives on *header overhead* (the total bandwidth spent on BFs headers) and packet processing issues. For example, in systems with standard data buses of 64-bits, a single 800-bit zFilter matching process would need 12.5 clock cycles. In MPSS this value is

$$ho_{MPSS} = m \cdot n \tag{4.23}$$

Note that this value may not be negligible. For instance, as we will see in Section IV, for $m = 800$ and $k = 5$, considering packets 1000-bytes long, around 10% of *header overhead* is generated. This result is even higher for shorter packets.

4.2.2 The D-MPSS Architecture

The analysis made in the previous section (sec. 4.2.1) reveals that Bloom filter-based approaches, like MPSS, have the following limitations:

- The set of elements to be encoded could be too large (n =total number of links of the tree).
- This implies the need for large Bloom filters, in order to keep the fpr low.
- Even relatively large Bloom filters can not make $fpr = 0$. False positives near the top of the tree are prone to cause more severe forwarding anomalies (longer looping sequences and duplicates only limited by the packet's TTL). Trying to tackle these problems, in the next subsections we describe the basic functioning and the architecture of the Depth-wise MPSS (D-MPSS) multicast forwarding technique.

4.2.2.1 Stackable Bloom Filters

The key idea of this proposal resides on the reduction of the space of elements (n) that are encoded into the Boom Filter as the packet progresses down the tree. To this end, we propose to create individual Bloom filters per each i-th level of the tree. Thus, a Bloom Filter BF_i represents the links belonging to the i-th level of the tree, as shown in Fig. 4.11.

Observe that, in this scenario, individual Bloom filters at the top levels can be very small, because the number of links to be inserted into the Bloom filter is reduced. An example of how the label stack is built is depicted in Fig. 4.12. Once the shortest path tree is computed, a Bloom filter stack is collected for every PE node (leaf node). Each stack is made up of the individual link IDs from the PE node until the source node. After having all the stacks gathered from the leaf nodes, the source OR-s individual filters at each corresponding level, in order to get a single Bloom filter stack for each level. In the example shown, stacks of PE nodes A, B, ..., E are computed individually, and then they are folded into a unique BF stack (BF-stack) of size 3.

4.2.2.2 Reduction of Size of Bloom Filters

In general, BFs from the top of the tree are the result of binding only a few links, as can be seen in the example of Fig. 4.12. The opposite occurs with the deepest BFs. Therefore, it is possible to use fewer bits to encode the links of the initial levels,

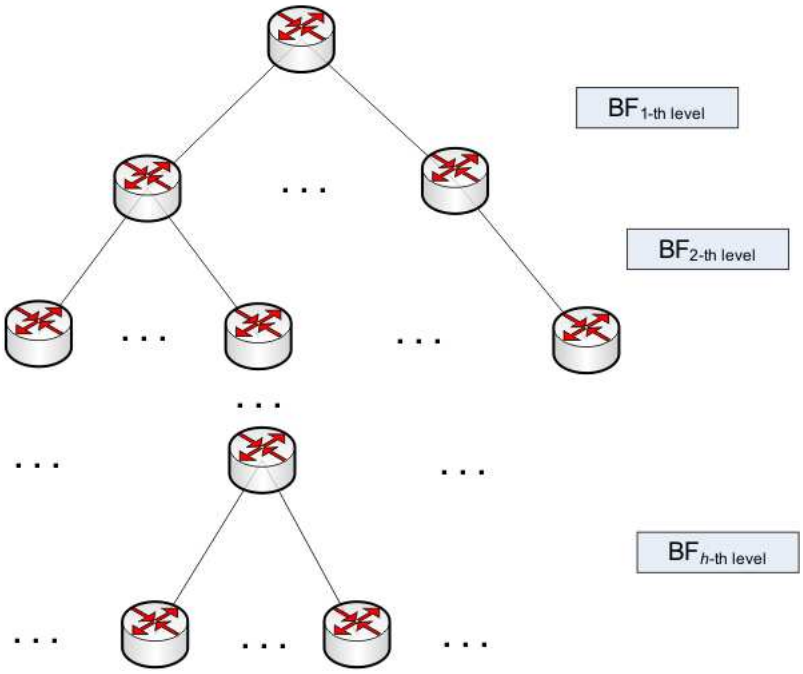


Figure 4.11: A Bloom filter for eah i-th level

and deeper levels would need more bits to encode more links. Hence, in a similar manner that authors described in [SRA+11], we propose to use variable-length BFs, but in our approach we use one BF per level instead of one BF for the whole tree. The idea is to first construct the BF_i (BF of the i-th level of the tree) as a large filter of length M. For this, link IDs must also have a length of M, which could be equal to 512, or 800, or 1024, etc. After that, ρ_i, the fill factor of BF_i, is obtained with

$$\rho_i = \frac{bits_1}{bits_0} \tag{4.24}$$

The optimal length of the Bloom Filter for the i-th level is

$$m_i = \lceil -M \log_2(1 - \rho_i) \rceil \tag{4.25}$$

As claimed by authors in [SRA+11], a filter of this length would result in a *fill factor* of 50%. As stated by authors in [RMM+11], for security reasons, in case an attacker changed the value of the Bloom filters setting too many bits to 1 (increasing in this manner the fpr), it would be desirable to control that always $\rho_i \leq 50\%$.

There are several ways to obtain the new *resized Bloom filter* rBF_i. Fig. 4.13 shows an example of the conversion of BF_i array of length $M = 5$ to a *resized BF* array of

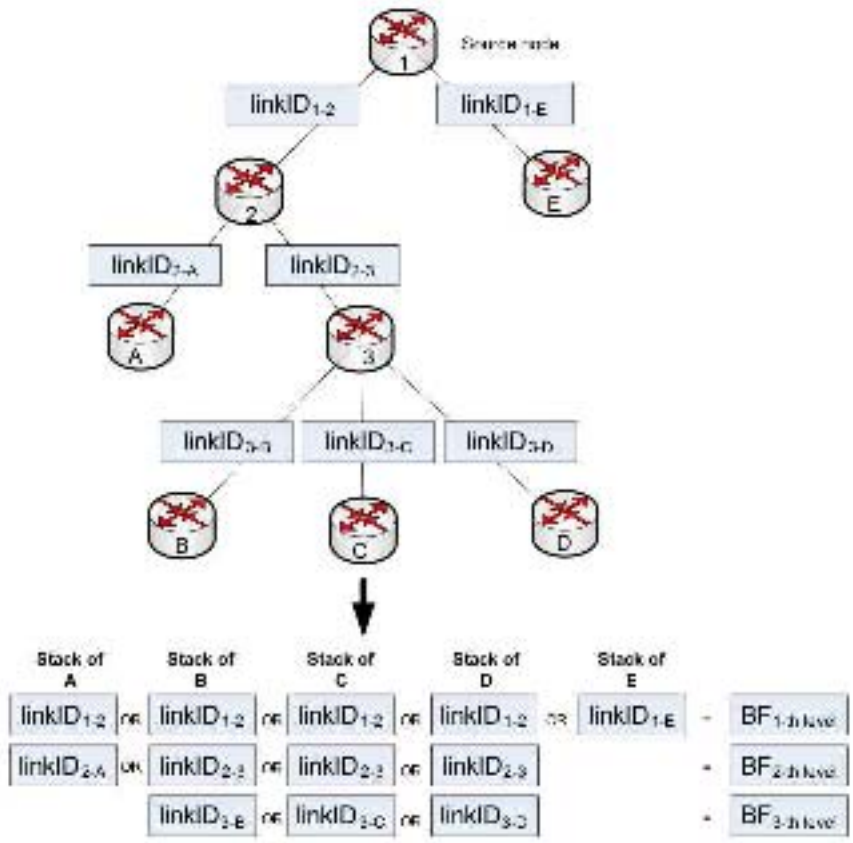


Figure 4.12: Example of the binding of a D-MPSS Bloom filter stack

length $m_i = 3$. We propose that the a-th position of the array of the rBF_i of length m_i to be:

$$rBF_i[a] = \bigvee_{j=1}^{\lceil M/m_i \rceil} BF_i\left[(a-1)\cdot m_i + j\right] \tag{4.26}$$

Note that values that are out of the array bounds are set to 0. In the example, given that $\lceil M/m_i \rceil = 2$, a 0-bit has to be added to BF_i at the 6-th position. In this way, we construct a stack of h *rBFs*, each one pseudo-optimally resized.

In addition, it is necessary to add a few signaling bits before every rBF_i in order to indicate its correspondent value of m_i. Since it is not desirable to add many more bits to the packet header for this function, only multiples of a given number, $mult$ (e.g., $mult = 32$), are accepted as possible lengths of any m_i. If f_i corresponds to the i-th level of the tree, we have that

$$m_i = f_i \cdot mult \tag{4.27}$$

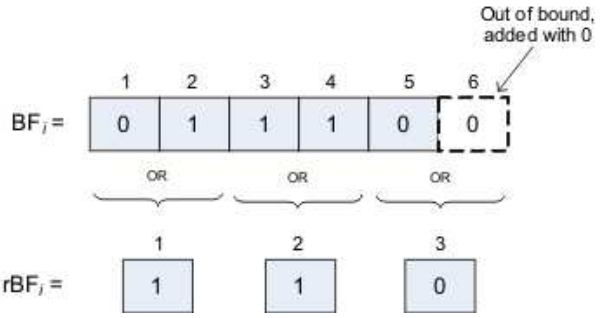


Figure 4.13: Example of converting BF_i, with $M = 5$ into rBF_i,with $m_i = 3$, according to 4.26

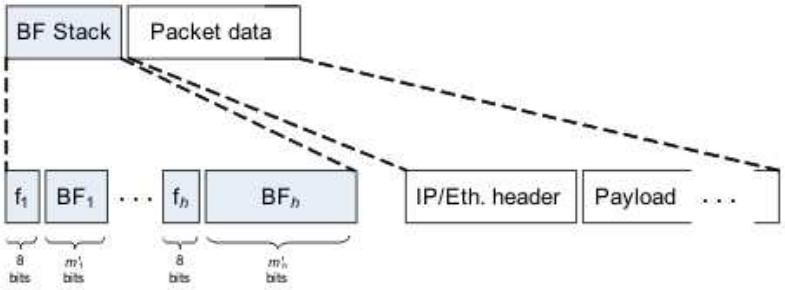


Figure 4.14: D-MPSS packet header

In the worst case, f_h, that corresponds to rBF_h (the largest rBF of the stack) will need only $\log_2 f_h+1$ bits to be represented. For example, if $m_i = 512$ bits, $mult = 32$ and $f_h = 16$, a 5-bit field (let us name it s) is added before each rBF_i, indicating the corresponding f_i value. It may happen that when calculating 4.28, it results that $m_i > M$. This means that, instead of being reduced, the BF needs to be resized to a larger array. This would derive on having a very large rBF_i, which can be very good to avoid false positives, but not convenient for keeping a reasonable header size. Therefore, in order to avoid this, M is taken also as the upper bound of m_i, so that always

$$m_i = f_i \cdot mult \leq M \tag{4.28}$$

Finally, the packet header has the structure shown in Fig. 4.14 (the subscripts indicate the level of depth of the tree).

4.2.2.3 Packet Forwarding

The stack, made up of $rBF_1, \ldots, rBF_h$ is computed and put into the packet header by the source node. At each hop in the i-th level of the tree, only rBF_i is evaluated

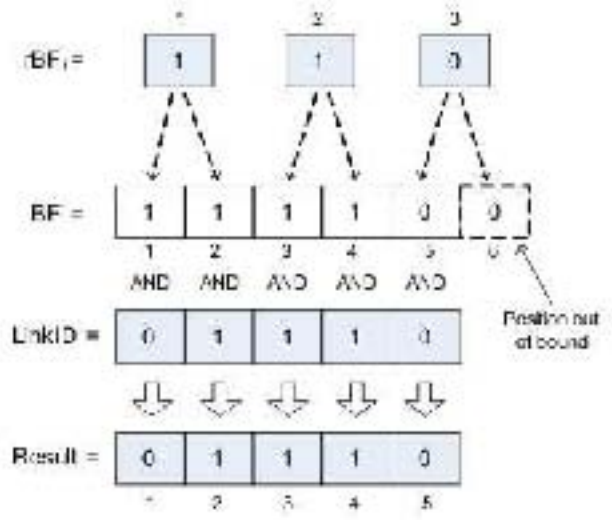


Figure 4.15: Example of matching evaluation of an rBF_i

for matching operations; after that the rBF_i and its correspondent f_i field are popped (removed) from the stack, in a similar manner to MPLS label stacking [RTF+01]. However, it should be noted that unlike in MPLS, there could be the case that the destination PE may have to pop all the *rBFs* that still remain in the stack. This will happen for all PEs that are at a shorter distance from the source node than the tree depth h. In the context of MPSS (and also in this proposed scheme), the PE node is then able to read the inner header with the VPN forwarding information to send it to the corresponding Customer Edges (CE).

Given that all the *rBFs* have been resized, the forwarding decision in a node is made as the example depicted in Fig. 4.15. First, each bit $rBF_i[a]$ is duplicated M/m_i times (in the example $M/m_i = 2$) and put on a BF' temporary array of size M. Then, every j-th bit of the BF' array is AND-ed with every j-th bit of the $LinkID$ array. If the resulting array is exactly the same as $linkID$, the matching operation is successful, then rBF_i is popped from the BF stack, and the resulting packet is copied and forwarded to all the outgoing successfully matched interfaces. Note that in the example the 6-th position of BF' is not taken into account because it is out of bound. Also note that for each matching evaluation the D-MPSS packet forwarding procedure realizes M AND operations, which is the same number of operations performed by MPSS.

4.2.3 Performance Evaluation

In this section we present the evaluation of the D-MPSS performance, made analytically and experimentally (with simulations).

4.2.3.1 False Positives Rate

In this method, assuming a fully regular network and delivery tree, the fpr_i of each rBF_i has to be calculated as in Eq. 4.14, but considering that n_i is the number of links of the i-th level. It should be noted that the reduction of the number of links to be encoded reduces the *false positives rate*. The calculation of fpr_i (fpr of the i-th level of the tree) in this case is not straightforward, because any given bit of rBF_i is the result of OR-ing together M/m_i bits of the BF_i array (Eq. 4.28). In the BF_i array (before being resized), the probability that a bit is still 0 after inserting n_i elements is given by the same formulation of Eq. 4.11 (considering that for practical effects, $M = m$):

$$\left(1 - \frac{1}{M}\right) \tag{4.29}$$

Given that rBF_i is the result of OR-ing groups of $\lceil M/m_i \rceil$ bits, the probability of a bit in rBF_i to still being 0 is

$$\left(\left(1 - \frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}} \tag{4.30}$$

Then, the probability that a bit is set to 1, which is equivalent to the *fill factor* of the i-th level, is

$$\rho_i = 1 - \left(\left(1 - \frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}} \tag{4.31}$$

For the membership test of an element, the probability that all the k array positions are set to 1 even if the element is not part of the set (i.e. fpr_i), is

$$fpr_i = \rho_i^k = \left(1 - \left(\left(1 - \frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}}\right)^k \tag{4.32}$$

Finally, considering that $M = m$, we show that, in general, $fpr \leq fpr_i$:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \leq \left(1 - \left(\left(1 - \frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}}\right)^k \tag{4.33}$$

$$1 - \left(1 - \frac{1}{m}\right)^{kn} \leq 1 - \left(\left(1 - \frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}} \tag{4.34}$$

$$\left(1-\frac{1}{m}\right)^{kn} > \left(\left(1-\frac{1}{M}\right)^{kn_i}\right)^{\frac{M}{m_i}} \tag{4.35}$$

$$\left(1-\frac{1}{m}\right)^{kn} > \left(1-\frac{1}{M}\right)^{kn_i\cdot\left(\frac{M}{m_i}\right)} \tag{4.36}$$

$$\left(1-\frac{1}{m}\right)^{n} > \left(1-\frac{1}{M}\right)^{n_i\left(\frac{M}{m_i}\right)} \tag{4.37}$$

Assuming that $m = M$, $fpr \leq fpr_i$ if and only if:

$$n > \frac{n_i \cdot M}{m_i} \tag{4.38}$$

Note that with D-MPSS, an invisible implicit time-to-live (TTL) field is added to the mechanism, because in the worst case, the packet will traverse only up to h links. A false positive or duplicate packet will, in the worst case, end its travel whenever the label stack gets empty. However, the impact of duplicates and loops will be much lower in practice, as the BF becomes different at each visited node.

4.2.3.2 Overhead

On the other hand, Eq. 4.19, that represents the overhead caused by each visited node for MPSS, needs to be reformulated for D-MPSS, obtaining

$$\begin{aligned} oh(i)_{DMPSS} = (d-v-1)\cdot fpr_i + (d-v-1)\cdot fpr_i \cdot (d-1)\cdot fpr_{i+1} + \cdots \\ +(d-v-1)\cdot fpr_i \cdot (d-1)^{h-i} \cdot fpr_{i+1} \cdots fpr_h \end{aligned} \tag{4.39}$$

$$oh(i)_{DMPSS} = (d-v-1)\cdot fpr_i \cdot \left(1+\sum_{j=1}^{h-i}\left((d-1)^j \cdot \prod_{r=i+1}^{i+j} fpr_r\right)\right) \tag{4.40}$$

The *total overhead* is calculated exactly as in Eq. 4.21. For example, let us consider a multicast routing tree with $v = 2$ and $h = 6$, which makes $n = 126$ the total number of links in the tree. In the network each node has d neighbors. In Tab. 4.3 and Fig. 4.16 we provide the calculations of the *total overhead* and the *forwarding efficiency* (fwe) for both MPSS and D-MPSS approaches. For MPSS the optimal number of hashes has been calculated according to $k_{opt} = \ln(2)m/n$, provided by authors in [TRL12]. We find results for different values of d, which show that the D-MPSS approach reduces the *total overhead* and improves the *forwarding efficiency*

d-1	MPSS				D-MPSS			
	m=800, k_{opt}=4		m=1024, k_{opt}=6		M=800, k=6		M=1024, k=7	
	oh	*fwe*	*oh*	*fwe*	*oh*	*fwe*	*oh*	*fwe*
5	68	65%	25	83%	4	97%	2	98%
10	238	35%	75	63%	12	91%	5	96%
15	534	19%	135	48%	20	86%	9	93%
20	1,071	11%	209	38%	29	81%	12	91%
25	2,054	6%	301	30%	38	77%	16	89%
30	3,814	3%	419	23%	47	73%	20	86%

Table 4.3: Total overhead (oh) and forwarding efficiency (fwe) of MPSS and D-MPSS techniques for a multicast tree with $v = 2$, $h = 6$ over a regular network with a constant node degree of d.

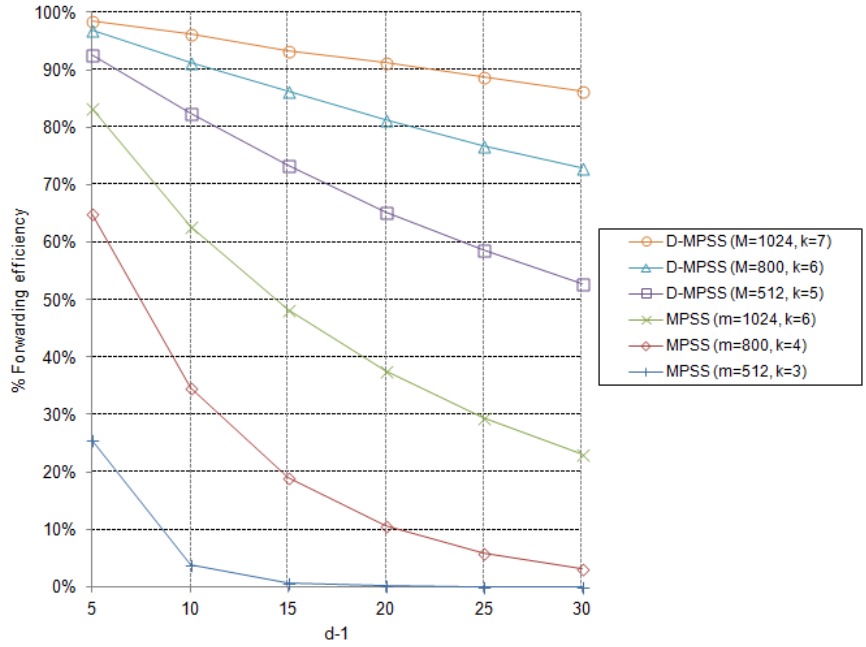


Figure 4.16: Forwarding efficiency of MPSS and D-MPSS techniques for a multicast tree with $v = 2$, $h = 6$ over a regular network with a constant node degree of d.

(fwe) in this dense network. It is important to observe that our approach outperforms MPSS even more in larger networks (or networks with a higher average degree per node).

v	n	**MPSS**		**D-MPSS**			
		m=800	m=1024	M=800, k=6		M=1024, k=7	
		k_{opt}=4	k_{opt}=6	$mult$=8	$mult$=32	$mult$=8	$mult$=32
2	126	10.0%	12.8%	4.6%	4.7%	5.3%	5.6%
3	1092	10.0%	12.8%	4.8%	4.8%	6.1%	6.1%
4	5460	10.0%	12.8%	3.3%	3.3%	4.3%	4.2%

Table 4.4: Calculations of total header overhead (ho) for a multicast tree over a regular network with a constant node degree and$v$ branches outgoing from each node.

4.2.3.3 Header Overhead

We call *header overhead* to the bandwidth wasted in the Bloom filter header that is necessary to forward packets. Considering that the top rBF_i is popped at each hop and that after the penultimate last hop the packets does not have any rBF anymore (because the stack is empty), in D-MPSS, unlike MPSS (Eq. 4.23) the total number of overhead bits caused by the headers is

$$ho_{DMPSS} = \sum_{i=2}^{h} \left((m_i + s) \sum_{j=1}^{i-1} n_j \right) \tag{4.41}$$

where m_i is the length of rBF_i, n_j the number of links of the tree at the j-th level, and s the number of bits to store the value of f_i (see 4.28) to indicate the size of rBF_i. The reason why the outer sum does not start in $i = 1$ is because the source node (at the top of the tree) pops the rBF_1 once it is evaluated and it is not actually forwarded to the next hop.

As an example, considering the same type of regular tree for the calculation of Tab. 4.3, results of the *header overhead* are shown in Tab. 4.4 and Fig. 4.17 for both approaches, varying the value of v (the number of branches at each node) and considering uniform packets of 1,000 bytes. For D-MPSS, different values of $mult$ have been considered, according to (Eq. 4.28).

It is clear that D-MPSS headers are shorter than MPSS headers, which is quite an important improvement. Observe that in D-MPSS the header overhead does not increase for $v = 4$. This happens because m_i depends on the value of ρ_i (Eq. 4.25), which finally depends on the value of k. The second reason is that M is the largest size allowed for any rBF_i, and for the lowest levels of the tree (where there are many more links and therefore bigger values of m_i would be needed) m_i should set to the M bound, when $m_i \geq M$.

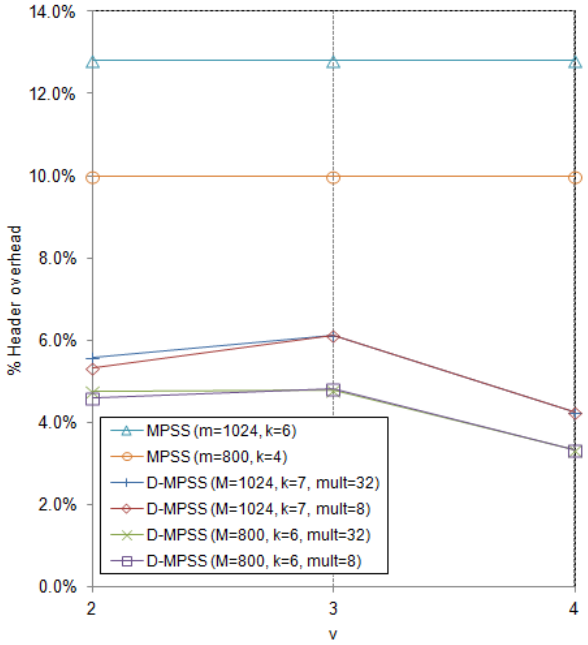


Figure 4.17: Header overhead of MPSS and D-MPSS techniques for a multicast tree over a regular network with a constant node degree and v branches outgoing from each node.

4.2.4 Simulations and Results

We have constructed a simulation model to work with real core network topologies, with the aim to test the MPSS and D-MPSS techniques in diverse real scenarios. The network topologies used are the same as in Tab. 4.1, which have been extended according to the provider aggregation model depicted in Fig. 4.3 and described in sec. 4.1.2. PE routers provide Internet and Layer 3 and Layer 3 MPLS VPN services from these major locations. In simulations used for the aggregation techniques (sec. 4.1.1) 1,000 random sets of VPN samples with uniform distributions were generated for each type of simulation, and the final results are the average of a number 10 iterations (making a total of 10,000 random VPNs).

Considering these scenarios, extensive simulations were programmed and run under Java 1.7.0_03, using the IDE Eclipse Juno. For the sake of fair comparison with the latest state-of-the-art techniques to mitigate Bloom filter-based forwarding anomalies, the MPSS technique was implemented with the bit permutation method proposed in [SRA+11] and including a *time-to-live* (TTL) field in the header with the h value of the tree, in order to limit further the scope of packet storms. We will refer to this hybrid technique in our results as *MPSS-ttl-perm*. Without this modification, false positive packets would keep traveling through the network until reaching PE nodes that discard them.

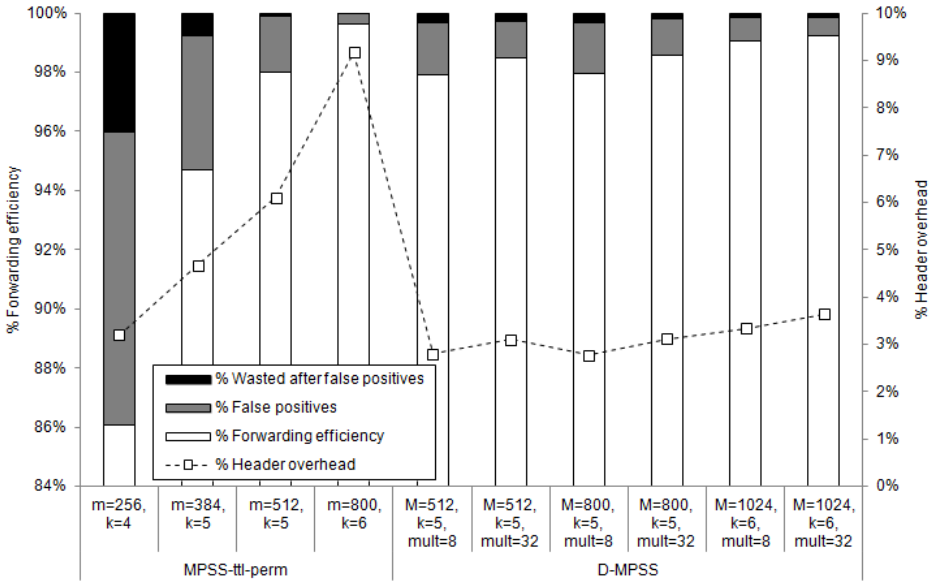


Figure 4.18: MPSS and D-MPSS results in the NSFNET network.

In Fig. 4.18, Fig. 4.19 and Fig. 4.20 we present the results corresponding to the *forwarding efficiency*, number of false positives, and bandwidth wasted (overhead) due to false positives. Results are given in percentage. In the case of D-MPSS, *mult* is the multiple considered to construct the *rBFs* (as explained in sec. 4.2.2.2). M is the initial length of the Bloom filter that acts also as the upper bound of m_i (Eq. 4.28). Left axes of these figures show the percentage of *forwarding efficiency* and right axes show the percentage of the *header overhead*.

The results show that, for the medium-sized network (NSFNET), D-MPSS outperforms MPSS except when MPSS makes use of a very large Bloom filter (Fig. 4.18). In this case MPSS needs to encode 800-bit Bloom filters (9.2% of header overhead) to achieve an acceptable fwe. However, the *header overhead* of D-MPSS is below 4.0%, unlike MPSS of 384 bits, 512 bits and 800 bits. Also, in D-MPSS, the *mult* value shows an important behavior regarding *header overhead*: the header size is better handled with small values of *mult* (remember that mult is the multiple of the size of *rBFs*). However, lower values of *mult* (like in the example, $mult = 8$) shows a slight decrease of forwarding efficiency (fwe).

Within larger networks, such as Abilene and COST-266 of Fig. 4.19 and Fig. 4.20, D-MPSS does achieve to improve the *forwarding efficiency*, but with some penalty for the *header overhead*. Since these networks may generate larger trees, the only BF used by MPSS decreases its forwarding efficiency too much. It is really interesting to observe that D-MPSS performs well irrespective of the size of the network.

As explained before, the TTL value (in MPSS) and the BF stack size (in D-MPSS) do not allow false duplicate packets to go beyond h hops. Despite this implementa-

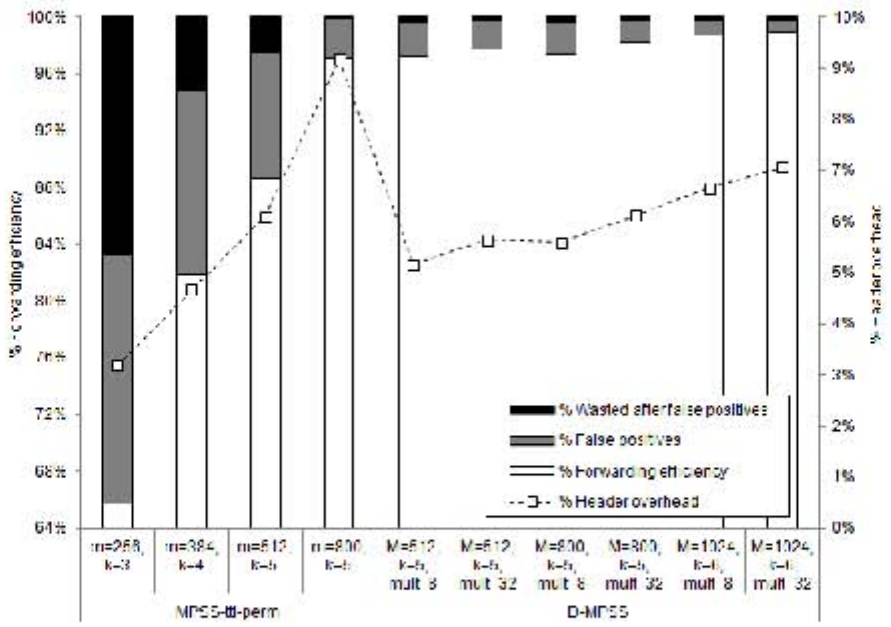


Figure 4.19: MPSS and D-MPSS results in the Abilene network.

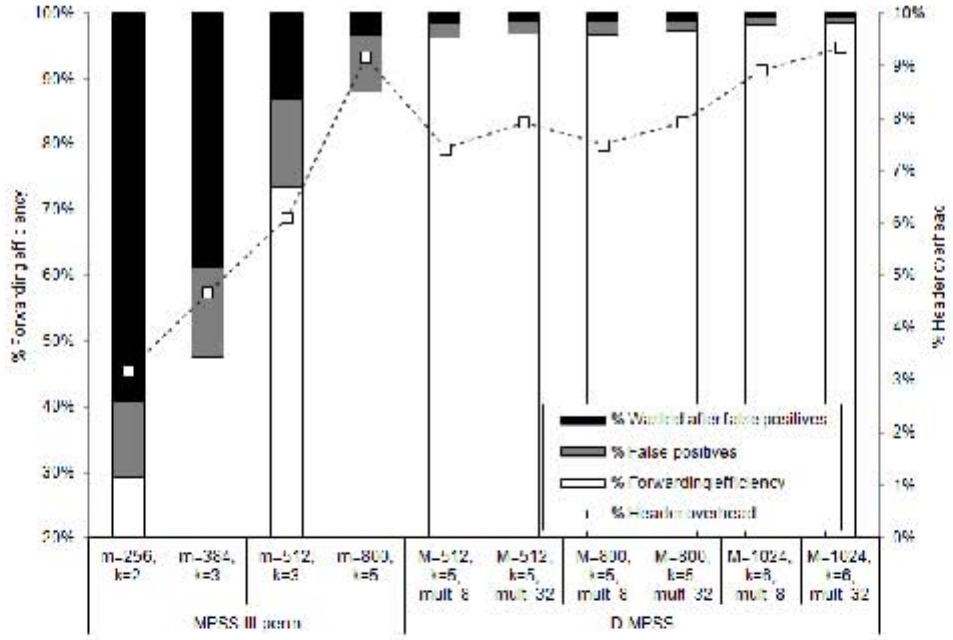


Figure 4.20: MPSS and D-MPSS results in the COST-266 network.

tion, loops may appear, but they are not infinite anymore (as would happen with the original proposal of MPSS). Fig. 4.21 shows the average number of loops de-

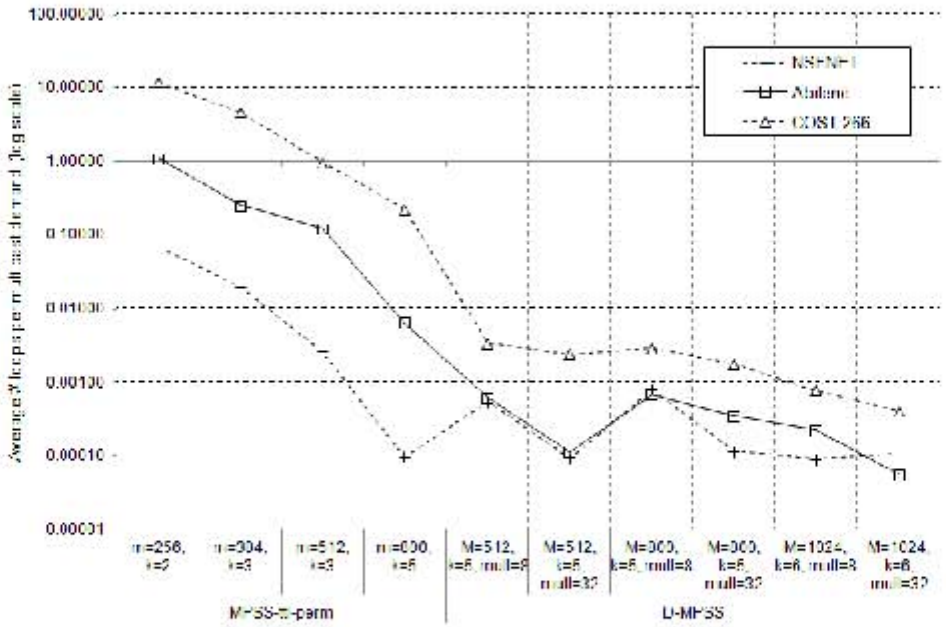


Figure 4.21: Average number of loops per multicast demand.

tected during simulations, per each multicast demand. In the refered figure, a value of 10 -for example- means that, on average, in each multicast tree 10 forwarding loops ocurred. A value of 1 means that only one forwarding loop appeared in each multicast tree, on average. These results show that in D-MPSS the number of loops are extremely low and negligible in all the cases, which does not happen in MPSS (see cases where $m = 256, 384, 512$ in larger networks).

4.2.5 Conclusion

In this section we have proposed and studied a new method to encode multicast trees and improve the forwarding efficiency of traffic delivery under the family of stateless Bloom filter-based distribution mechanisms. Results show that D-MPSS is an effective method to get rid of forwarding anomalies while achieving high forwarding efficiency, and that it outperforms the best known variant of the MPSS concept up to date [SRA+11]. Although the chosen scenario in this work is that of the multicast MPLS VPN-based networks present also in the MPSS technique, this mechanism could be applicable for other types of networks where Bloom filters has been recently studied, such as inter-domain publish-subscribe networks.

4.3 Hybrid Aggregation - Bloom Filter-based Forwarding (H-ABF)

Although D-MPSS [FL12] (described in sec. 4.2) is a very efficient multicast forwarding technique based on Bloom filters that outperforms other related similar state-of-the-art techniques, forwarding anomalies and security vulnerabilities are still present in it. It seems that the *almost-stateless* characteristic is desirable on one side, but it is obvious that the fact of not keeping forwarding state information should hinder the network performance in some way. The original multicast forwarding system that maintains state entries at nodes, in spite of being impractical when its number is large, provides a little more control and security for network administration. On the other hand, as it has been seen previously in sec. 4.1.1, an adjustable reduction of the state information can be achieved with *aggregation* techniques at the price of larger bandwidth waste.

In this section, we explore the possibility of realizing multicast forwarding by combining *aggregation* with *Bloom filters*. This could be made by having two fields in the packet header: One corresponding to the MPLS label corresponding to the *shared tree*, and the other to the *Bloom filter*. Our proposal is mainly supported by the idea that the reduction of outgoing links to be evaluated (with any of the *aggregation* techniques) at each node would reduce the *overhead traffic* generated by false positives. According to the example presented in Fig. 4.22, part (a) presents an individual tree to forward packets from A to B, C and E nodes. In part (b) the same tree is aggregated to a given *shared tree* (note that the individual tree is a subset of the ST). Observe that in the first case (without aggregation) the number of links to be tested (done with any of the Bloom filter-based techniques) in nodes A and C, are 6 in all; but in the second case there are only 4.

4.3.1 Discussion and Basic Functioning

In some previous works ([RJN+09], [SRZ+10] and [Ant11]) security vulnerabilities of the Bloom filter-based forwarding techniques were studied, like the *attack of the 100% fill factor*. In this, an attacker sends a packet with a Bloom filter with all the bits set to 1; thus, the packet would match possitively with every outgoing *link ID,* causing a high rate of false positives and a chain reaction. In the case of MPSS [ZJS+10] this would generate an infinite packet storm, unless a *TTL* value included into the header were used for not allowing any packet to exist for more than h hops. Although the most simple way of protection against this attack should be by limiting the proportion of 1s that can be present in every Bloom filter to, for example 50% ([RJN+09], [SRZ+10] and [Ant11]), that action could be not convenient for the performance of the forwarding system, because it would also limit the capability of BFs for inserting new elements, in which case the only solution would be to have larger BFs.

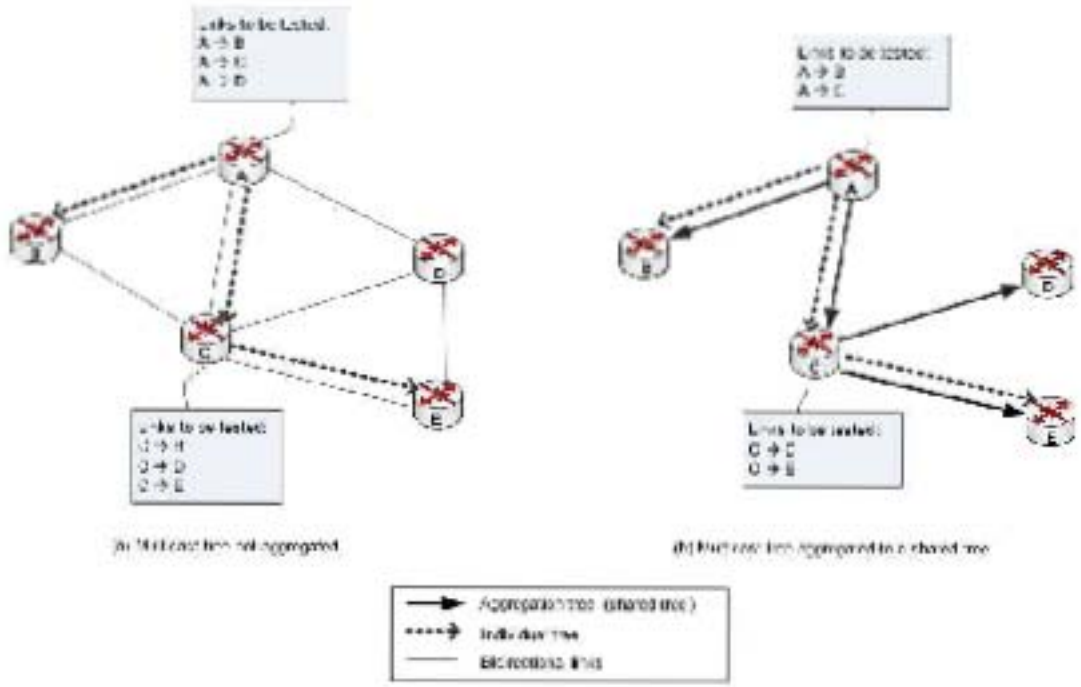


Figure 4.22: Number of links to be tested with any Bloom filter-based technique without/with aggregation.

With the use of *shared trees* it would be possible to remove the *fill factor* limitation, because they implicitly create some kind of a *subdomain* of the network graph, limiting the BF-matching evaluation of packets to a reduced number of links at every node. In the event of an injection of 100% *fill factor* packets, the generated packet storm would only spread over the *shared tree* bounds. In the example depicted in Fig. 4.23 (a), the *100% fill factor attack* would generate a packet storm in any of the BF-based schemes, that would only stop when $TTL = 0$ (in the case of MPSS with TTL and permutation, described in sec. 4.2.3) or when the stack of BFs was empty (in the case of D-MPSS). In return, if the same tree was aggregated to a *shared tree* like in Fig. 4.23 (b), the packet storm would be limited only to the active links of the ST. With this situation described here, it is useful to state that, in the worst case, all the wasted packets would traverse only the links of the *shared tree* in one direction, and only once.

Another important feature of H-ABF is that the *forwarding loops* present in Bloom filter-based approaches can also be mitigated with the support of the *aggregation* approach. Let us suppose that the packets are encoded by using the original MPSS design [ZJS+10], i.e. without considering any $TTL = h$ field. Whithout *aggregation*, in the worst case 100% fill factor BFs would not only generate a packet storm like in Fig. 4.23 (a), but packet storms amplified with infinite loops. However, in the aggregation case, any chain reaction would always be limited to the unidirectional

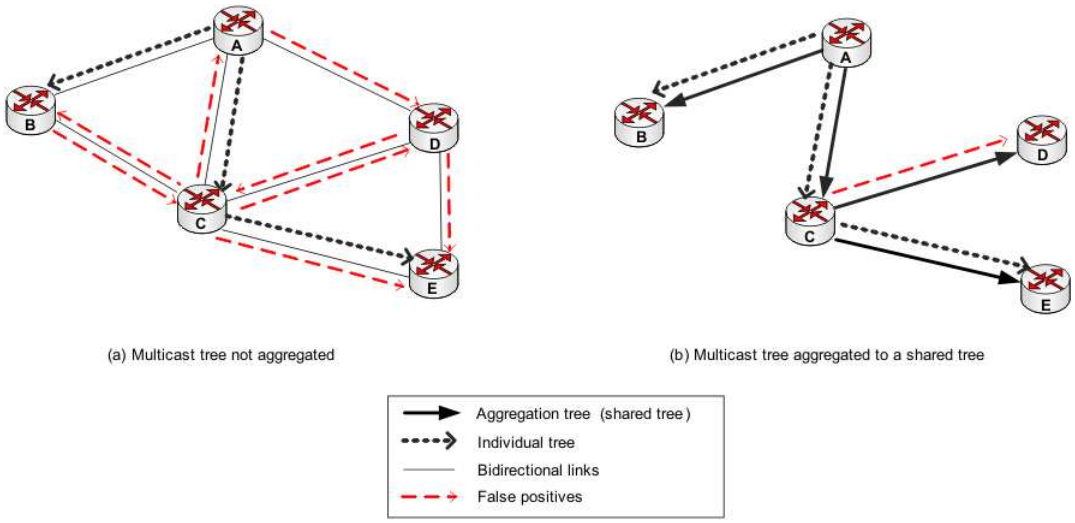


Figure 4.23: Packet storm with a Bloom filter-based technique without and with aggregation.

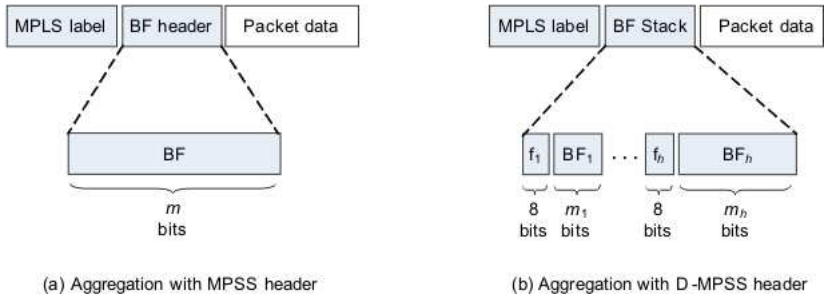


Figure 4.24: Packet header of the H-ABF technique, using MPSS (a) and D-MPSS (b) as BF methods.

links of the *shared tree*, as shown in *Fig. 4.23* (b).

Although H-ABF may be used for any type of network that requires to enable its provision of multicast services, we develope it for the scenario of an MPLS VPN-based network, as explained in sec. 4.1 and described in Fig. 4.1. As in sec. 4.1, in our scheme, u multicast VPNs (MVPNs) are aggregated to s *shared trees*, which are set up among the transport network. Core intermediate nodes keep information only for the forwarding of the STs, but do not have any knowledge about the VPN specific information of the packets that are traversing them. The VPN specific information is encapsulated in the packets at the Provider Edge (PE) source node, and is only inspected at the arrival to destination PEs. The main difference here is that the Bloom filter is located between the MPLS label (corresponding to the shared tree) and the packet data, as shown in Fig. 4.24.

4.3.1.1 Packet Forwarding

For the realization of packet delivery, every node maintains two forwarding tables: The standard *MPLS forwarding table,* with the corresponding entries for the routing of the shared trees; and the *link IDs table,* with the entries of every outgoing interface encoded as Bloom filters. When a packet arrives at a core node, it first makes the look up process of the header within the *MPLS forwarding tabl*e; if it succeeds, then it performs the matching evaluation for the Bloom filter header within the *links IDs table*. Depending on what filtering technique is being used, the matching process follows the steps described in sec. 4.2 for MPSS (with permutation, as in [SRA+11]) or D-MPSS.

Let us have the example provided in Fig. 4.25, that represents the state information of nodes A and C. The two shared trees, ST-1 and ST-2, are correctly configured with their respective forwarding entries at the *MPLS forwarding tables*. To serve a given multicast VPN, the individual multicast tree of the figure has been aggregated to ST-1. The fact that the multicast tree is aggregated to ST-1 means that all the traffic of this tree would use the state information assigned to ST-1 (*MPLS forwarding table*). Since more individual multicast trees will be aggregated to ST-1 and ST-2, the inner Bloom filter of the packet will have to be matched according to the BF technique and the *linkIDs table.*

Following the same example of Fig. 4.25, in Fig. 4.26 two packets of the multicast VPN with its corresponding MPLS - BF headers are launched to the network from node A (for this example we assume that the BF technique adopted is MPSS). The multicast tree of the VPN demand should visit nodes B, C and E. By following the first check (the MPLS look up), according to the state information of the STs at nodes A and C (Fig. 4.25), packets should be delivered to nodes B, C, D and E. After the second check (the BF matching) the packet in Fig. 4.26 (a) is filtered adequately and it is only delivered to nodes A and E. However, the packet in Fig. 4.26 (b) gives a false positive and is delivered also to node D.

4.3.1.2 Types of Shared Trees Models

Fit Shared Trees (FST)

In this model, as it was introduced in sec. 4.1, s shared trees are created for aggregating u multicast VPNS, obtaining the *aggregation ratio* (Eq. 4.1). By following the IA and IA+R aggregation techniques proposed in that section, the multicast VPNs are included (aggregated) to the most similar available ST -that is why it is called *fit shared trees* (FSTs). If necessary, after the aggregation the chosen ST can be extended in order to contain the new VPN. Every ST has its own *rendesvouz point* (RP) node (the root of the shared tree) and every PE source node has first to deliver the packet to it encapsulated in a *label switched path* (LSP), and then the RP performs the multicast forwarding process.

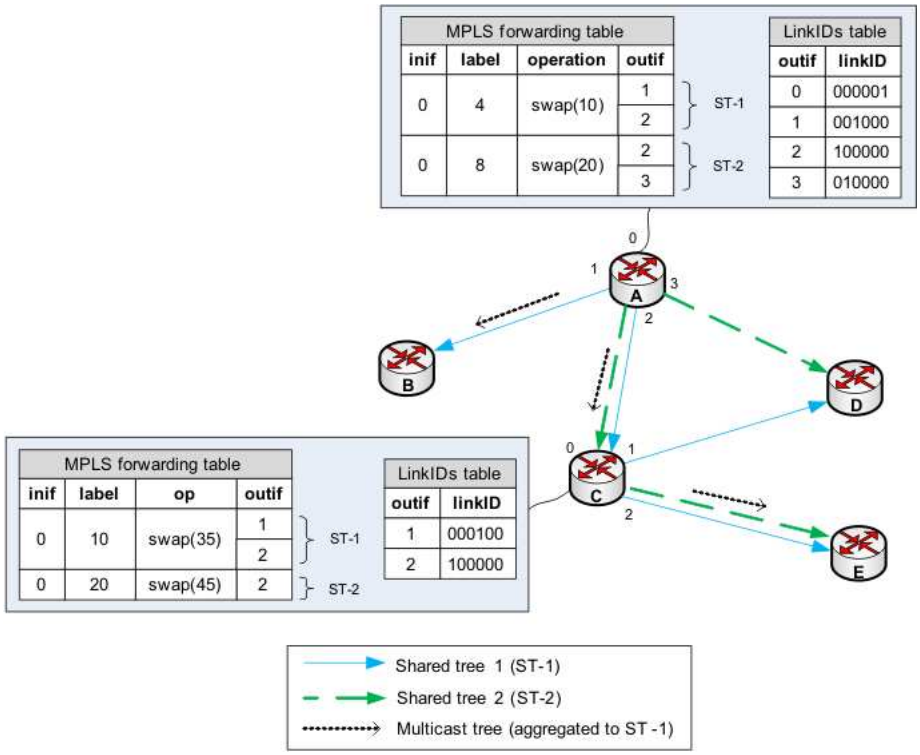


Figure 4.25: Example of *MPLS* and *link ID forwarding tables*.

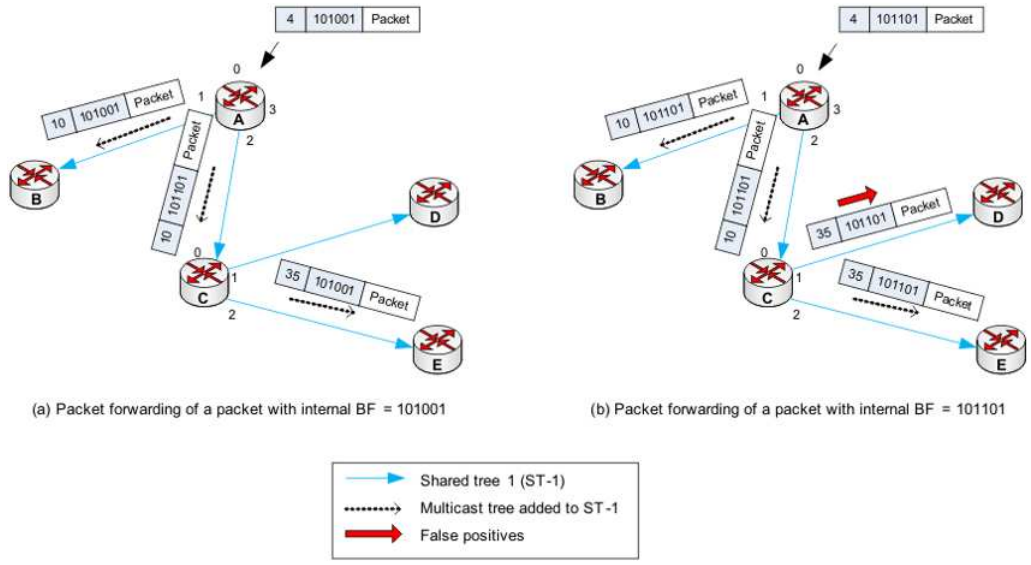


Figure 4.26: A packet forwarding example of the H-ABF technique.

This model presents two drawbacks:

1. The creation of FSTs and *on-the-fly* computing for the accommodation of multicast demands into them is not a trivial task.
2. Moreover, the RP of a FST is supposed to be the node with the lowest ϵ value, i.e. the average distance (in number of hops) from the RP to the rest of nodes that take part of the MVPN (sec. 4.1.1). It means that, in addition to the bandwidth wasted because of packets delivered to non-destination PEs, ϵ links will be wasted in average.
3. Another problem is that, given s the number of FSTs and $degree(node_i)$ (i.e. the number of entries of its *linkIDs forwarding table*), the upper bound for the number of state entries at each node is given by 4.42. Since s changes according to the desired AR (Eq. 4.1), when $AR \geq 50\%$ the list of entries at *MPLS forwarding tables* of core nodes could be really large in the worst case (more than a half of the number of multicast demands).

$$\#entries_{FSTs}(node_i) \leq s + degree(node_i) \tag{4.42}$$

Broadcast Shared Trees (BST)

In order to improve the FST model, we propose the creation of *broadcast shared trees* (BSTs). In this model a number of broadcast trees are created, all of them with their roots located very close to the PE nodes. Taking the same MPLS-VPN network model of Fig. 4.1 and the *provider aggregation model* of Fig. 4.3, the PA nodes (nodes located between core and PE nodes) are selected as the roots of the broadcast trees, and there is a single BST for each PA node acting as the *rendesvouz point* (RP). For a better explanation we provide an example in Fig. 4.27. Note that in this case the number of state entries at each node is exactly given by

$$\#entries_{BSTs}(node_i) = |PAnodes| + degree(node_i) \tag{4.43}$$

Given that $|PAnodes|$ is a constant value, the number of entries in this model remains also constant at each node and, depending on the network size, in general $\#entries_{BSTs}(node_i) \leq \#entries_{FSTs}(node_i)$, except for high values of AR (e.g. $AR = 100\%$), when a very few STs are built (e.g. one ST for aggregating all the traffic). If state consumption is definitely an issue in the design, a single *spanning tree* can do the job, leading to

$$\#entries_{BSTs}(node_i) = degree(node_i) \tag{4.44}$$

In Fig. 4.27, we present an example of a BST, which is supposed to be one of the 8 BSTs built by default before any transmission is done (each per PA node). As it may be observed, VPN 1 is attached to 6 of the points of presence or PE nodes, and all of them can potentially be the root of another multicast transmission. Let us suppose that the PE node labeled as *PE source node* originates a multicast transmission, in

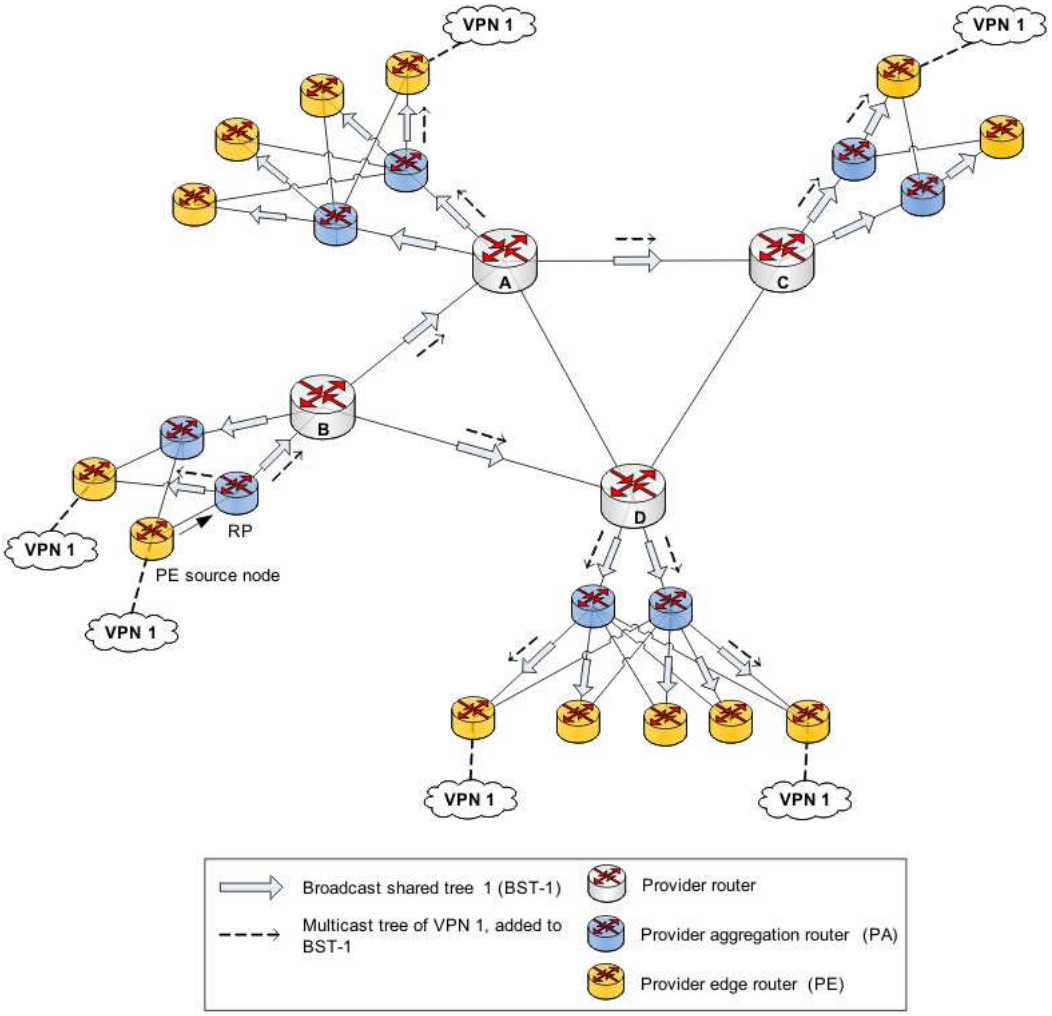


Figure 4.27: Example of a *broadcast shared tree* (BST) in an MPLS-VPN network

which case it chooses to be aggregated -among the 6 BSTs available- to the BST with the nearest RP (in this case, *BST-1* with its RP attached to the provider backbone node B). As *BST-1* has been set up to reach every node in the network, multicast packets will be delivered through it to all PE nodes. Since the routing mechanism has to perform a double check, the inner Bloom filter-based technique (MPSS or D-MPSS) is then applied in order to filter the transmission through not desired links. When any other site of VPN 1 has to send packets, the multicast transmission has to be aggregated to the BST with the nearest RP.

4.3.2 Performance Evaluation

This proposal, as being a hybrid technique that combines aggregation with Bloom filters for the multicast packet forwarding, is expected to reduce the overhead traffic generated by false positives, overcoming in this aspect -to the best of our knowledge- all the previous and current approaches that aim with the same problem. Nevertheless, as explained before, the counteract of this proposal is that it needs a little of state information (specially for the BST model) and some extra processing. In this subsection we formulate the corresponding numerical analysis in order to measure these facts.

4.3.2.1 False Positives and Overhead

Both in MPSS and D-MPSS techniques, the *false positives rate* (fpr) depends on m, n and k values (see 4.14 and 4.32), i.e. the Bloom filter size, the number of links/elements to be added to the BF, and the number of hash functions, respectively. In this sense, the fpr of the H-ABF technique has the same formulation as its internal BF technique used.

However, the real benefits of the hybrid technique should be seen in the reduction of the overhead traffic. Since now the domain of links for the BF matching process at each node is reduced to the set of branches of the *fit shared tree* (FST) or *broadcast shared tree* (BST), less overhead packets are generated. Computing how many links are part of an ST might be an impossible task to be realized analytically in the case of the FST model, because it depends on the number of STs built, in the first place; i.e. note that the more STs are built, the less number of links they have. It also depends on the distribution of the multicast VPNs sites and the similitude among the multicast MVPN demands.

However, the BST model is indeed feasible to be modelled analytically. Let us consider a regular network, like the one presented for the formulation of Eq. 4.19, where d is the constant degree of each node, v the number of branches of the tree at each node, h the depth of the tree, and i the depth of a node in the tree. Remember that in Fig. 4.10 an example of the behavior of the overhead propagation in such a regular network is shown. The shortest BST from the RP node will have $N-1$ edges, where N is the total number of nodes of the network. Giving that N is a value dependent on the network topology, and impossible to derivate from the other network variables, we consider that at the i-th level of the tree there are $\delta \cdot v^{i-1}$ actual nodes. Let us consider δ as the scalar value that defines the network dimensions, so that

$$N = \delta \cdot n \tag{4.45}$$

where n (the constant number of branches of the tree) is

$$n = \sum_{i=1}^{h} v^{i-1} \tag{4.46}$$

$$n = v \cdot \frac{(1 - v^h)}{(1 - v)} \tag{4.47}$$

Now the average fanout, i.e. the average number of branches of the BST at any node of the i-th level is

$$fanout(i)_{BST} = \frac{\delta \cdot v}{(d-1)^{i-1}} \tag{4.48}$$

Thus, following the formulation from Eq. 4.19, when a multicast tree is encoded as an MPSS Bloom filter and is aggregated to the BST described before, the total number of consecutive false positives generated is

$$\begin{aligned} totalOverhead_{BST-MPSS} = \quad & fpr \cdot fanout(1)_{BST} + fpr^2 \cdot fanout(1)_{BST} \cdot fanout(2)_{BST} + \cdots \\ & + fpr^3 \cdot fanout(1)_{BST} \cdots fanout(h)_{BST} \end{aligned} \tag{4.49}$$

$$totalOverhead_{BST-MPSS} = \sum_{i=1}^{h} \left(fpr^i \cdot \prod_{j=1}^{i} fanout(j)_{BST} \right) \tag{4.50}$$

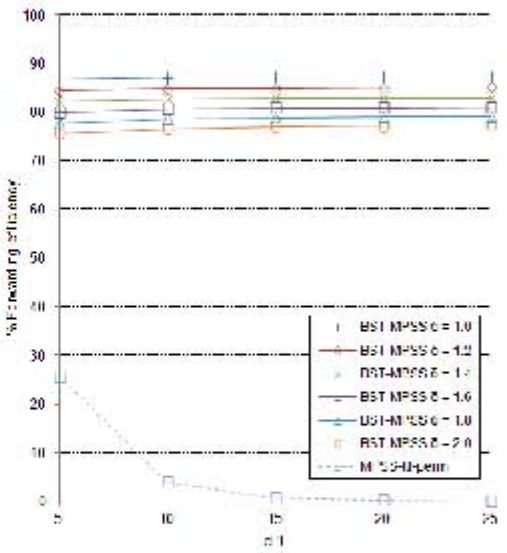
Let us consider a multicast tree with $v = 2$ and $h = 6$, which makes $n = 126$ the number of edges of the tree. In Fig. 4.28 we provide the calculations of the *forwarding efficiency* (defined in Eq. 4.22) for BST-MPSS, with different values of d and δ, and comparing it with native MPSS. As expected, analytical results show clearly the great reduction of the overhead of the BST aggregation, leading on an important improvement of the forwarding efficiency.

In the case of using D-MPSS as the inner BF technique, we have that the total overhead given by

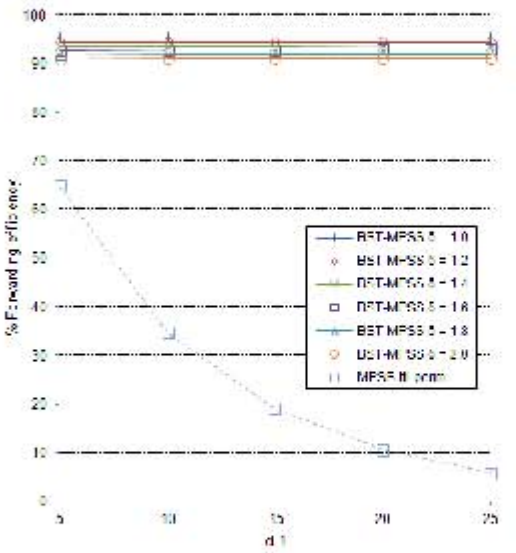
$$\begin{aligned} totalOverhead_{BST-DMPSS} = \quad & fpr_1 \cdot fanout(1)_{BST} + fpr_1 \cdot fpr_2 \cdot fanout(1)_{BST} \cdot fanout(2)_{BST} + \cdots \\ & \cdots fpr_1 \cdots fpr_2 \cdot fanout(1)_{BST} \cdots fanout(h)_{BST} \end{aligned}$$

$$totalOverhead_{BST-DMPSS} = \sum_{i=1}^{h} \left(\prod_{j=1}^{i} fpr_j \cdot fanout(j)_{BST} \right) \tag{4.51}$$

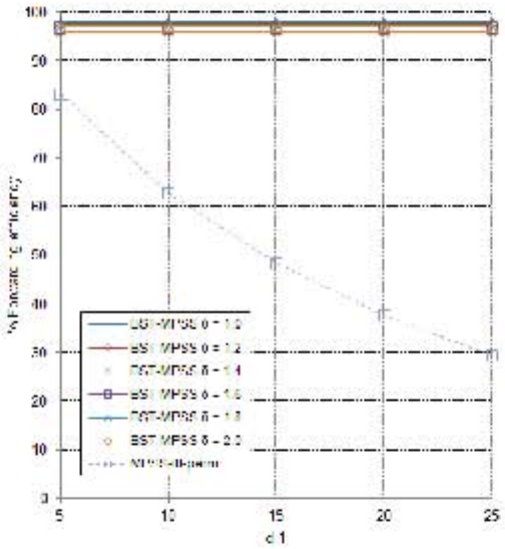
Let us consider the same multicast tree described previously. Fig. 4.29 provides the analyticial results of the *forwarding efficiency* for BST-DMPSS, with different values of d and δ, and comparing it with D-MPSS. In general, hybrid BST aggregation makes the forwarding effciency be higher than 99%, outperforming native D-MPSS and BST-MPSS methods.



(a) m=512, k=3



(b) m=800, k=4

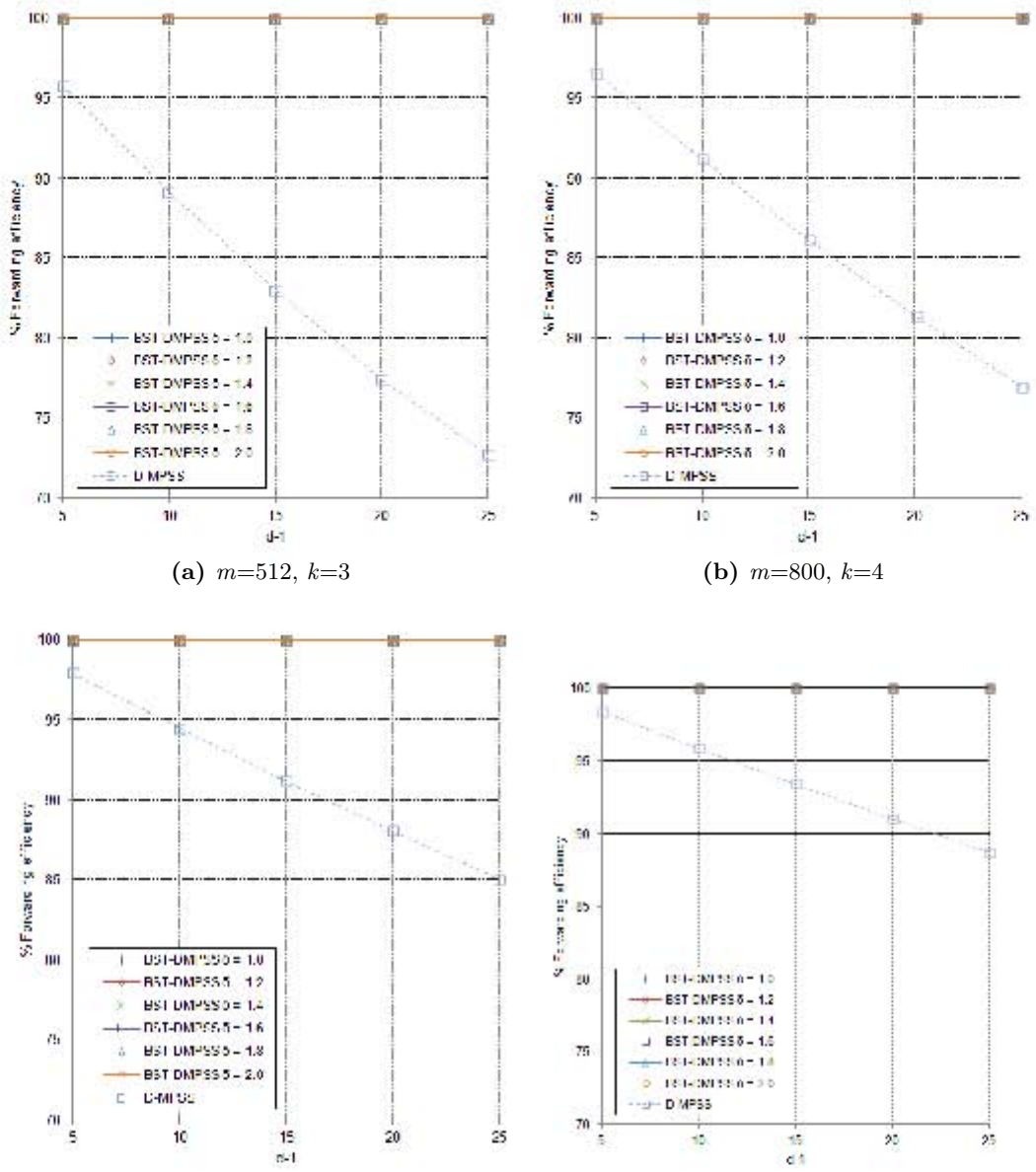


(c) m=1024, k=6

Figure 4.28: Forwarding efficiency of BST-MPSS for a multicast tree with $v = 2$, $h = 6$ over a regular network with a constant node degree of d and different values of δ (multiplier factor of v to determine the number of nodes N).

4.3.2.2 Number of Forwarding Entries

It is not possible to calculate analytically the number of forwarding entries for H-ABF techniques that involve the use of the *fit shared trees* (FST-MPSS and FST-



(a) m=512, k=3 **(b)** m=800, k=4

(c) m=1024, k=6 **(d)** m=1024, k=6

Figure 4.29: Forwarding efficiency of BST-DMPSS for a multicast tree with $v = 2$, $h = 6$ over a regular network with a constant node degree of d and different values of δ (multiplier factor of v to determine the number of nodes N).

DMPSS), since the creation of STs is not a trivial task (the RP of an ST is actually the core node that gives the average shortest distance in number of hops from the RP to the rest of nodes). Also the state entries for the LSP from the source PE node to the RP node must considered.

However, in the case of BST-MPSS and BST-DMPSS this formulation is feasible. Let $|PA|$ be the total number of BSTs built (one per each PA aggregation node as RPs) and E the total number of unidirectional links (which gives the number of forwarding entries of the inner native Bloom filter-based technique used). The total number of state entries over the network is

$$totalState_{BST} = |PA| \cdot (N-2) + E \tag{4.52}$$

4.3.2.3 Header Overhead

The header overhead is calculated in a similar manner to Eq. 4.23, with the only difference that now the 32 bits of the MPLS label (for the signaling of the ST or the BST) has to be added. Thus, with the same regular network and the same regular tree we have that

$$headerOverhead_{H-ABF} = (m+32) \cdot n \tag{4.53}$$

For the hybrid *aggregation-D-MPSS technique*, the formulation presented in Eq. 4.41 needs to be reformulated, obtaining:

$$headerOverhead_{H-ABF} = \sum_{i=2}^{h} \left(32 \cdot n_i + (m_i + s) \sum_{j=1}^{i-1} n_j \right) \tag{4.54}$$

4.3.3 Simulations Results

The simulations have been run over the same diverse and real network scenarios than in sec. 4.1 and sec. 4.1. The network topologies used are listed in Tab. 4.1, extended according to the provider aggregation model (Fig. 4.3) and described in sec. 4.1.2. A set of 1,000 random VPN samples with uniform distributions have been generated for the following H-ABF techniques:

- *Fit shared trees* with MPSS (FST-MPSS)
- *Fit shared trees* with D-MPSS (FST-DMPSS)
- *Broadcast shared trees* with MPSS (BST-MPSS)
- *Broadcast shared trees* with D-MPSS (BST-DMPSS)

A set of 1,000 random VPN samples are created, and the final results are the average of 10 iterations (giving a total of 10,000 VPNs created). Simulations were run under Java 1.7.0_03, using the IDE Eclipse Juno. Native MPSS has been implemented with the bit permutation method describbe in [SRA+11] but not including the *time-to-live* (TTL) field in the header, since it is not necessary in an environment where

shared trees are used (because any packet storm would be stopped as soon as it reaches one of the ST leaves, signaled by the MPLS labels). Since the FST method is not source-rooted, there is a bandwidth wasted for sending packets from the PE sources to the rendesvouz point node (RP) of the shared tree, that is taken into account.

4.3.3.1 Bandwidth Used and Header Overhead

Fit Shared Trees with MPSS (FST-MPSS)

In Fig. 4.30 we can see that although the FST hybrid approach reduces (and for low levels of AR, eliminates) the bandwidth wasted after false positives, it has the inconvenient of using bandwidth to deliver packets from PE sources to RP nodes of the FSTs. However, it has to be pointed out that even for high values of AR, where only a few STs are built (therefore, less state is needed), these approach outperforms the MPSS native implementation (with permutation), specially when short and medium-sized filters are used (e.g. $m = 256$ and $m = 384$). Therefore, FST-MPSS represents a viable alternative when the global percentage of header overhead means an issue for the network performance and it is necessary to lower it, because in this cases it does not represent more than 5% of the traffic.

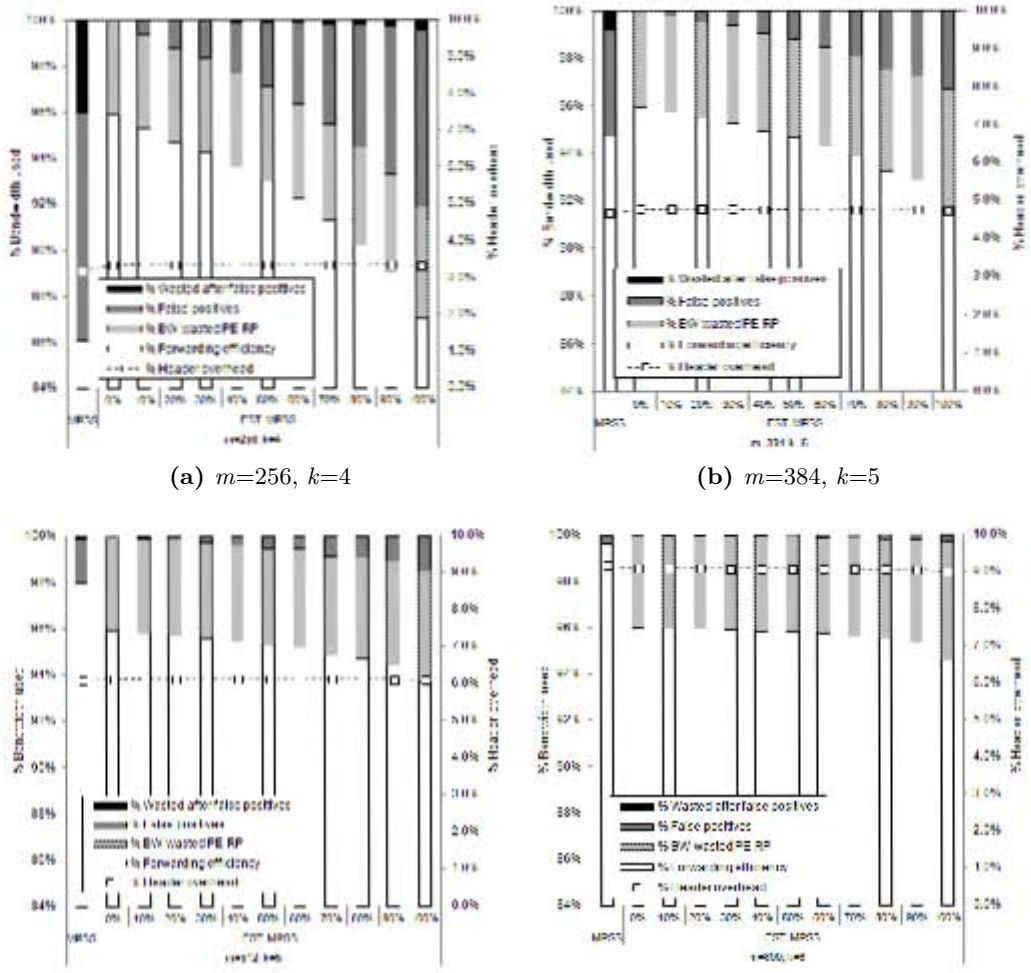
In general, as it was expected from the analytical formulations (Eq. 4.50), although fpr does not change in the shared trees scenario, the overhead that follows the false positives are minimized and almost withdrawn. As explained before, it happens because the fanout at each node for the Bloom filter matching test is reduced to the number of branches of the shared tree. For an easier comparison of these techniques we provide a summary of the forwarding efficiency metric in Fig. 4.31.

Fit Shared Trees with D-MPSS (FST-DMPSS)

Results of the hybrid FST-DMPSS technique simulations are shown in Fig. 4.32. As native D-MPSS is more effective than the MPSS-perm (as shown in sec. 4.2), this technique shows better general results in every item evaluated. However, FSTs does not seem to contribute to improve native D-MPSS here. Note that in spite of eliminating the bandwidth wasted after false positives and the amount of false positives (specially for low rates of AR), the packets used for PE-RP transmission impoverish the forwarding efficiency results. Fig. 4.31 shows a joint overview of the forwading efficiency of these hybrid D-MPSS techniques, that contributes to depict this behavior.

Broadcast Shared Trees with MPSS (BST-MPSS)

Unlike FST, it is expected that the BST model contributes to save the bandwidth wasted for PE-RP transmissions, because BST roots are located at one hop of distance from PE nodes. According to Fig. 4.34, this is effectively shown. Observe that



(a) m=256, k=4 **(b)** m=384, k=5

(c) m=512, k=5 **(d)** m=800, k=6

Figure 4.30: Bandwidth used and header overhead of the MPSS and FST-MPSS approaches (NSFNET network).

the hybrid technique means a real contribution for the reduction of useless packets of its correspoing native Bloom fiilter-based implementation, specially the ones wasted after the ocurences of false positives.

Broadcast Shared Trees with D-MPSS (BST-DMPSS)

The results gathered regarding this technique follow the same behavior as the previous one, with the only difference that, since the inner Bloom filter-based method (D-MPSS) performs better (better forwarding efficiency and header overhead), the global results are also improved with the inclusion of broadcast shared trees.

Notice that, from all of the above techniques tested, this one performs the best,

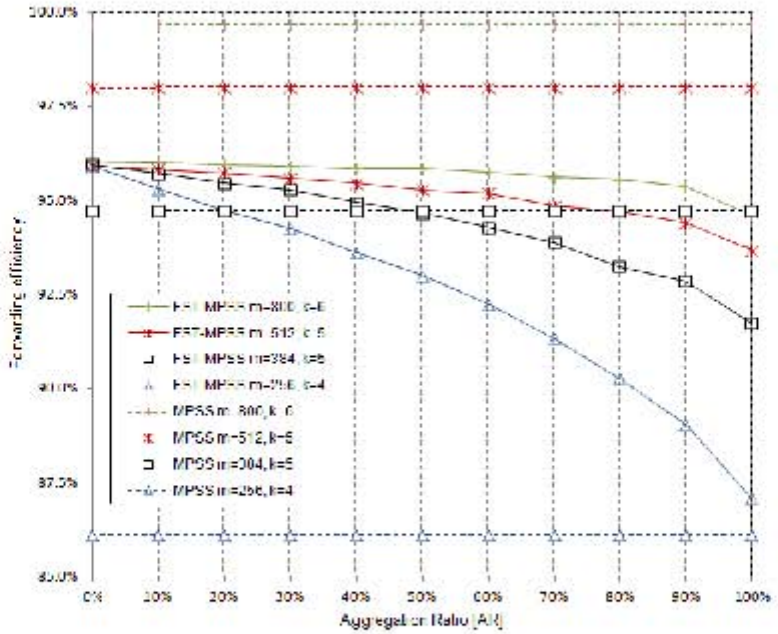


Figure 4.31: Forwarding efficiency of the FST-MPSS and native MPSS with permutation (NSFNET).

regarding both forwarding efficiency and the load of header overhead. Thus, BST-DMPSS with $M = 1024$, $k = 6$, and $mult = 32$ may be used by Internet Service Providers to support multicast communications with only less than 1% of bandwidth wasted and 4% of header overhead.

4.3.3.2 Forwarding State Entries

The number of state forwarding entries at intermediate nodes to forward the whole set VPNs samples through shared trees is summarized in Fig. 4.36 and Fig. 4.37. The figures also presents the state amount for MPSS and D-MPSS cases, which are a expected very low (each node has to maintain one entry per each outgoing link).

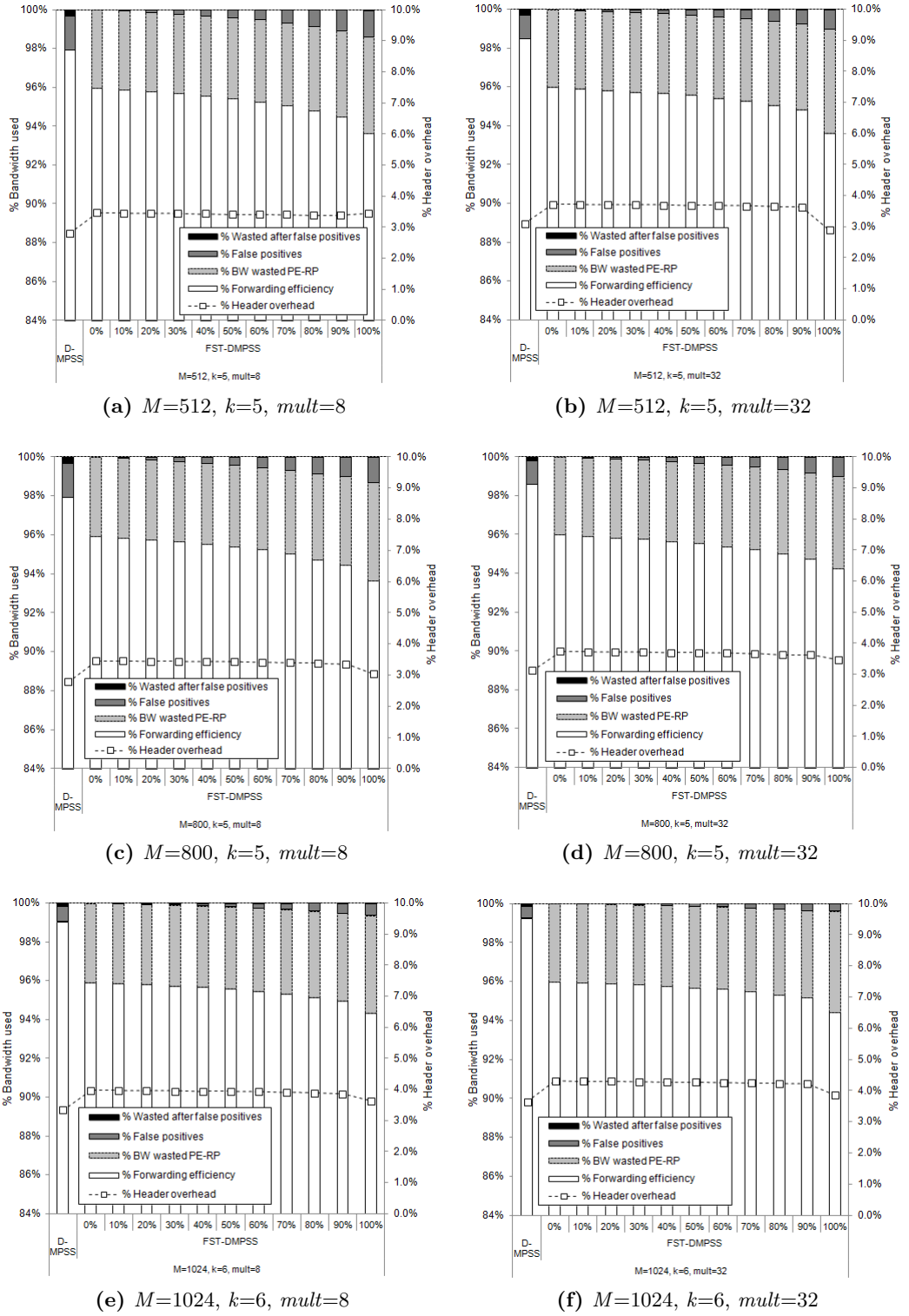
As we can see, FST show acceptable results from $AR = 80\%$, which would make the bandwidth anf forwarding efficiency metrics get worse. Also observe that the BST model does show a very low amount level of state too (in this simulation, below 20 entries per node in average), as it was expected by the analytical results (Eq. 4.52). In this sense, the BST provides a good trade-off between the forwarding efficiency, header overhead and state information.

4.3.4 Conclusion

In this section, we have presented the hybrid aggregation - Bloom filter-based forwarding approach for multicast traffic (H-ABF) with two models for the construction of shared trees: fit shared trees (FST) and broadcast shared trees (BST).

Although the contribution of the aggregation make native Bloom filter approaches (MPSS and D-MPSS) to improve their forwarding efficiency results, FST presents the counterback of having to use an important amount of links for sending packets from the provider edge nodes (PE nodes) to their corresponding shared tree roots (which realizes the multicast transmission). Another important issue of this model is the amount of forwarding entries it needs in most of the cases.

In order to overcome this difficulties, the BST model of aggregation was proposed, which constructs broadcast shared trees with their rendesvouz points (RP) near the PE nodes, in order to save PE-RP bandwidth. Results have shown that this aggregation method indeed contributes to improve native techniques, and also bounds the state information up to very low levels.



(a) M=512, k=5, $mult$=8 **(b)** M=512, k=5, $mult$=32

(c) M=800, k=5, $mult$=8 **(d)** M=800, k=5, $mult$=32

(e) M=1024, k=6, $mult$=8 **(f)** M=1024, k=6, $mult$=32

Figure 4.32: Bandwidth used and header overhead of the D-MPSS and A-DMPSS approaches (NSFNET network).

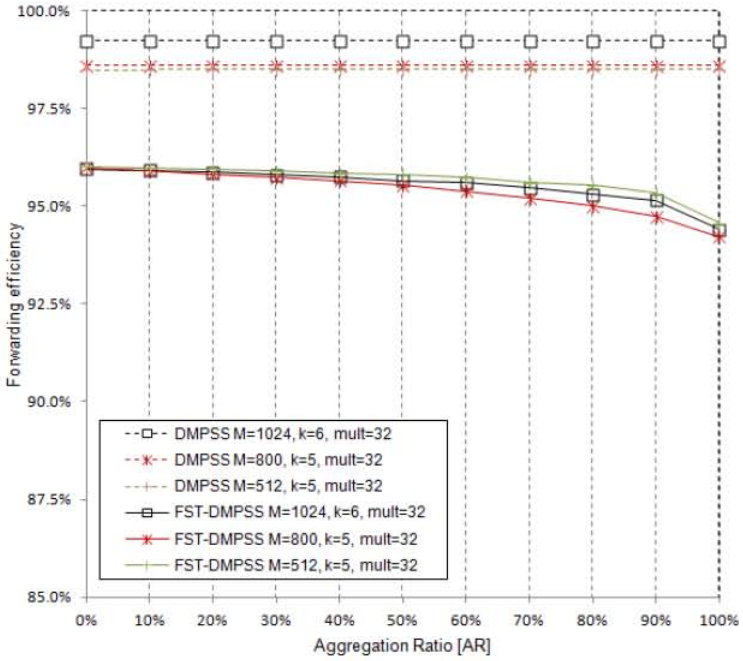


Figure 4.33: Forwarding efficiency of the FST-DMPSS and native D-MPSS techniques (NSFNET network).

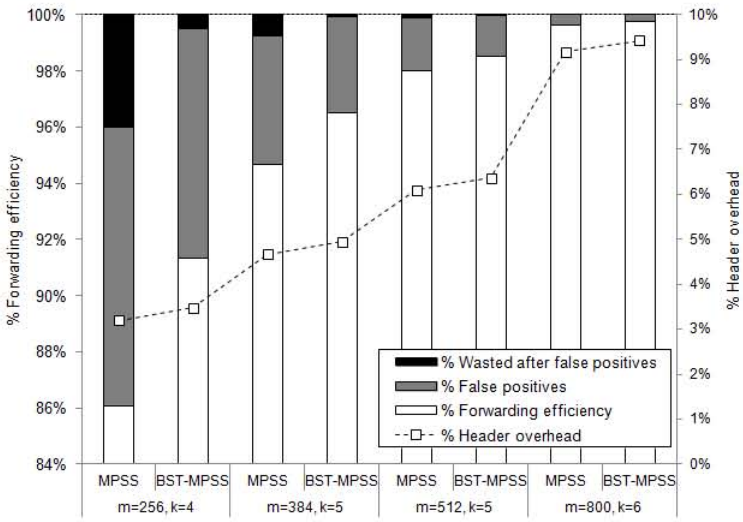


Figure 4.34: Bandwidth used and header overhead of the BST-MPSS and native MPSS with permutations techniques (NSFNET network).

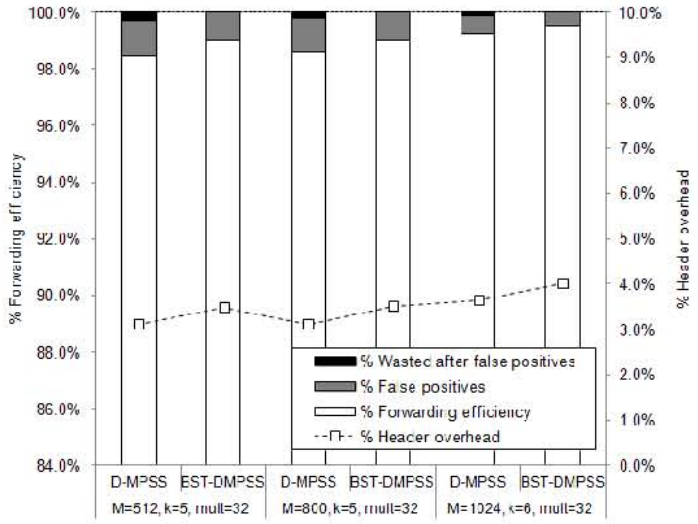


Figure 4.35: Bandwidth used and header overhead of the BST-DMPSS and D-MPSS techniques (NSFNET network).

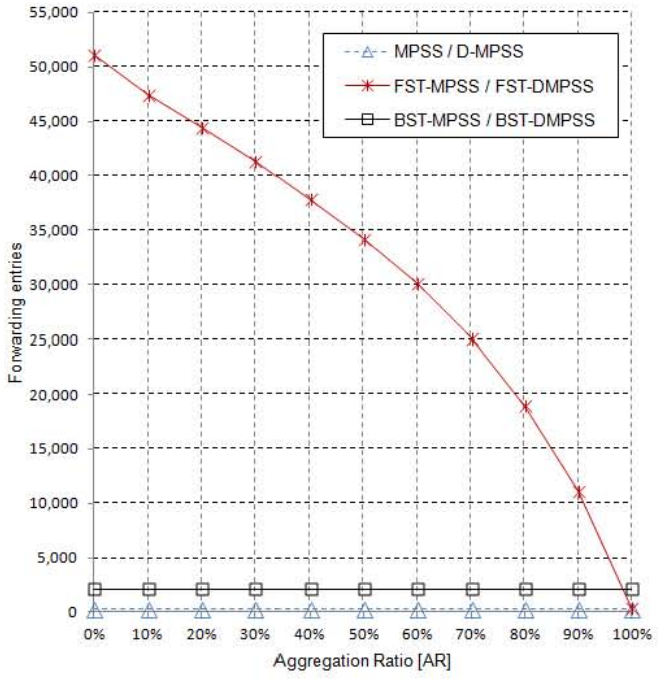


Figure 4.36: Total number of state forwarding entries for native and H-ABF techniques (NSFNET network).

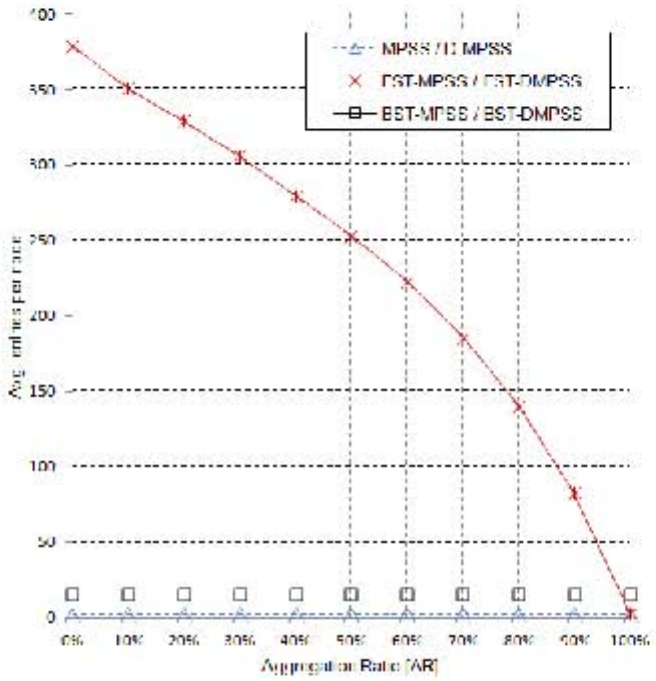


Figure 4.37: Average entries per node for native and H-ABF techniques (NSFNET network).

5 Conclusions: Outline of Contributions and Future Work

In this chapter we conclude this Ph.D thesis with the presentation of the conclusions. For this purpose, we outline the contributions (i.e. technical papers published) that have been realized while developing our work. We also present some important ideas for research that would be important to be done to follow up this work.

5.1 Optical Layer

1. We have presented a multicast capable optical cross connect (MC-OXC) node called 2-STC (*2-Split-Tap-and-Continue*), that performs TaC (tap-and-continue) and Ta2S (tap-and-2-split) operations to support multipoint transmissions in WDM networks. The design shows a relevant improvement in power efficiency when used in tap-and-continue mode w.r.t. to the tap-shared TaC architecture. When used in tap-and-2-split mode, the 2-STC combines the properties of both TaC and SaD in order to reduce the need for optical amplification w.r.t. SaD with lesser transmission resources than TaC. The 2-STC is based on 2-STCM modules whose main component is a novel tap-and-2-split switch (Ta2S) which has been designed using MMI taps and MZI switches. The work focuses on the analysis of power loss in the three architectures under study using a similar integrated optics implementation. The results show good scalability of 2-STC in terms of components used and power loss. The simulations illustrate that the binary-splitting constraint of 2-STC has a reduced impact on optimality. Indeed, the network can serve any multicast request in typical network topologies at a small additional link cost, thus achieving an interesting trade-off between power efficiency and usage of network resources.
2. On the other hand, we have extended the 2-STC node for providing the capability of traffic grooming, which takes advantage of the available bandwidth at wavelength-links. This work consists of the proposal of the 2-split-tap-and-continue traffic grooming MC-OXC node (2-STCg node). This extends the 2-STC node by adding the capability to perform multicast routing at sub-wavelength levels. As expected, the resulting architecture allows to take full advantage of the wavelengths capacity.

Further research may be focused on the development of multicast routing heuristics for both lambda and sub-lambda demands. In order to compare their efficiency

with respect to the optimal, integer linear programming (ILP) formulations should be done. Note that ILP calculations are not efficient regarding the execution time and it is not realistic to use them for real purposes, however they are necessary for measuring the distance to the optimal of any heuristic approach.

Given its grooming features, more advantages can be taken from the 2-STCg node, such as the realization of bidirectional trees, emulation of light-trails, etc. In this sense, these future 2-STCg applications -and their corresponding routing algorithms- should be studied and compared against other similar proposals found in the literature.

Published contributions regarding the optical layer are the following:

Tap-and-2-Split Switch Design Based on Integrated Optics for Light-Tree Routing in WDM networks [FVLL09], *Fernández, G. M., Contreras,P., Vazquez, C., Larrabeiti, D., IEEE Journal of Lightwave Technology (JCR Impact Factor: 2.196), vol. 27, issue 13, pp. 2506-2517, July 1, 2009, ISSN: 0733-8724.*

Power-cost-effective node architecture for Light-tree routing in WDM networks [FLVL08], *Fernández, G. M., Larrabeiti, D., Vazquez, C., Contreras, P., Proceedings of Global Telecommunications Conference 2008 (IEEE GLOBECOM 2008), Page(s): 1 - 6, Nov. 30 2008-Dec. 4 2008, New Orleans, LO, USA, ISSN: 1930-529X, ISBN: 978-1-4244-2324-8*

Contributions for all-optical multicast routing in WDM networks [FL09], *Fernández, G. M., Larrabeiti, D., XVI Congreso Internacional de Ingeniería Eléctrica, Electrónica y Sistemas (IEEE INTERCON 2009) (Invited Paper), Aug. 10* 2009-Aug. 14 2009, Arequipa, Perú.

2-STCg Optical Multicast Traffic Grooming Node for the Fishbone-Like Peruvian WDM Core Network [LF11], *Llerena, N. C., Fernández, G. M., 2nd. National Conference on Telecommunications 2011 (IEEE CONATEL 2011), Page(s): 1 - 6, May. 17 2011-May. 20 2011, Arequipa, Perú, ISBN: 978-1-4577-1046-9.*

5.2 Network Layer

1. In order to improve the scalability of the multicast provision at the network layer, we have analysed different new aspects of multicast traffic aggregation in an MPLS-based VPN network that can be a useful input to create automatic tools for multicast VPN traffic engineering, and for design and deployment purposes. In shared trees, although some bandwidth is wasted because a fraction of the packets are delivered to non-member leaves (either not in the

VPN broadcast or multicast group), there is a wide working range where a fair state vs. bandwidth trade-off is achieved. We enhanced and improved previous works that analyze this trade-off. We proposed new techniques for multicast traffic aggregation of VPNs in MPLS-based networks, with the objective of observing the behavior of the aggregation philosophy for different aggregation degrees, which are very useful for network design and deployment purposes. We assessed the aggregation heuristics over different reference networks and VPN geographic distributions. Simulations has given a quantitative indication of the relevance of intelligent aggregation, of geographical distribution and group sizes.

2. Secondly, with the aim to reduce the state due to multicast forwarding, we have proposed and studied a new method to encode multicast trees and improve the forwarding efficiency of traffic delivery under the family of stateless Bloom filter-based distribution mechanisms, such as MPSS. Our proposal looks into improving previous works and at the same time to solve the forwarding anomalies observed in them. We propose to encode the multicast tree into a stack of variable-length Bloom filters representing the set of output interfaces at a given tree depth, instead of a single filter for the whole tree. The results show that our proposal, called D-MPSS (depth-wise multi-protocol stateless switching), is an effective method to get rid of forwarding anomalies while achieving high transmission efficiency, and that it outperforms the best known variant of the MPSS concept up to date. Although the chosen scenario in this work is that of the multicast MPLS VPN-based networks present also in the MPSS technique, this mechanism could be applicable for other types of scenarios, such as inter-domain publish-subscribe networks.

3. Finally, we have developed hybrid aggregation - Bloom filter solutions for the improvement of the forwarding efficiency and the reduction (and practically in some cases, the elimination) of most of the forwarding anomalies typically present in BF-based forwarding techniques. We have explored the possibility of realizing multicast forwarding by combining aggregation with Bloom filters. It has been observed that the reduction of the number of outgoing links to be evaluated at each node diminishes the overhead traffic.The really high forwarding efficiency percentage achieved with these solution is really important and means a sifnificant advance w.r.t. the state-of-the-art approaches.

Regarding the multicast aggregation, new improved heuristics (compared against optimal ILP formulations) can be proposed, directed to reduce the bandwidth wasted. Also, some other important challenges arise for the Bloom filter-based forwarding mechanisms, such as security vulnerabilities that require further study. Finally, the future work in the H-ABF line could be to study of optimization of the number and the construction of broadcast shared trees.

Published contributions regarding the optical layer are the following:

On Forwarding State Control in VPN Multicast Based on MPLS Multipoint LSPs [FLdlF12], *Fernández, G. M., Larrabeiti, D., De-la-Fuente, J. A., 2012 IEEE Conference on High Performance Switching and Routing (IEEE HPSR 2012), Page(s): 133 - 140, Jun. 24 2012 - Jun. 27 2012, Belgrade, Serbia, ISBN: 978-1-4577-0831-2.*

Depth-Wise Multiprotocol Stateless Switching for Multicast Traffic [FL12], *Fernández, G. M., Larrabeiti, D., 2012 IEEE Latin-American Conferences on Communications (IEEE Latincom 2012), (Accepted for publication), Nov. 7 2012 - Nov. 9 2012, Cuenca, Ecuador.*

Hybrid Aggregation - Bloom Filter-based Approach for Multicast Traffic with Elimination of Forwarding Anomalies *Fernández, G. M., Larrabeiti, D., Journal of Computer and Telecommunications Networking, (submitted for peer review), Oct. 2012.*

Acknowledgements

I would like to thank professor Dr. David Larrabeiti for his extraordinary support, and professors Dr. Carmen Vázquez and Dr. Pedro Contreras for their unvaluable contribution to this Ph.D. thesis.

Thanks to my lovely wife, my parents, brothers and friends.

Thanks to my colleagues and my students.

This work was realized thanks to the joint Ph.D. scholarship provided by Carolina Foundation (Spain) and Universidad Católica San Pablo (Peru), and with the support of the ADSCOM group of Universidad Carlos III de Madrid.

Bibliography

[AC05] G. Apostolopoulos and I. Ciurea. Reducing the forwarding state requirements of point-to-multipoint trees using mpls multicast. In *Proceedings of the IEEE International Symposium on Computers and Communiations (ISCC)*, pages 713–718, 2005.

[Act11] 2011 annual report. Technical report, Activision Blizzard Inc., 2011.

[AD00a] M. Ali and J. Deogun. Cost-efficient implementation of multicasting in wavelength-routed networks. *IEEE/OSA Journal of Lightwave Technology*, 18(12): 1628–1638, 2000.

[AD00b] M. Ali and J. Deogun. Power-efficient design of multicast wavelength-routed networks. *IEEE Journal on Selected Areas in Communitations*, 18(10): 852–862, 2000.

[AKF12] R. Aggarwal, Y. Kamite, and L. Fang. Multicast in vpls. Internet draft draft-ietf-l2vpn-vpls-mcast-10.txt, standards track, IETF, February 02 2012.

[AMBM07] N. Ben Ali, J. Moulierac, A. Belghith, and M. Molnar. mqma: Multi-constrained qos multicast aggregation. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1927–1932, Washington, DC, USA, 2007.

[Ant11] Markku Antikainen. On the security of in-packet bloom-filter forwarding (master thesis). aalto university. espoo, finland. Master's thesis, Aalto University, Espoo, Finland, June June 2011.

[AZS11] Norberto Amaya, Georgios S. Zervas, and Dimitra Simeonidou. Architecture on demand for transparent optical networks. In *Proceedings of the 13th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2011.

[BFI+07] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms. Explicit multicast (xcast) concepts and options. Rfc 5058 (experimental): http://tools.ietf.org/html/rfc5058, IETF, November 2007.

[BG12] Saeed Barkabi and Akbar Ghaffarpour. Mc-brm: A distributed rwa algorithm with minimum wavelength conversion. *Journal of Optical Fiber Technology*, 18(4): 230–241, 2012.

[BGK+08] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel Smid, and Yihui Tang. On the false positive rate of bloom filters. *Information Processing Letters*, 108(4): 210–213, 2008.

[Blo70] Burthon H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7): 422–426, 1970.

[BM05] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Journal of Internet Mathematics*, 1(4): 485–509, 2005.

[BV95] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. In IEEE, editor, *Proceedings of the IEEE INFOCOM*, pages 369–376, Boston, MA, USA, 1995. IEEE.

[BWA03] A.R.B. Billah, Bin Wang, and A.A.S. Awwal. Multicast traffic grooming in wdm optical mesh networks. In IEEE, editor, *Proceedings of the IEEE Global Telecommunications Conference (IEEE Globecom)*, volume 5, pages 2755–2760, San Francisco, USA, 2003. IEEE.

[CC00] F.Y.Y. Cheng and R.K.C. Chang. A tree switching protocol for multicast state reduction. In *Proceeding of the 5th IEEE Symposium on Computers and Communications (ISCC)*, pages 672–677, Antibes-Juan les Pins, France, 2000.

[CFGF01] J. H. Cui, A. Fei, M. Gerla, and M. Faloutsos. An architecture for scalable qos multicast provisioning. Technical report, UCLA CSD Technical Report Num.010030, August 2001 2001.

[CM04] G.V. Chowdhary and C. S. Ram Murthy. Grooming of multicast sessions in wdm mesh networks. In *Proceedings of the Workshop on Traffic Grooming*, Dpt. Computer Science & Engineering, Indian Institute of Technology Madras Chennai (India), 2004.

[DJ11] Li Du and Yanning Jia. A dynamic multicast rwa algorithm for wdm network with sparse splitting and wavelength conversion capability. In *Proceedings of the Electronics, International Conference on Communications and Control (ICECC)*, pages 613–616, 2011.

[DR02] Rudra Dutta and George N. Rouskas. Traffic grooming in wdm networks: Past and future. *IEEE Network*, 16(6): 46–56, 2002.

[FCGF01a] Aiguo Fei, JunHong Cui, Mario Gerla, and Michalis Faloutsos. Aggregated multicast: an approach to reduce multicast state. In *Proceedings of the 6th Global Internet Symposium (GI2001)*, 2001.

[FCGF01b] Aiguo Fei, Junhong Cui, Mario Gerla, and Michalis Faloutsos. Aggregated multicast with inter-group tree sharing. In *Proceedings of the 3rd International COST264 Workshop on Networked Group Communication*, London, 2001. Springer-Verlag.

[FL09] G. M. Fernandez and D. Larrabeiti. Contributions for all-optical multicast routing in wdm networks. In IEEE, editor, *Proceedings of the IEEE INTERCON*, Arequipa, Peru, 2009. IEEE.

[FL12] Gonzalo M. Fernandez and David Larrabeiti. Depth-wise multiprotocol stateless switching of multicast traffic. In *Proceedings of the IEEE Latin American Conference on Communications (LATINCOM)*, volume (Submitted for acceptance), Cuenca, Ecuador, 2012.

[FLC07] J. Y. Feng, T. S. Lay, and T. Y. Chang. Waveguide couplers with new power splitting ratios made possible by cascading of short multimode interference sections. *OSA Optics Express*, 15(4): 1588–|593, 2007.

[FLdlF12] Gonzalo M. Fernandez, David Larrabeiti, and Juan A. de-la Fuente. On forwarding state control in vpn multicast based on mpls multipoint lsps. In *Proceedings of the IEEE 13th International Conference on High Performance Switching and Routing (HPSR)*, Belgrade, Serbia, 2012.

[FLVL08] G. M. Fernandez, D. Larrabeiti, C. Vaquez, and P. C. Lallana. Power-cost-effective node architecture for light-tree routing in wdm networks. In IEEE, editor, *Proceedings of the IEEE Global Telecommunications Conference (Globecom)*, pages 1–6, New Orleans, LO, USA, 2008. IEEE.

[FSZ11] Jie Feng, Qinghua Shi, and Cuicui Zhang. A new ant colony based algorithm for grouptree matching in aggregated multicast. In *Proceedings of the International Conference on Computer Science and Service System (CSSS)*, pages 37 – 40, Nanjing, 2011.

[FVLL09] G. M. Fernandez, C. Vazquez, P. C. Lallana, and D. Larrabeiti. Tap-and-2-split switch design based on integrated optics for light-tree routing in wdm networks. *IEEE Journal of Lightwave Technology*, 27(13): 2506–2517, 2009.

[GFV05] Jim Guichard, Francois Le Faucheur, and Jean-Philippe Vasseur. *Definitive MPLS Network Designs*. Cisco Press, 2005.

[HCT01] Jingyi He, S.-H.G. Chan, and D.H.K. Tsang. Routing and wavelength assignment for wdm multicast networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, volume 3, pages 1536–1540, 2001.

[HL04] J.Q. Hu and Brett Leida. Traffic grooming, routing, and wavelength assignment in optical wdm mesh networks. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1, 2004.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. The steiner tree problem. *Elsevier Annals of Discrete Mathematics*, 53, 1992.

[HZ98] W. S. Hu and Q. J. Zeng. Multicasting optical cross connects employing splitter-and-delivery switch. *IEEE Photonics Technology Letters*, 10: 970–972, 1998.

[JZR+09] Petri Jokela, Andras Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. Lipsin: Line speed publish/subscribe inter-networking. In ACM, editor, *Proceedings of the ACM SIGCOMM Conference of the Special Interest Group on Data Communications*, pages 195–206, Barcelona, Spain, 2009.

[Kam06] A.E. Kamal. Design algorithms for multicast traffic grooming in wdm mesh networks. *IEEE Communications Magazine*, 44(11): 96–195, 2006.

[KHF+05] T. Kawanishi, K. Higuma, T. Fujita, J. Ichikawa, T. Sakamoto, S. Shinada, and M. Izutsu. High-speed optical fsk modulator for optical packet labeling. *Journal of Lightwave Technology*, 23(1), 2005.

[KR07] K. Kompella and Y. Rekhter. Virtual private lan service (vpls) using bgp for auto-discovery and signaling. Rfc 4761 (standard): http://tools.ietf.org/html/rfc4761, IETF, January 2007.

[Lal11] Pedro Contreras Lallana. *Advanced Devices Based on Fibres, Integrated Optics and Liquid Crystals for WDM Networks*. PhD thesis, Universidad Carlos III Madrid, May 2011.

[LF11] N. C. Llerena and G. M FernÃ¡ndez. 2-stcg optical multicast traffic grooming node for the fishbone-like peruvian wdm core network. In IEE, editor, *Proceedings of the 2nd. National Conference on Telecommunications (IEEE CONATEL)*, volume 1, pages 1–6. IEEE, IEEE, May 2011.

[LJ01] J. Leuthold and C. H. Joyner. Multimode interference couplers with tunable power splitting ratios. *IEEE/OSA Journal of Lightwave Technology*, 19(5): 700–706, 2001.

[LPYP07] Taehan Lee, Kyungchul Park, Jaekyung Yang, and Sungsoo Park. Optimal multicast routing and wavelength assignment on wdm ring networks without wavelength conversion. *IEEE Communications Letters*, 11(11): 898–900, 2007.

[LR07] M. Lasserre and Y. Rekhter. Virtual private lan service (vpls) using label distribution protocol (ldp) signaling. Rfc 4762 (standard): http://tools.ietf.org/html/rfc4762, IETF, January 2007.

[LZY08] Ling-zhi Li, Yan-qin Zhu, and Zhe Yang. Deploying bidirectional multicast shared trees in mpls networks. In *Proceedings of the International Conference on Computer and Electrical Engineering (ICCEE)*, pages 276–280, Phuket, 2008.

[MCL+03] Sophie De Maesschalck, Didier Colle, Ilse Lievens, Mario Pickavet, Piet Demeester, Christian Mauz, Monika Jaeger, Robert Inkret, Branko Mikac, and Jan Derkacz. Pan-european optical transport networks: An availability-based comparison. *Photonic Network Communications*, 5(3), 2003.

[MGM06a] J. Moulierac, A. Guitton, and M. Molnar. Cam02-5: On the number of mpls lsps using multicast tree aggregation. In *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, pages 1–5, San Francisco, CA, 2006.

[MGM06b] Joanna Moulierac, Alexandre Guitton, and Miklos Molnar. Hierarchical aggregation of multicast trees in large domains. *Journal of Communications*, 1(6): 33–44, 2006.

[MYLS07] Isaias Martinez-Yelmo, David Larrabeiti, and Ignacio Soto. Multicast traffic aggregation in mpls-based vpn networks. *IEEE Communications Magazine*, 45(10): 78–85, 2007.

[OKY+03] Young-Kyu Oh, Dong-Keun Kim, Hun-Je Yoen, Mi-sun Do, and Jaiyong Lee. Scalable mpls multicast using label aggregation in internet broadcasting systems. In *Proceddings of the 10th IEEE International Conference on Telecommunications (ICT)*, volume 1, pages 273–280, 2003.

[PEAH10] T. Panayiotou, G. Ellinas, N. Antoniades, and A. Hadjiantonis. Node architecture design and network engineering impact on optical multicasting based on physical layer constraints. In *Proceedings of the 12th International Conference on Transparent Optical Networks (ICTON)*, 2010.

[PEAL10] T. Panayiotou, G. Ellinas, N. Antoniades, and A.M. Levine. Designing and engineering metropolitan area transparent optical networks for the provisioning of multicast sessions. In IEEE, editor, *Proceeding of the 2010 Conference on Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference (OFC/NFOEC)*, pages 1–3, 2010.

[RA12] E. Rosen and R. Aggarwal. Multicast in mpls/bgp ip vpns. Rfc 6513 (standard): http://tools.ietf.org/html/rfc6513, IETF, February 2012.

[RES06] Sylvia Ratnasamy, Andrey Ermolinskiy, and Scott Shenker. Revisiting ip multicast. In *Proceedings of the ACM SIGCOMM Conference of the Special Interest Group on Data Communications*, pages 15–26, New York, NY, USA, 2006.

[RGE99] P. I. Radoslavov, R. Govindan, and D. Estrin. Exploiting the bandwidth-memory tradeoff in multicast state aggregation, technical report. Technical report, USC Dept. of CS Technical Report 99-697 (Second Revision), July 1999 1999.

[RJN+09] C. Rothenberg, P. Jokela, P. Nikander, M. Sarela, and J. Ylitalo. Self-routing denial-of-service resistant capabilities using in-packet bloom filters. In IEEE, editor, *Proceeding of the European Conference on Computer Network Defense*, pages 46–51. IEEE, 2009.

[RMM+11] Christian Esteve Rothenberg, Carlos Alberto Braz Macapuna, Mauricio Ferreira Magalhaes, Fabio Luciano Verdi, and Alexander Wiesmaier. In-packet bloom filters: Design and networking applications. *Computer Networks*, 55(6): 1364–1378, 2011.

[Rou02] G. N. Rouskas. Light-tree routing under optical layer power budget constraints. In IEEE, editor, *Proceedings of the 17th IEEE Computer Communications Workshop*, 2002.

[Rou03] G. N. Rouskas. Optical layer multicast: Rationale, building blocks, and challenges. *IEEE Network*, (17): 60–65, 2003.

[RR06] E. Rosen and Y. Rekhter. Bgp/mpls ip virtual private networks (vpns). Rfc 4364 (standard): http://www.ietf.org/rfc/rfc4364.txt, IETF, Feb. 2006.

[RTF+01] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. Mpls label stack encoding. Rfc 3032 (standard): http://www.ietf.org/rfc/rfc3032.txt, IETF, January 2001.

[SA11] Tianping Shuai and Wenbao Ai. Approximation algorithms for multicast routing and wavelength assignment in multi-hop optical wdm networks. In *Proceeedings of the 4th International Joint Conference on Computational Sciences and Optimization (CSO)*, pages 1291–1295, 2011.

[SCS+08] J. E. Sierra, L. F. Caro, F. Solano, R. Fabregat, and Y. Donoso. S/g light-tree: Multicast grooming architecture for improved resource allocation. In IEEE, editor, *Proceedings of the IEEE 12th International Conference on Optical Networking Design and Modeling (ONDM)*. IEEE, 2008.

[SFD08] J. E. Sierra, R. Fabregat, and Y. Donoso. Modelo de costos para redes opticas wdm que emplean la arquitectura s/g light-tree. In *Proceedings of the Jornadas de Sistemas de Telecomunicaciones*, volume Ecuador, Quito, Ecuador, 2008.

[SFD09] J. E. Sierra, R. Fabregat, and Y. Donoso. Static and dynamic provisioning of unicast/multicast traffic demands using s/g light-tree in wdm optical networks. *Int. Journal Communication Networks and Distributed Systems*, 2009.

[SM99] L. H. Sahasrabuddhe and B. Mukherjee. Light-trees: Optical multicasting for improved performance in wavelength-routed networks. *IEEE Communications Magazine*, 37(2): 67–73, 1999.

[SMM06] J. Sanchez, P. Manzanares, and J. Malgosa. Spanish telco strategies facing new integrated digital transmission advances. *Global Communications Newsletter, IEEE Communications Magazine*, 44(2), 2006.

[SRA+11] Mikko Sarela, Christian Esteve Rothenberg, Tuomas Aura, Andras Zahemszky, Pekka Nikander, and Jorg Ott. Forwarding anomalies in bloom filter-based multicast. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 2399–2407, Shangai, China, 2011.

[SRZ+10] Mikko Sarela, Christian Esteve Rothenberg, Andras Zahemszky, Pekka Nikander, and Jorg Ott. Bloomcasting: Security in bloom filter based multicast. *Lecture Notes in Computer Science (15th Nordic Conference on Secure IT Systems, Revised Selected Papers)*, 7127: 1–16, 2010.

[TC12] Xiaohua Tian and Yu Cheng. Loop mitigation in bloom filter based multicast: A destination-oriented approach. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 2131–2139, Orlando, FL, USA, 2012.

[TH00] David Thaler and M. Handley. On the aggregability of multicast forwarding state. In IEEE, editor, *Proceedings of the IEEE International Confeerence on Communications INFOCOM*, volume 3, pages 1654–1663, Tel-Aviv, Israel, 2000. IEEE.

[TRL12] Sasu Tarkhoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1): 131–155, 2012.

[UMK06] R. Ul-Mustafa and A. E. Kamal. Design and provisioning of wdm networks with multicast traffic grooming. *IEEE Journal on Selected Areas in Communications, Part II: Optical Communications and Networking*, 24(4), 2006.

[VMHG95] C. Vazquez, F. J. Mustieles, and J. F. Hernandez Gil. Three-dimensional method for simulation of multimode interference couplers. *IEEE Journal of Lightwave Technology*, 13(11): 2296–2299, 1995.

[VVSP05] C. Vazquez, S. Vargas, and J. M. S. Pena. Sagnac loop in ring resonators for tuneable optical filters. *IEEE Journal of Lightwave Technology*, 23(8): 2555–67, 2005.

[VZC+03] K. Vlachos, J. Zhang, J. Cheyns, Nan Chi, E. Van Breusegem, I. Tafur Monroy, J. G. L. Jennen, P. V. Holm-Nielsen, C. Peucheret, R. O'Dowd, P. Demeester, and A. M. J. Koonen. An optical im/fsk coding technique for the implementation of a label-controlled arrayed waveguide packet router. *IEEE Journal of Lightwave Technology*, pages 2617–21, 2003.

[WGZY09] Hua Wang, Zuquan Ge, Fangji Zhu, and Chaoying Yu. An immune algorithm for the optimization of aggregated multicast. In *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT)*, volume 1, pages 786–789, 2009.

[WMKT11] Ed. IJ. Wijnands, Ed. I. Minei, K. Kompella, and B. Thomas. Label distribution protocol extensions for point-to-multipoint and multipoint-to-multipoint label switched paths. Rfc 6388 (standard): http://www.ietf.org/rfc6388, IETF, Nov. 2011.

[WSW05] W. Wattanavarakul, S. Segkhoonthod, and L. Wuttisittikulkij. Design of multicast routing and wavelength assignment in multifiber wdm mesh networks for asymmetric traffics. In *Proceedings of the TENCON 2005 2005 IEEE Region 10*, pages 1–6, 2005.

[WWY01] K. D. Wu, J. C. Wu, and C. S. Yang. Multicast routing with power consideration in sparse splitting wdm networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 513–517, Helsinki, Finland, 2001.

[WZYY09] Hua Wang, Z.G., Chaoying Yu, and Shanwen Yi. A modified genetic algorithm for the optimization of aggregated multicast. In *Proceedings of the International Conference on Advanced Communication Technology (ICACT)*, pages 83–88, 2009.

[XR04] Y. Xin and G. N. Rouskas. Multicast routing under optical layer constraints. In IEEE, editor, *Proceedings of the 23rd Conference of the IEEE Communications Society (INCOFOM)*, volume 4, pages 2731–42, Hong Kong, China, 2004.

[YKWS+09] Ed. Y. Kamite, Y. Wada, Y. Serbest, T. Morin, and L. Fang. Requirements for multicast support in virtual private lan services. Rfc 5501 (informational): http://tools.ietf.org/html/rfc5501, March 2009.

[ZJM00] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. *Optical Networks Magazine*, 1(1): 47–60, 2000.

[ZJS+10] Andras Zahemszky, Petri Jokela, Mikko Sarela, Sami Ruponen, James Kempf, and Pekka Nikander. Mpss: Multiprotocol stateless switching. In *Proceedings of the IEEE Conference on Computer Communications INFOCOM Workshops*, pages 1–6, San Diego, CA, 2010.

[ZP05] Y. Zhou and G-S. Poo. Optical multicast over wavelength-routed wdm networks: A survey. *Optical Switching and Networking*, (1), 2005.

[ZPSZ08] Y. Zhou, S. Poo, G.-S.and Chen, P. Shum, and L. Zhang. Dynamic multicast routing and wavelength assignment using generic graph model for wavelength-division-multiplexing networks. *Journal of IET Communications*, 2(7): 951–959, 2008.

[ZW10] Fangjin Zhu and Hua Wang. An aco algorithm to tackle aggregated multicast problem. In *Proceedings of the IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT)*, pages 53–56, 2010.

[ZW11] Fangjin Zhu and Hua Wang. A modified aco algorithm for multicast state scalability problem based on multicast tree similarity. In *Proceedings of the International Conference on Advanced Communication Technology (ICACT)*, pages 972–976, Seoul, 2011.

[ZXWY09] Fangjin Zhu, X.M., Hua Wang, and Shanwen Yi. An ant colony optimization algorithm to aggregated multicast using the idea of bin packing. In *Proceedings of the IEEE Youth Conference on Information, Computing and Telecommunication (YC-ICT)*, pages 194–197, Beijing, 2009.

Nomenclature

2-STC node	2-Split-Tap-and-Continue node
2-STCg node	2-Split-Tap-and-Continue Traffic Grooming Node
2-STCM	2-Split-Tap-Continue Module
ACO	Ant Colony Optimization
AoD	Architecture on Demand
AOMTG	All-Optical Multicast Traffic Grooming
BPM	Beam Propagation Method
BST	Broadcast Shared Tree
COST-266	COST-266 Optical Transport reference network (http://www.ibcn.intec.ugent.be/INTERNAL/NRS/index.html)
D-MPSS	Depth-Wise Multi-Protocol Stateless Switching
DWDM	Dense Wavelength Division Multiplexing
DXC	Digital Cross Connect
e-SaD	Electrical Split-and-Deliver
FDL	Fiber Delay Line
FST	Fit Shared Tree
H-ABF	Hybrid Aggregation - Bloom Filter-based Forwarding
HDTV	High-Definition Television
ILP	Integer Linear Programming
Light-tree	All-optical point-to-multipoint tree
Lightpath	All-optical point-to-point path

LIT	Link ID Tag
LSP	Label Switched Path
LTE	Line Terminal Equipment
MC-OXC	Optical multicast capable OXC node
MC-RWA	Multicast routing and wavelength assignment
MEMS	Micro-Electro-Mechanical switches
MMI	Multi Mode Interference
MOSPF	Multicast Open Shortest Path First
MPLS	Multi-Protocol Label Switching
MPSS	Multi-Protocol Stateless Switching
MVPN	Multicast Virtual Private Network
ns-TaC	Non-shared-tap TaC
o-SaD	Optical Split-and-Deliver
O-SAN	Optical Storage Area Network
ORR	Optical Ring Resonators
p2p	Point-to-Point
PIM-DM	Protocol Independent Multicast with Dense Mode
RP	Rendesvouz Point
S/G Lightree	Stop-and-Go Lightree
SaD	Split and Delivery
SaD switch	Split-and-delivery switch
SC-RR	1x2 Switch - 2x1 Coupler made with Optical Ring Resonators
ST	Shared Tree
Ta2S switch	Tap-and-2-Split switch
TaC node	Tap-and-continue optical node

TCM	Tap-and-continue module
VOA	Variable Optical Attenuators
VoD	Video-on-Demand
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
WDM	Wavelength Division Multiplexing
WRS	Wavelength routing switching