Planningsalgoritmen voor optische pakket- en burstgeschakelde netwerken

Scheduling Algorithms for Optical Packet and Burst Switched Networks

Kurt Van Hautegem

Promotoren: prof. dr. ir. W. Rogiest, prof. dr. ir. H. Bruneel
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen

UNIVERSITEIT
GENT

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. J. Walraevens
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2018 - 2019

ISBN 978-94-6355-167-0
NUR 986
Wettelijk depot: D/2018/10.500/85

Members of the examination board

Chair

em. prof. dr. ir. Hendrik Van Landeghem (Ghent University)

Reading committee

prof. dr. Jacques Resing (Eindhoven University of Technology)
prof. dr. Chris Blondia (University of Antwerp)
prof. dr. ir. Sabine Wittevrongel (Ghent University)
prof. dr. ir. Chris Develder (Ghent University)
dr. Bart Steyaert (Ghent University)

Other members

prof. dr. ir. Wouter Rogiest (Ghent University)
prof. dr. ir. Herwig Bruneel (Ghent University)

Dankwoord

Er zijn maar weinig zaken die ik kan of wil tot mijn diepste persoonlijke overtuigingen rekenen. Eén ervan is echter zonder twijfel dat er in een mensenleven, in vergelijking met spelingen van het lot en omgevingsfactoren, maar zelden veel persoonlijke verdienste is aan welk bereikt succes dan ook. Door haar geheugenloze en willekeurige karakter, hoeft het lot geen dank. De omgeving heeft er des te meer recht op. Aangezien het dankwoord bovendien niet zelden het meest gelezen stukje tekst van een proefschrift is, verdienen deze eerste woorden dan ook de nodige aandacht. De lijst is lang, uw aandacht kort, laat ons er dus maar aan beginnen.

De herinnering is nog scherp, dat zegt al iets. Het was op een zonnige zomerdag, na een stevige nacht op de Gentse Feesten, dat ik verwacht werd op Herwig zijn bureau. Enkele weken eerder had hij mij een e-mail gestuurd om te vragen of ik geïnteresseerd was om te doctoreren onder zijn supervisie. Aangezien ik al een job op het oog had, was het eerder uit beleefdheid, en op aanraden van mijn moeder, dat ik toch eens langsging. Een beginnersfout, zo ging ik snel leren, voor zij die niet op de hoogte zijn van de man zijn overtuigingskracht.

Zes jaar later ben ik enorm dankbaar voor de kans die ik toen kreeg. Herwig heeft in zijn onderzoeksgroep een unieke werkomgeving gecreëerd waar iedereen zich, omringd door fantastische collega's en met een ongeziene persoonlijke vrijheid, intellectueel kan ontplooien. Ook al is Herwig zowat het tegenvoorbeeld van een micromanager, toch kon hij bij het finale nalezen van de vele teksten zonder de minste moeite op kleine en grotere fouten wijzen. Het was mij vaak een mysterie hoe ik er zelf over gelezen had. Met een zelfde, schijnbaar kinderlijke, naïviteit kan hij de vinger op de wonde leggen met eenvoudige maar prangende vragen tijdens eender welke presentatie. Deze en nog veel andere zaken maken dat Herwig zijn invloed en persona nog lang zullen blijven nazinderen op TELIN en bij iedereen die er ooit passeerde.

Waar Herwig me meer van op afstand begeleidde en gidste, was Wouter dichter betrokken bij mijn verwezenlijkingen. Ik kan zelf met vrij grote zekerheid zeggen dat vele ervan zonder zijn begeleiding niet tot stand waren gekomen. Zelfs nadat hij koos voor een carrière tussen de patenten bleef hij mijn papers, en dit werk, met de grootste zorg nalezen en verbeteren. Een cruciale stap, weet iedereen die ooit een wetenschappelijke bijdrage publiceerde. Met onze bijdragen voor Eos, Knack en andere media bereikten we dan weer een veel groter publiek, tot over de landsgrenzen heen. Deze zaken behoren misschien niet tot mijn oorspronkelijke verwachtingen van een doctoraat maar ze vormen toch meer dan het peper en het zout van onze samenwerking. Ik heb er in ieder geval ontzettend veel uit geleerd en zonder Wouter als drijvende kracht, had ik het nooit ontdekt. Dit alles overpeinzend hoop ik nog zo'n mentors tegen te komen in mijn carrière en als ik er zelf ooit één ben, zal ik zonder twijfel nog vaak aan hem terugdenken als voorbeeld.

Omgevingsfactoren gaan uiteraard veel ruimer dan de professionele omkadering. Ook vrienden huppelden vrolijk door mijn leven in de afgelopen jaren en verdienden zo hun plaatsje in dit dankwoord. Als ik over vriendschap mijmer, komt deze zeemzoete quote[1] - u vergeeft me hopelijk enige pathos - al te vaak terug: "Understand that friends come and go, but with a precious few you should hold on". Geen idee wanneer die precious few in mijn leven komen, maar ondertussen stel ik me met veel plezier tevreden met het zootje ongeregeld dat er voor moet doorgaan. Wat ooit geïntoxiceerde nachten in de krochten van de Overpoort en last-minute verplaatsingen naar Leuven waren, zijn ondertussen spelletjesavonden, trouwpartijen en bezoekjes aan de eerste pagadder geworden. Geen idee hoe onze samenkomsten er binnen enkele jaren zullen uitzien maar ik kijk er alvast naar uit. Ondertussen zullen jullie wel iets anders moeten zoeken om me te jennen dan de eeuwige "Wanneer ga je nu een echte job zoeken?". Nu ja, mijn "Voor mijn dertigste werk ik niet!"-antwoord heeft ondertussen ook zijn houdbaarheidsdatum bereikt.

Waar mijn dankbaarheid voor mijn collega's en vrienden nog enigszins uit te drukken valt met mijn beperkte gave van het geschreven woord, is dit een hopeloze zaak voor de dank die ik mijn familie verschuldigd ben. In het bijzonder denk ik hier aan onze, ik denk dat ik ook in naam van mijn broer mag spreken, moeder. Elk eigenbelang negerend, ving zij gedurende vele jaren alle schokken op. Eindeloos goed, zoals alleen moeders kunnen zijn maar tegelijk nog zoveel meer. Zelf zal je je inspanningen wegzetten als

1. Eén van de vele wijsheden die Mary Schmich deelde met de wereld in haar column voor de Chicago Tribune in juni 1997. Ondertussen beter bekend als de muzikale hit "Everybody's Free (To Wear Sunscreen)" van Baz Luhrmann. Een aanrader voor al uw existentiële crisissen.

de normaalste zaak van de wereld. Dat zijn ze niet. En dat zullen we nooit vergeten. Nu ik écht student af ben, wordt het misschien wel eens tijd dat ik mijn eigen wasmachine en strijkijzer koop. Maar geen nood, op zondag mag je mij nog steeds verwachten aan de feestdis.

Moest je mijn broer niet zijn, zouden we misschien niet in eenzelfde vriendengroep thuis horen. Toch is onze band eindeloos sterk en apprecieer ik je meer dan ik ooit zal toegeven. Het is dan ook met veel trots en plichtsbewust dat ik peter zal zijn van je eerstgeborene. Nu ja, iemand moet jullie kind leren voetballen, van jou zal het niet geleerd worden! Jammer genoeg voor jou kan ik nu wel weer de titel claimen van slimste van de familie met een dr. en een ir. voor mijn naam. De volgende keer dat ik je in een houdgreep houd mag je je alvast verwachten aan een allesbehalve retorische "who is the smartest?" in plaats van "who is the strongest?".

Dan zijn we aangekomen bij de laatste maar, zoals het cliché het wil, zeker niet de minste persoon die ik moet bedanken. Q. De laatste drie jaar hebben we naast jammer genoeg leed ook, en vooral, veel liefde gedeeld. Tijdens onze drie jaar is niet alleen mijn liefde maar ook mijn bewondering voor jou alleen maar gegroeid. Je weet het misschien niet maar zelden heb ik zo naar iemand opgekeken. Je hebt een steiler pad beklommen dan zowat iedereen die ik ken en al zeker mezelf. Dat getuigt van enorm veel karaktersterkte en onversneden talent. Twee zaken die ervoor zullen zorgen dat het leven je nog veel zal teruggeven. Ik ben blij dat ik mijn laatste conferenties kon uitbreiden met reisjes in jouw gezelschap. Ik kijk dan ook naar niets meer uit dan samen met jou de wereld en ons leven verder te verkennen. Wat denk je, zijn we weg?

Kurt Van Hautegem
Gent, oktober 2018

Contents

Samenvatting

In dit proefschrift ontwikkelen en bestuderen we de prestatie van verscheidene planningsalgoritmen voor optische pakketgeschakelde (Optical Packet Switching, OPS) en optische burstgeschakelde (Optical Burst Switching, OBS) netwerken. Zowel OPS- als OBS-netwerken worden in de literatuur naar voor geschoven als alternatieven om de capaciteit van de glasvezelbackbone van het internet te vergroten. Aangezien door de stijgende populariteit van cloud computing en media streaming de vraag naar bandbreedte alleen maar toeneemt, moet de onderliggende fysieke structuur van het internet immers mee evolueren. Hoewel de vraag naar bandbreedte al decennia stijgt, hebben de doorbraak van glasvezeltechnologie en bijhorende multiplexingtechnieken het mogelijk gemaakt om de capaciteit op de verbindingen nog sterker te doen groeien. Met snelheden tot 1 Petabit/s (=1000 terabit/s) lijkt het wel alsof aan onze onstilbare honger naar data zonder problemen kan voldaan worden. Niets is minder waar. In de huidige glasvezelbackbonenetwerken wordt de capaciteit immers niet begrensd door de onderlinge verbindingen maar door de knooppunten waar deze samenkomen. Dit is een gevolg van de pakketgebaseerde manier waarop data wordt getransporteerd en wordt verwerkt in deze knooppunten. Om de bestemming van elk pakket te bepalen, worden in deze knooppunten de optische signalen tijdelijk omgezet naar en opgeslagen als elektronische signalen. Het eigenlijke schakelen gebeurt dus elektronisch en vereist omzettingen tussen het optische en elektronische domein (O/E/O conversies). De snelheid waarmee deze conversies mogelijk zijn, is echter aan een trager tempo geëvolueerd dan de capaciteit van de glasvezelverbindingen. Hierdoor is het schakelen in de knooppunten het knelpunt van netwerkcapaciteit geworden.

Om tegemoet te komen aan de groeiende bezorgdheid over de netwerkcapaciteit in de knooppunten van elektronisch geschakelde backbones, werd optisch circuitschakelen (optical circuit switching, OCS) naar voor geschoven als alternatief. In deze netwerken worden de tussenliggende omzettingen vermeden door de gegevens op te slaan in het oorsprongsknooppunt en deze

te versturen over een continu, ononderbroken en vooraf bepaald optisch pad (mogelijks over meerdere tussenliggende knooppunten) tot aan het bestemmingsknooppunt. Ondanks enkele voordelen, zoals constante pakketvertraging en verminderde overhead, heeft circuitschakelen het nadeel dat het de beschikbare bandbreedte inefficiënt gebruikt. Zolang een verbinding tussen twee knooppunten aanhoudt, kunnen de gebruikte golflengtes op de tussenliggende segmenten immers enkel gebruikt worden door deze verbinding. Door de pakketgebaseerde manier van het verzenden van gegevens resulteert dit in veel tussenliggende periodes waarop verbindingen er, ondanks capaciteitsdruk, ongebruikt bijliggen.

Om de O/E/O conversies te vermijden maar tegelijk de flexibiliteit en efficiëntie van pakketgeschakelde netwerken te behouden, wordt OBS gezien als de volgende stap in de evolutie van schakelen in de backbone. In OBS worden pakketten met hetzelfde bestemmingsknooppunt gegroepeerd in grotere bursts in de knooppunten aan de rand van het backbone-netwerk. Na deze groepering wordt de bestemming en eventueel andere overheadinformatie van deze burst apart en op een toegewijd kanaal verstuurd. Dit laat toe deze informatie elektronisch te verwerken in het volgende knooppunt alvorens de eigenlijke burst er aankomt. Door de overhead apart te versturen en te verwerken wordt de optische extractie en verwerking van de overhead, beide knelpunten op technologisch vlak, vermeden. Belangrijker is echter dat de burst gedurende het volledige traject in het optische domein blijft, waardoor de nood aan elektronische RAM opslag in tussenliggende knooppunten omzeild wordt. OBS zit momenteel nog in onderzoeksfase hoewel er, door onder meer Huawei, reeds indrukwekkende prototypes werden gebouwd in de afgelopen jaren.

Bij OPS worden pakketten niet gegroepeerd in bursts maar worden ze individueel verstuurd over het netwerk. De overheadinformatie wordt dan ook niet apart verstuurd maar samen met het eigenlijke datapakket. Dit zorgt ervoor dat er veel minder tijd is om de header te verwerken en dat deze bij voorkeur dus optisch wordt verwerkt. De eerste wetenschappelijke literatuur die OPS voorspelde als de toekomstige standaard voor pakketschakeling in de backbone, verscheen reeds meer dan 30 jaar geleden. Grote hinderpalen die de implementatie van OPS in de backbone verhinderen, zijn het optische bufferen en vooral de schaalbaarheid van de nodige technologie naar enerzijds een groot aantal parallelle golflengtes en anderzijds de hoge beoogde schakelingsfrequenties. Aangezien een volledig optische implementatie van de OPS-componenten resulteert in niet-lineaire vervorming van het optische signaal, is er bovendien ook technologie vereist die de optische signalen kan

herstellen en versterken. Ondanks deze hindernissen blijft de interesse in OPS als toekomstige schakelingstechnologie bestaan.

Een essentieel aspect van zowel OPS- als OBS-netwerken is het oplossen van transmissieconflicten. Door de eis van een toegewijd kanaal tussen oorsprong en bestemming te laten vallen, worden tussenliggende netwerkverbindingen immers gedeeld door verschillende communicatiepaden. Hoewel dit resulteert in een efficiënter gebruik van de beschikbare bandbreedte, zorgt dit ook voor het ontstaan van transmissieconflicten in de knooppunten wanneer verschillende pakketten (of bursts) op hetzelfde moment naar hetzelfde uitgangskanaal geschakeld worden. Hierin verschillen OBS en OPS niet van andere pakketgeschakelde technieken maar wel in de manier waarop deze conflicten (kunnen) opgelost worden. In elektronisch pakketgeschakelde netwerken worden deze conflicten simpelweg opgelost door één van pakketten op te slaan in het elektronische geheugen en het andere te versturen. Wanneer het uitgangskanaal weer beschikbaar is, wordt het pakket dat opzijgezet werd weer uit het elektronische geheugen gehaald en verstuurd. Aangezien een elektronisch geheugen de inspanningen om de O/E/O conversies te vermijden zou tenietdoen, moet er naar alternatieven gezocht worden.

De meest gebruikte methode om met transmissieconflicten in zowel OPS als OBS om te gaan is een gecombineerd gebruik van golflengteomzetting en optische buffering. Bij het parallelliseren van meerdere golflengtes worden pakketten tegelijk verstuurd over één enkele verbinding maar op verschillende golflengtes. Optische buffering lost transmissieconflicten op door één van de pakketten door een voldoende lange opgerolde glasvezel te sturen waardoor dit pakket vertraagd wordt. Een veelvoorkomende vorm maakt gebruik van zogenaamde Fiber Delay Lines (FDLs), waarin een pakket dat moet gebufferd worden door één van een set glasvezeldraden van verschillende lengte kan gestuurd worden. In een alternatief ontwerp met zogenaamde fiber loops ondergaan pakketten één of meerdere recirculaties in dezelfde glasvezellus. Beide ontwerpen bieden evenwel slechts een discreet aantal mogelijkheden aan toegewezen vertragingen. De optische buffers zijn, hoewel ze opgerold zijn, bovendien relatief groot qua fysieke dimensies. In tegenstelling tot bij elektronisch geheugen is de totale buffergrootte dan ook sterk beperkt en kan pakketverlies nooit vermeden worden.

De planningsalgoritmen controleren zowel de toewijzing van de vertraging als de golflengte van de aankomende pakketten door de beschikbaarheid aan golflengteomzetters en optische buffercapaciteit in rekening te brengen. Aangezien de capaciteit van zowel de golflengteomzetters als de optische buf-

fers, door fysische restricties, steeds sterk beperkt is, kunnen goed gekozen planningsalgoritmen een substantieel verschil maken in prestaties zoals pakketverlieskans. Ze vormen dan ook een essentieel onderdeel van het oplossen van transmissieconflicten.

In hoofdstuk 2 geven we eerst een inleiding tot de basis van planningsalgoritmen wanneer een FDL buffer en golflengteomzetters aanwezig zijn. Vervolgens focussen we op parametrische kostgebaseerde planningsalgoritmen die verschillende karakteristieken in rekening brengen om tot een betere prestatie te komen. In hoofdstuk 3 breiden we de kostgebaseerde planningsalgoritmen uit hoofdstuk 2 uit om het overmatig gebruik van de golflengteomzetters af te straffen. Hierdoor vermindert niet alleen het gebruik van de golflengteomzetters en het bijhorende energieverbruik maar verbetert ook de algemene prestatie. Dit komt doordat de golflengteomzetters een schaars goed zijn dat op deze manier effectiever gebruikt wordt. In hoofdstuk 4 gaan we dieper in op een type van planningsalgoritmen die het ontstaan en opvullen van periodes tussen pakketten, de zogenaamde leemtes, doelbewust optimaliseert. Aangezien dit een ingewikkeldere controle vereist van de FDL buffer, focussen we op dit aspect en gaan we uit van geen of onbeperkte capaciteit aan golflengteomzetters waardoor we onze analyse kunnen beperken tot één enkele golflengte. Hoofdstuk 5 presenteert nieuw ontwikkelde post-reservatie planningsalgoritmen voor een ontwerp met fiber loops. Om gelijkaardige redenen als in hoofdstuk 4 gaan we hier uit van één enkele golflengte.

Hoewel er een waaier aan analytische oplossingsmethoden bestaat, zijn de in dit proefschrift onderzochte systemen en algoritmes vaak zo ingewikkeld dat hun gedrag alleen kan geanalyseerd worden met behulp van simulaties. Hoewel een methodologie van discrete-event- en Monte-Carlo-simulatie gebruikt wordt doorheen dit proefschrift, zijn waar mogelijk de algoritmes wel gekaderd binnen een wiskundige afbakening die het mogelijk maakt om de gebruikte algoritmes op een doordachte manier verder te ontwikkelen. Dit kader vergroot het inzicht in de werking van de algoritmes en maakt het mogelijk om ze te gebruiken in andere omstandigheden zonder veel bijkomend simulatiewerk.

Of en wanneer OBS en OPS commercieel en operationeel levensvatbare oplossingen zullen zijn voor het schakelen in backbone-knooppunten is moeilijk te voorspellen. De technische en operationele motivatie achter de visie van het optische schakelen heeft in ieder geval enkel aan kracht gewonnen: een steeds stijgende vraag naar bandbreedte, de fundamentele beperkingen van

elektronisch schakelen en de technologische vooruitgang bij optisch schakelen. Het blijft ook bijzonder aantrekkelijk om data die als (IP-)pakketten binnenkomt ook eenvoudigweg als pakketten te verwerken, omdat dit de meest transparante, flexibele en adequate aanpak is. Hierdoor blijven zowel OBS als OPS grote kanshebbers voor toepassing in toekomstige netwerken. Wanneer dit het geval zal zijn, zullen optische buffering en de bijhorende planningsalgoritmen een essentieel onderdeel zijn van het ontwerp van schakelelementen voor de knooppunten. De auteur hoopt dan ook dat dit doctoraatsproefschrift zijn plaats heeft bij het toekomstige gebruik van optische pakketgeschakelde technologie in netwerken.

Summary

In this dissertation we develop and study the performance of a variety of scheduling algorithms for optical packet switched (OPS) and optical burst switched (OBS) networks. Both OPS and OBS networks are proposed as alternatives to increase the capacity of the Internet's optical backbone in the future. For instance cloud computing and (high definition) streaming media services are expected to vastly increase the demand for bandwidth. In response, the Internet's underlying physical network and its demand and supply issues will have to be addressed. Although demand for bandwidth has risen dramatically, the dawn of optical fiber and multiplexing techniques allowed the link capacity to outpace this surge. With dazzling bandwidths of up to 1 Petabit/s (=1000 terabit/s), it seems that our unlimited demand for bandwidth can be met without a problem. In existing optical networks, however, capacity is not limited by the connections (links) but by the intersections (nodes). This is a consequence of the packet-based way data is transferred and processed in these nodes. In order to extract the header data, and thus to determine the packet's destination, the optical signals are temporarily converted and stored into electronic signals in the nodes of backbone networks. The actual switching is thus done electronically and requires conversions between optical and electronic domain (O/E/O conversions). As optical transmission capacity increases these conversions are increasingly outpaced and switching has become the bottleneck in terms of core network speed.

To address the growing concerns of network capacity in the nodes of electronic switched backbones optical circuit switching (OCS) is proposed as an alternative. In these networks intermediate conversions to electronic signals are avoided by storing the data at the origin node until a continuous optical path to the destination node is established. Despite some benefits as constant delay and low overhead, circuit switching brings the disadvantage of an inefficient use of the available bandwidth. During the time a connection is established between an origin and destination node, the reserved wavelengths

in the different sections of the origin-destination path are only available for this connection. Because of the packet-based way data is transferred, despite demand for capacity, this results in many intermediate periods in which no data is transmitted on the reserved wavelengths.

To avoid the O/E/O conversions while maintaining the flexibility and statistical multiplexing capabilities of packet switched networks, optical burst switching (OBS) is proposed as the next step in the switching evolution. In OBS, packets with the same destination are grouped into larger bursts in the OBS edge nodes. After this burstification, the control information of this burst is transmitted in advance on a separate and dedicated control channel, allowing for the electronic processing of the header information by the time the burst arrives in the node. By keeping the control packet on a separate channel, optical header extraction and processing, which are both optical bottleneck technologies, are avoided. More importantly, however, the burst itself stays in the optical domain, eliminating the need for electronic RAM in the intermediate nodes. OBS is currently still in its research phase, although impressive prototypes, by Huawei for example, have emerged in recent years.

In OPS, packets are not grouped in bursts but sent individually over the network. The optical header data is thus not sent in advance or on a dedicated control channel but together with the payload. As a consequence there is a lot less time to process the header and the header processing is preferably done optically as well. The first scientific papers that envisioned optical packet switching as the future backbone technology appeared more than 30 years ago. Major hurdles for OPS backbone implementation are scalability to a huge number of wavelength and spatial channels, optical buffering and most importantly increasing the switching speeds to enable packet-based traffic at the required small time scale. Moreover as all-optical implementations of these OPS elements cause major non-linear impairments, reliable and high-speed optical signal regeneration solutions are required. Despite these considerable hurdles, interest in OPS as a future switching technology remains alive.

A vital aspect of both OPS and OBS networks is contention resolution. By dropping the requirement of establishing a dedicated communication channel between origin and destination node, the network links are shared among different communication sessions. Although this results in a more efficient use of the available bandwidth, it also raises the concern that in these packet-based switching techniques contention may arise in the network nodes when more than one packet (or burst) heads for the same output port at the same

time. In this, OBS and OPS do not differ from other packet-switched techniques in the extent to which contention occurs, but rather in the way it is resolved. In electronic packet switching contention is simply resolved by storing one of the contending packets in the RAM while the other packet is sent immediately. When the output line is available again, the packet that was put aside is retrieved from the RAM and sent. As the use of electronic storage in OBS or OPS would nullify the efforts of avoiding O/E/O conversions for the data payload, alternatives have to be employed.

The most used method to deal with contention (for both OPS and OBS) is a combined use of wavelength conversion and some type of optical buffering. In wavelength multiplexing, packets can be sent coincidentally on a single fiber but on a different wavelengths. Optical buffering resolves contention by sending a packet through a sufficiently long piece of coiled fiber to delay it for the time needed. A frequently occurring design is one with Fiber Delay Lines (FDLs), in which a packet that needs to be buffered is sent through one of a set of fibers with different lengths. In the alternative recirculation design of so called fiber loops on the other hand, a packet is recirculated a variable number of times in the same smaller fiber loop. Both fiber delay line and fiber loop designs can only provide a discrete set of delays. Moreover, although the fibers are coiled, their physical size remains significant. As opposed to electronic memory (RAM), the total buffer size will thus always be severely limited and packet loss not entirely avoidable.

Scheduling algorithms control both the delay and wavelength assignment of the arriving packets by taking into account both the availability of wavelength converters and optical buffer capacity. As, because of physical limitations, the capacity of both wavelength converters and optical buffers is always severely limited, selecting adequate contention resolution algorithms makes a substantial difference in performance and lies at the heart of effective contention resolution in OPS/OBS. Throughout this dissertation the performance of several different scheduling algorithms in a variety of settings is evaluated.

In Chapter 2 we first give an introduction to the basics of scheduling when an FDL buffer and wavelength converters are present. Next we focus on parametric cost-based scheduling algorithms that take into account both gap and delay characteristics to achieve superior performance. In Chapter 3 the cost-based scheduling algorithms of Chapter 2 are extended to penalize the use of the wavelength converters. This does not only result in a reduced use of the wavelength converters, and thus reduced energy consumption, but, in the

case of a limited number of wavelength converters, also in a more effective use of a scarce resource and thus an improved performance. In Chapter 4 we focus on a type of algorithms that optimize the creation and filling of idle periods in between packets; i.e., the so-called voids. As this requires a more complex control for the FDL buffer, we focus on this aspect and confine our analysis mostly to a single wavelength, assuming either no or unlimited wavelength conversion capability is present. As such, wavelength converters are not taken into account explicitly in the algorithms. In Chapter 5 new post-reservation algorithms are developed for a setting with fiber loops. For similar reasons as in Chapter 4, the focus is put on the control of the optical buffer and wavelength converters are not taken into account.

Despite the wide range in analytical solution methods, the systems and algorithms investigated in this work are so complex that their behavior can only be analyzed by means of simulation. Although a methodology based on discrete event and Monte Carlo simulation is used throughout this dissertation, wherever possible we try to back our algorithms by a mathematical framework that may help in extending our algorithms in a sensible way. This framework deepens the insight in the algorithms and makes it possible to apply them in other settings without (a lot of) additional simulations.

Whether and when optical packet or burst switching will be a commercial and operational viable solution to address switching in the core Internet nodes is hard to tell. The technical and operational drivers for the vision of the optical layer are nevertheless genuine: the ever increasing demand for Internet traffic, the fundamental limitations of electronic switching and the technological advances in optical switching. Moreover as IP traffic is packet-based, using packet-based switching techniques remains the most natural, flexible and transparent choice for optical networks. Because of this, optical packet and burst switching remain viable alternatives for future network architectures. If and when optical packet or burst switching becomes the go-to technology for core node switching, optical buffering and the complementary scheduling algorithms will be at the heart of switch design. The author therefore hopes that this dissertation will have its place in the deployment of optical packet and burst switching technology in future networks.

List of abbreviations, variables and terms

4K

Also known as UHD (Ultra-High-Definition). Refers to a horizontal screen display resolution in the order of 4000 pixels. Requires roughly 4 times as much data to collect, move and store as HD (High-Definition).

5G

5^{th} generation wireless systems: improved mobile networks deploying in 2018 and later.

AO-WC

All-Optical Wavelength Converter: Type of wavelength converter (WC) that changes the wavelength of an optical signal without the need of an intermediate electronic step.

ASON

Automatically Switched Optical Networks: OCS networks in which origin-destination paths do not have to be set up manually by an operator but are managed by the control plane on a sub-second time scale.

B

Parameter denoting the packet or burst size.

BHP

Burst Header Packet: the control information of a burst in an OBS network. Transmitted in advance and on a separate channel.

Burst

Group of packets with the same destination node that are processed together in OBS networks.

C

Cost function associated with each SP that equals a weighted average of the delay and gap.

C

Cost: non-void-filling scheduling algorithm that schedules on the SP with the lowest cost C.

c

Parameter denoting the number of wavelengths packets can arrive and depart on.

C-VF

Cost-Void-Filling: void-filling scheduling algorithm that schedules on the SP with the lowest cost C.

CDN

Content Delivery Network: a network of proxy servers and their data centers geographically distributed in order to provide data with low latency.

Channel

Depending on the context either a synonym for wavelength or referring to a spatial channel in SDM.

C_W

Cost function associated with each SP that equals a weighted average of the delay and gap, and an extra term that incorporates (and reduces) the usage of the wavelength converters.

CW

Cost and Wavelength: non-void-filling scheduling algorithm for a setting with unlimited wavelength conversion capacity that schedules on the SP with the lowest cost C_W.

CW-VF

Cost and Wavelength-Void-Filling: void-filling scheduling algorithm for a setting with unlimited wavelength conversion capacity that schedules on the SP with the lowest cost C_W.

$C_{W,A}$

Cost function associated with each SP that equals a weighted average of the delay and gap, and an extra term that incorporates (and reduces) the usage of the wavelength converters. Equal to the C_W cost.

CWA

Cost and Wavelength: non-void-filling scheduling algorithm for a setting of a limited number of wavelength converters that schedules on the SP with the lowest cost $C_{W,A}$.

CWA-VF

Cost and Wavelength-Void-Filling: void-filling scheduling for a setting of a limited number of wavelength converters algorithm that schedules on the SP with the lowest cost $C_{W,A}$.

$C_{W,B}$

Cost function associated with each SP that equals a weighted average of the delay and gap, and an extra term that incorporates (and reduces) the usage of the wavelength converters. The conversion cost is only added if the last vacant WC has to be used.

CWB

Cost and Wavelength: non-void-filling scheduling algorithm for a setting of a limited number of wavelength converters that schedules on the SP with the lowest cost $C_{W,B}$.

CWB-VF

Cost and Wavelength-Void-Filling: void-filling scheduling for a setting of a limited number of wavelength converters algorithm that schedules on the SP with the lowest cost $C_{W,B}$.

$C_{W,C}$

Cost function associated with each SP that equals a weighted average of the delay and gap, and an extra term that incorporates (and reduces) the usage of the wavelength converters. Assigns a linearly rising extra cost as less WCs are vacant.

CWC

Cost and Wavelength: non-void-filling scheduling algorithm for a setting of a limited number of wavelength converters that schedules on the SP with the lowest cost $C_{W,C}$.

CWC-VF

Cost and Wavelength-Void-Filling: void-filling scheduling algorithm for a setting of a limited number of wavelength converters algorithm that schedules on the SP with the lowest cost $C_{W,C}$.

$C_{W,D}$

Cost function associated with each SP that equals a weighted average of the delay and gap, and an extra term that incorporates (and reduces) the usage of the wavelength converters. Assigns an exponentially rising extra cost as less WCs are vacant.

CWD

Cost and Wavelength: non-void-filling scheduling algorithm for a setting of a limited number of wavelength converters that schedules on the SP with the lowest cost $C_{W,D}$.

CWD-VF

Cost and Wavelength-Void-Filling: void-filling scheduling for a setting of a limited number of wavelength converters algorithm that schedules on the SP with the lowest cost $C_{W,D}$.

D

The granularity: parameter value of which the delay values in a degenerate FDL buffer are consecutive multiples.

D-G

Delay-Gap: non-void-filling scheduling algorithm that schedules on the SP with the lowest delay. Ties are chosen in favor of the SP with the lowest gap.

D-G-VF

Delay-Gap-Void-Filling: void-filling scheduling algorithm that schedules on the SP with the lowest delay. Ties are chosen in favor of the SP with the lowest gap.

DCN

Data Center Network: interconnects all of the hundreds and thousands of servers in a data center. Today's data centers are constrained by the interconnection network.

Degenerate

FDL buffer settings in which the FDLs have lengths that are multiples of the granularity D.

DES

Discrete Event Simulation: simulation model that models the operation of a system as a discrete sequence of events in time.

DSL

Digital Subscriber Line: family of technologies that are used to provide Internet access over telephone lines.

DWDM

Dense Wavelength Division Multiplexing: a more recent high-capacity version of WDM.

FCFS

First-Come, First-Served: scheduling policy according to which the packet that has been waiting the longest is served first.

FDL

Fiber Delay Line: optical buffer in which packets are sent through sufficiently long pieces of coiled fiber to delay it for the time needed.

FTTH

Fiber To The Home: broadband network architecture using optical fiber as far as the outside wall of homes.

FTTN

Fiber To The Node: broadband network architecture using optical fiber up until a street cabinet with the final connections being copper.

G-D

Gap-Delay: non-void-filling scheduling algorithm that schedules on the SP with the lowest gap. Ties are chosen in favor of the SP with the lowest delay.

G-D-VF

Gap-Delay-Void-Filling: void-filling scheduling algorithm that schedules on the SP with the lowest gap. Ties are chosen in favor of the SP with the lowest delay.

HOL-blocking

Head Of Line blocking: type of congestion only caused in input buffered switches.

IP

Internet Protocol: together with TCP the principal communication protocols of the Internet.

IT

Information Technology: the application of computers to store, retrieve, transmit and manipulate data or information, often in the context of a business or other enterprise.

JSQ

Join-the-Shortest-Queue: non-void-filling scheduling algorithm that schedules on the channel with the lowest horizon.

K

Parameter denoting the number of input fibers of the overall switch configuration.

LP

Loss Probability: performance measure denoting the percentage of lost packets.

LPsize

Loss Probability Size: performance measure denoting the sum of all packet sizes of lost packets divided by the sum of all packet sizes of arrived packets (i.e. lost and accepted).

LTE

Long Term Evolution: standard for high-speed wireless Internet access based on GSM/EDGE and UMTS/HSPA technologies mainly used for mobile devices.

M

Parameter denoting the number of output fibers of the overall switch configuration.

MEMS

Micro-Electromechanical Systems: the technology of microscopic devices, particularly those with moving parts. Also referred to as micromachines (Japan) and micro systems technology (Europe).

N

Parameter denoting the optical buffer size. The number of lines in an FDL buffer, including the line with length zero, is $N + 1$.

Non-Void-Filling (NVF)

Type of scheduling algorithm according to which packets can only join at the back and voids between already scheduled packets can not be filled.

O/E/O

Optical Electronic Optical conversion: time-consuming signal conversions, needed in electronic packet switched networks.

OBS

Optical Burst Switching: optical switched network in which packets with the same destination node are grouped in larger bursts before being sent over a packet-based network, whereby header and payload are transmitted separately.

OCS

Optical Circuit Switching: optical networks in which a continuous optical path is established between origin and destination node.

OEO-WC

Optical-Electronic-Optical Wavelength Converter: Type of wavelength converter (WC) that changes the wavelength of an optical signal by using an intermediate electronic step.

OPS

Optical Packet Switching: optical switched network in which packets are sent individually over the network.

PSP

Potential Scheduling Point: each intersection between a horizontal line and a vertical line in the provisional schedule.

QoS

Quality of Service: description of the overall ability to prioritize packets and apply resource reservation control mechanisms.

r

Parameter denoting the number of wavelengths converters available for converting packets.

RAM

Random-Access Memory: a form of electronic computer data storage used widely in today's computer networks.

S

Parameter denoting the length of the fiber loops in a optical fiber loop buffer.

SC

Scheduling Criterium: property regarding the provisional schedule, the incoming wavelength or the number of available WCs that scheduling algorithms take into account to choose a SP.

SDM

Space Division Multiplexing: a technology which spatially multiplexes a number of optical carrier signals onto a single optical fiber by either using multicore fiber (MCF) or multimode fiber (MMF).

SOA

Semiconductor Optical Amplifiers: device that amplifies an optical signal directly, without the need to first convert it to an electrical signal.

SP

Scheduling Point: those PSPs allowing the packet to be scheduled with the used algorithm and the vacant WCs.

T

Parameter denoting the inter-arrival time between packets.

T-WAS

Extension of WAS in which an additional threshold variable is introduced and varied to achieve further improved performance.

T-XAS

Extension of XAS in which an additional threshold variable is introduced and varied to achieve further improved performance.

TCP

Transmission Control Protocol: together with IP the principal communication protocols of the Internet.

TWC

Tunable Wavelength Converter: Type of all-optical wavelength converter (AO-WC) in which the output wavelength can be changed for different packets.

VAS

Void Avoiding Schedule: post-reservation scheme for fiber loop buffers in which after each recirculation packets are transmitted immediately if the outgoing line is available.

VOD

Video On Demand (services): allows users to watch video when they choose to, rather than at a specific broadcast time.

Void

Unscheduled periods followed by one or more packet already scheduled beyond it.

Void-Filling (VF)

Type of scheduling algorithm in which packets have the possibility of filling up voids.

VoIP

Voice over Internet Protocol: group of technologies for the delivery of voice communications and multimedia sessions over Internet Protocol (IP) networks, such as the Internet.

WAS

Variation of VAS selectively delaying packets longer than strictly necessary for attaining better switching performance.

WC

Wavelength Converter: device used to change the wavelength of a packet or burst.

WDM

Wavelength Division Multiplexing: a technology which multiplexes a number of optical carrier signals onto a single optical fiber by using different wavelengths (i.e., colors) of laser light. This technique enables bidirectional communications over one strand of fiber, as well as multiplication of capacity.

WiFi

Technology for wireless local access to the Internet. The Wi-Fi Alliance used the advertising slogan "The Standard for Wireless Fidelity" for a short time after the brand name was created. The name was, however, never officially "Wireless Fidelity".

WiMAX

Worldwide Interoperability for Microwave Access: family of wireless communication standards used to provide wireless broadband access in the last mile as an alternative for DSL and cable.

XAS

Variation of VAS selectively delaying packets longer than strictly necessary for attaining better switching performance.

List of Figures

List of Tables

Chapter 1

Introduction

I think there is a world market for maybe five computers.
Thomas Watson, chairman of IBM, 1943

In this chapter we first give a helicopter view of the current state of the Internet, its underlying physical network and its demand and supply issues. Next, taking a gradual descend we show how various all-optical networks, and especially the packet-switched ones, could provide vital solutions for near-future network bottlenecks. As contention resolution is one of the major issues in these all-optical packet-switched networks, we further explore the components that enable the implementation of the accompanying scheduling algorithms. Finally, after demarcating our playground completely we elaborate on our methodology and show how this thesis is constructed and how its different parts interconnect.

1.1 Internet demand

The Internet has come a long way from the small research network it once was at ARPA (Advanced Research Projects Agency). In 1969, the first message that was meant to be sent on ARPANET was "login", but only the L and O were transmitted before the system crashed [5]. Today, almost 50 years later our demand for data seems insatiable and the global economy is woven into and dependent of the Internet in a both astonishing and sometimes alarming way. 2016 was the first year with an annual global IP traffic of more than one zettabyte [3], that is 10^{21} bytes or, on a logarithmic scale, almost equal to the number of stars in the Universe (10^{22} – 10^{24} [6]). Moreover, this global demand is expected to grow at a compounding annual rate of 24 % between

2016 and 2021. The main drivers behind this evolution are a higher number of people connected to the Internet (from (only) 44 in 2016 to 58 % of the global population in 2021), a growing shift towards smartphones (accompanied by a higher available bandwidth) and an explosive growth of IP video traffic (up to 80 % of the total IP traffic by 2021). The latter is due to the adoption to 4K image quality and an increased popularity of video on demand services (VOD). The implications of this are difficult to overstate as video has a true multiplier effect on both the higher number of people with internet access and the shift in devices.

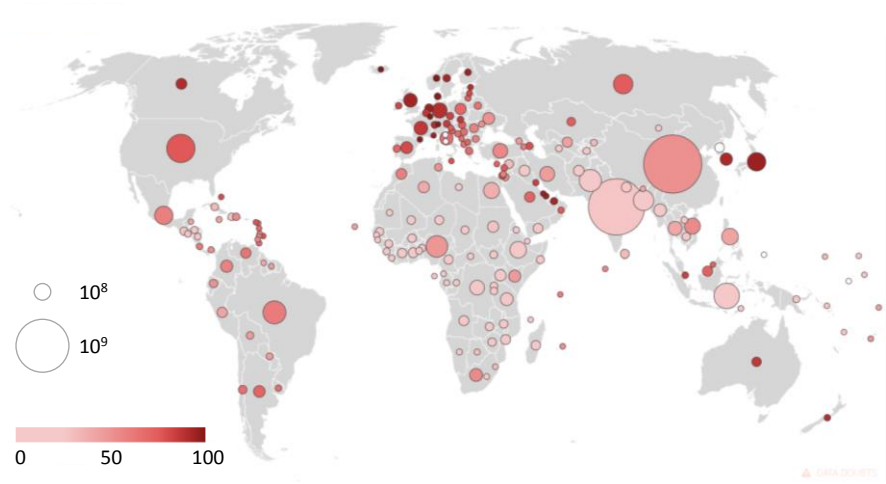


Figure 1.1: Bubble size: country population, bubble color: Internet users per 100 people. [1]

1.2 The physical structure of the Internet

Given the intense and seemingly unstoppable growth of demand for data, the network that has to accommodate all of this IP traffic needs its capacity to grow at an equal or higher pace. Although somewhat an abstraction due to their overlap, the physical network of the Internet can be divided roughly into the backbone (or core) network, the metro network and the access network. Like the trunk, branches and twigs of a tree they have a different topology and serve a different purpose.

The access network connects a limited number of individual users to the network. As the aggregated traffic amount of these users is limited (and in most cases restricted in small print in their subscription contract) the

physical layer of the wired access network is in most countries and regions often still implemented by means of older technologies as twisted pair (DSL) and coax cable. In some countries however, e.g., Japan, high speed optical fiber networks have been fanned out to individual households for more than a decade (FTTH: fiber to the home). With the growing shift towards smartphone use, mobile (i.e., not WiFi) access technologies have become increasingly important in bridging the so called last-mile to end users (18 % of global IP traffic by 2021). As these wireless technologies arose more recently and do not require expensive foot pavement works to be updated they are often implemented with more recent technology such as WiMAX and LTE in most of the densely populated regions in the world.

The metro network has a higher hierarchy and connects different access networks and some large individual users with each other. The metro network often still has a physical ring topology (although today often logically in a star topology [7]) and is frequently implemented by means of an optical fiber network nowadays (FTTN: fiber to the neighborhood). The percentage of IP traffic delivered within the same metro network is expected to increase from 22 % in 2016 to 35 % in 2021 as content delivery networks (CDNs) become more popular. These virtual networks, that will carry as much as 71 % of total Internet traffic by 2021, store a cached version of content in multiple geographical locations (a.k.a., points of presence, or PoPs) to reduce latency for the end user. Moreover, as the increased demand for video traffic tends to have a "prime time" on the local geographical level, peak demand (which determines capacity requirements) in metro networks will increase even stronger than the overall average.

The backbone network connects different metro networks on an international and intercontinental level. Because of this each link in this network has to transmit the aggregated data of thousands and even millions of users, which with current individual user demand levels amounts to the mentioned one zetabyte [3]. This has to happen in a robust way, i.e., with a low bit error rate and without the possibilty of a failure of a single link or node of the network corrupting the overall data transfer. This thus requires alternative paths and a high degree of redundancy in the network. Besides this, latency has to be low enough to allow (near) real-time applications, such as online games or teleconferencing. Because of these requirements (mainly the last one) satellite communication is not suited to provide backbone capacity. Nevertheless there are moonshot projects by some tech moguls like Elon Musk to install a low altitude (and thus low latency) Internet satellite network. As the low altitude limits the range of each satellite however, these networks

require thousands of satellites and are currently economically not viable [8]. Because of these drawbacks of satellite, the connections between backbone nodes nowadays remain centered around optical cables [9, 10]. These cables contain one or more optical fibers that are coated in plastic and contained in a protective layer depending on the environment the cable will be placed in. By combining Dense Wavelength Division Multiplexing (DWDM, a more recent version of WDM) and multicore fiber technology (MCF, space-division multiplexing (SDM)) a single optical fiber can easily accommodate dozens of both wavelength and spatial channels. As a result data transfer capacity has risen spectacularly over the past decades with in 2017 a 1 Pbit/s (10^{15} bit/s) single fiber lab record [11]. This was achieved over a distance of 200 km. Such distance is considerable, particularly in view of optical signal strength degeneration, which can only be partially mitigated by intermediate optical amplifiers, and is typically larger as the number of multiplexed channels increases. Over longer distances and in real-life conditions the achievable speeds are more modest, yet impressive nevertheless. The 6600 km long MAREA transatlantic optical cable owned by Microsoft and Facebook which went into use in Q1 of 2018, achieves a 160 Tbit/s (10^{12} bit/s) bandwidth through a cross section about the size of a garden hose. This is equivalent to simultaneously streaming more than 10 million 4K movies. As a reference: the first transatlantic optical fiber (TAT-8) allowed for a bandwidth of 280 Mbit/s (10^{6} bit/s) when it went into operation in 1988 [12]. This amounts to a capacity increase of a factor 571000 over 30 years, or a factor 1.94 per 18 months, which, in view of Moore's law [13], sounds familiar. And then again not.

1.3 Technological evolution mismatch

Although demand for bandwidth has risen dramatically, the dawn of optical fiber and multiplexing techniques allowed the link capacity to outpace this surge. Moreover, while once only used for the most important backbone links, today the optical fiber technology unrolls deeper and deeper into the network up until right at your doorstep with FTTH. As such it seems that our unlimited demand for bandwidth can be met without a problem. However, by eradicating the bottleneck in the links, a new one was created in the nodes of the network. This is a consequence of the way data is transferred and processed in these nodes. As TCP/IP is a packet-switched protocol it comes only naturally that the optical signals are also sent and processed as individual packets. In order to extract the header data, and thus to determine the packet's destination, these optical signals are temporarily converted into electronic signals in some of the nodes of backbone networks. The actual

switching is thus to be done electronically and requires two conversions: from optical to electronic (O/E) and then back from electronic to optical (E/O).

With increasing optical transmission capacity these resulting O/E/O conversions demand merely three concurrent technological evolutions to evolve at an equal pace [2]. A first one involves information processing, i.e., the processing of the data header. As more packets arrive per second, more headers have to be processed. As these headers contain more and complex guidance regarding the QoS, the burden only increases. However, as some other switching paradigms (see below) also suffer from this issue, it is less of a differentiator between them and should not be mentioned as a shortcoming of merely electronic switching. Moreover, although Moore's law (transistor density doubling every two years) seems too ambitious nowadays [13], processors proved to be able to handle increasing traffic and QoS. This is as IP traffic growth decreased in recent years from the heights of the nineties as can be seen from Fig. 1.2. A second technological evolution is the size and cost of RAM. As RAM footprint (and cost) tends to halve every 1.5 years, RAM capacity and physical footprint issues of core routers in electronic switched backbone networks are, similar to processor speed issues, more an issue of the past than they are today. A third and true bottleneck issue however is the frequency at which the RAM can be accessed. As this frequency has been improving at less than 10 % a year [14, 15], it could not keep up with the ever increasing rate at which packets arrive. Because of this, significant latencies in backbone networks have been introduced. These are so persistent that they have been the driving factor of the increasing popularity of CDNs. As CDNs require huge data centers around the world, they are a significant cost (both monetary and energy wise) of the Internet nowadays. Moreover for many real-time applications such as VoIP calls, data cannot be stored locally in CDNs, causing an inescapable lag for, for example, long distance (and thus many hops) Skype calls [16].

1.4 Optical circuit switching as an interim solution

To address the growing concerns of network capacity in the nodes of electronic switched backbones optical circuit switching (OCS) was proposed as an alternative around the turn of the century [2]. In these networks intermediate conversions to electronic signals are avoided by storing the data at the origin node until a continuous optical path to the destination node (passing through multiple intermediate nodes) is established [17, 18]. Once set up, data can flow in optical form, i.e., without intermediate O/E/O conversions,

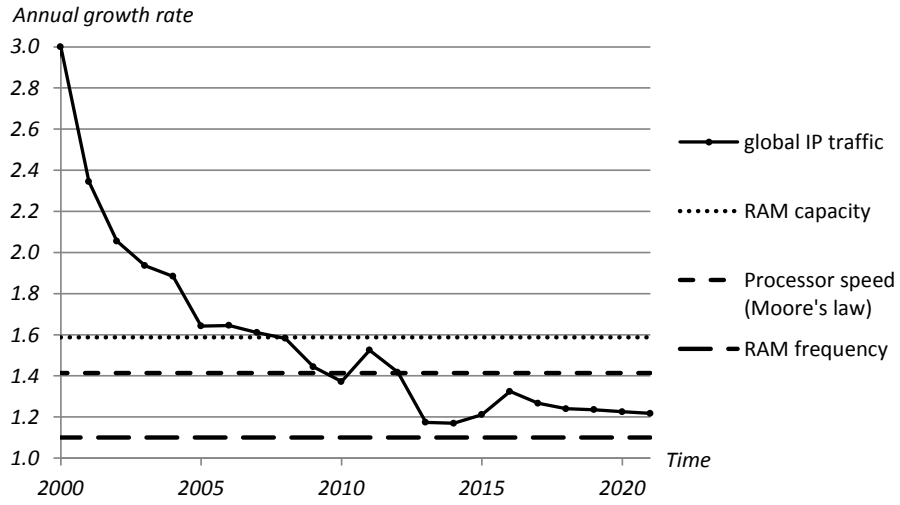


Figure 1.2: Past and future annual growth rate of global IP traffic and the required technology in electronic switched backbone nodes. [2, 3]

from origin to destination. Such a connection is called transparent, as it allows data to be sent through the network as a beam of light. This is as opposed to the opaque network connections in electronic switched networks in which O/E/O conversions prevent the formation of an uninterrupted optical path. As TCP/IP is a packet-switched protocol, i.e., no higher-level route is set up in advance and routing choices are made online on a packet-to-packet basis, using a similar packet-switched principle for the physical layer is a natural choice. Nevertheless, TCP/IP combined with circuit switching at the physical level is not infeasible, and it even brings some benefits as constant delay and a lower overhead [19]. Moreover as video demand IP traffic is heavy tailed, i.e., a small number of flows carry a large fraction of the total data, it is well suited to address this type of traffic.

Despite these benefits, circuit switching also brings a few disadvantages of which the major one is the inefficient use of the available bandwidth [20]. During the time a connection is established between an origin and destination node, the reserved wavelengths in the different sections of the origin-destination path are only available for this connection. This means that these wavelengths on these sections cannot be used by other data paths even if there are intermediate periods in which no data is transmitted between the original origin-destination pair. As the upper network layers (TCP/IP) are packet-based, these idle periods will inevitably take up a large share of the

time. A second disadvantage is the fact that the origin-destination paths had to be set up manually in the early days and flexibility thus was low. As these disadvantages outweighed the benefits, the development of optical circuit switching bore very little resemblance to early visions of how the technology would be deployed [21]. The components making OCS possible were mainly used in other technologies. MEMS (Micro-Electromechanical Systems) for example were and are still widely used in wavelength multiplexing (WDM) and to reroute traffic mechanically upon link failure.

Interest in OCS however was renewed with the development of Automatically Switched Optical Networks (ASONs) [22–24], in which origin-destination paths do not have to be set up manually by an operator but are managed by the control plane on a sub-second time scale. By doing so the session lengths during which origin-destination paths are kept online, are well managed and automatically adapted to current traffic conditions. Because of this OCS was being gradually recognized as the premier option for optical network development, resulting in ASONs being used in parallel with electronic packet-switched networks, often on the same equipment. Today, ASONs are even used as the core technology of prestigious state of the art projects, like the backbone of the 5G mobile network in South Korea, used at the 2018 Winter Olympics in Pyeongchang [25]. An approach where OCS is used in parallel with electronic packet switching is also widely used in data centers, where OCS handles the long-duration transfers while electronic switching is used for the short-duration transfers [26–28]. The main drivers in this choice towards OCS are potential advantages in cost and power consumption. As cloud applications requiring multiple servers simultaneously create a shift towards higher network throughput in datacenters, this trend is expected to last.

1.5 Optical burst switching as a natural successor

Despite the fact that bandwidth efficiency of OCS is increased by the use of ASONs, it remains subpar and is easily outperformed in this aspect by electronic packet-switched networks. Electronic switching is therefore still preferred for small data sizes for which the time to establish a channel from origin to destination node remains too large of an overhead [26]. Moreover, there will always be a certain unignorable degree of friction between a circuit-switched physical layer and the packet oriented IP traffic. In this view, OCS may be seen not as a shift from packet-switched to circuit-switched networks but rather as an intermediate step in the shift from electronic to optical

technologies in the backbone. This started with the medium itself, optical fiber, which was truly disruptive (when it was not yet fashionable) and set the bar a few orders of magnitude higher. Where at first all enabling technologies such as wavelength conversion and signal regeneration still happened in the electronic domain, these functions are gradually taken over by purely optical implementations. As eventually complete optical cross connects reached maturity, OCS became feasible in the backbone.

To avoid the O/E/O conversions while maintaining the flexibility and statistical multiplexing capabilities of packet-switched networks optical burst switching (OBS) is proposed as the next step in this evolution [29–32]. In an OBS network (Fig. 1.3), packets with the same destination are grouped into larger Bursts in the OBS edge nodes. After this burstification, the control information of this burst, the so called burst header packet (BHP), is transmitted in advance on a separate and dedicated control channel. With a carefully chosen offset time between the BHP and the actual payload, this allows for the electronic processing of the header information by the time the burst arrives in the node. As the control overhead in OBS is grouped for multiple packets, the control overhead per bit is reduced [33]. Moreover, as fewer headers have to be processed, their impact on the overall latency is reduced. By keeping the control packet on a separate channel, optical header extraction and processing, which are both optical bottleneck technologies, are avoided. More importantly however the burst itself stays in the optical domain, eliminating the need for electronic RAM in the intermediate nodes.

A key aspect of OBS is the burstification process [34, 35] as shown in Fig. 1.4. Different burstification assembly algorithms are possible but in general they can be classified as timer-based, burst-length-based or a combination of both. In the timer-based algorithm packets with the same destination edge node that arrive within a certain time frame are assembled in a burst. The length of the time frame should be set carefully as with a too large value, the delay at the destination edge node might be unsatisfactory. If the timer value is too short, too many bursts will be generated, nullifying the intended overhead reduction. Burst-length-based algorithms avoid too many small bursts, but have less control on the resulting delays. Appropriate combinations of both time and length algorithms can alleviate their downsides. Besides the same destination edge node, burst assembly algorithms can also take into account different packet classes and priorities, assembling them into different bursts with possibly different type of assembly algorithms and/or algorithm parameters. Moreover adaptive algorithms can take into account real-time traffic properties when determining algorithm parameters.

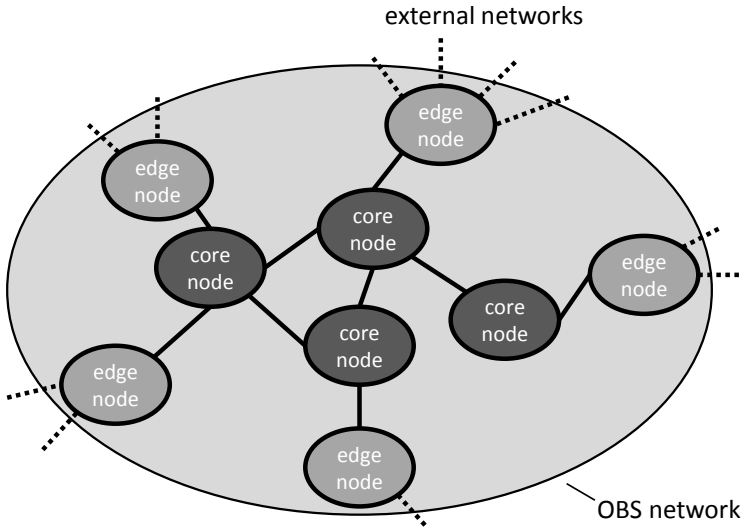


Figure 1.3: Illustration of an OBS network connected with external networks in the edge nodes.

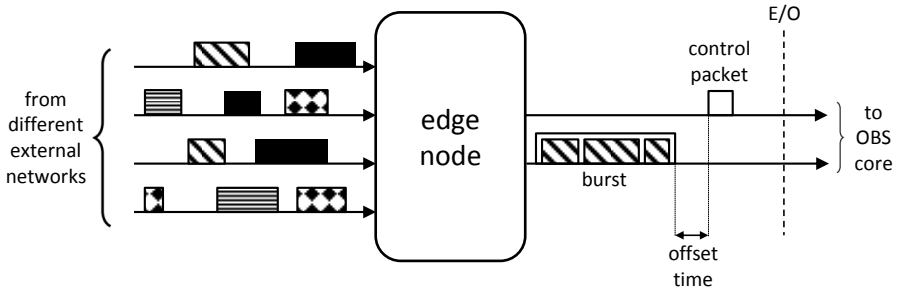


Figure 1.4: Burstification in an edge node. Packets with the same shading have the same destination edge node and possibly belong to the same packet class.

OBS is currently still in its research phase, although impressive prototypes have emerged in recent years. Huawei presented a 10 Pbit/s OBS prototype switch in 2012 [36] that dwarfs the capacity of current state of the art electronic switches. Moreover, because of its all-optical design, this prototype is both smaller in size and consumes less power. As throughput and energy consumption are as much of a concern in data centers and data center networks (DCN), OBS is increasingly identified as a major contender for future switching technology in this setting [37–40].

1.6 Optical packet switching as the Holy Grail

In OPS packets are not grouped in bursts but are sent individually over the network [41]. As a consequence the optical header data is thus not sent in advance or on a dedicated control channel but together with the payload as a label. The labelling is either done in a serial way, i.e., the label is directly attached in front of the payload, or in a parallel fashion by means of an optical subcarrier [42].

Regardless of the used method to attach the label, the functions and building blocks of the label processing are the same. These functions are shown in Fig. 1.5 [43]. In a first step the label is extracted from the payload. Depending on whether the labelling is done in a serial or parallel way, different methods can be applied. In the label interpretation block, the label is processed. Based on this information a new label is created and the routing information (destination, wavelength, length, priority ...) is forwarded to the switch control logic. Next the new label is attached to the payload which, to allow for label interpretation, was delayed using an optical delay line. The newly created packet is then forwarded to the optical switch, where the control logic already set up everything to route the packet to the correct destination.

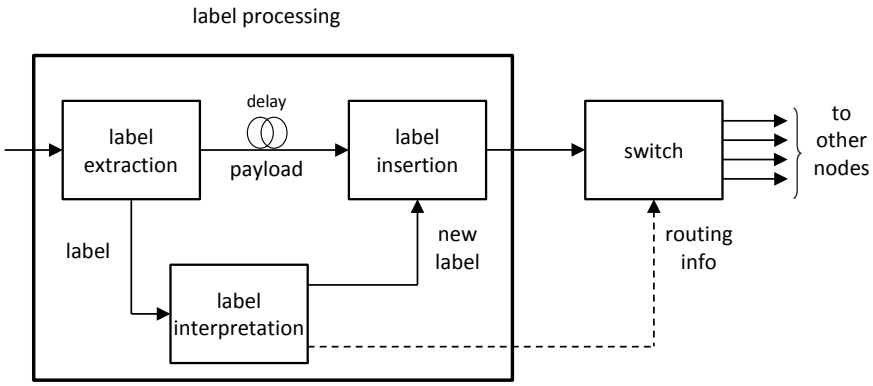


Figure 1.5: Label processing building blocks in Optical Packet Switching.

Besides a serial or parallel label, a few other key aspects distinguish different implementations of OPS. A first one is the degree to which the controls are moved to the optical domain. So called all-optical packet switching interprets the label completely in the optical domain. This solution, called all optical label swapping [44, 45], bypasses the electronic domain altogether and therefore, at least theoretically, allows for the fastest switching. As an

alternative the label can be interpreted in the electronic domain, relaxing the stringent technological requirements for OPS at least partially.

A second key aspect is synchronous or asynchronous operation [46–48]. In synchronous OPS networks, the state of the switch fabric can only be reconfigured at discrete times, i.e., time is slotted. As packets arriving on different input fibers encounter different propagation distances, they will not arrive in a synchronized manner. As a consequence all packets have to be aligned on slot boundaries before they enter the switch. As these propagation distances can change because of temperature differences and other disturbances, the necessary synchronizations will vary dynamically over time and require advanced synchronization methods. The packet size in slotted networks is fixed and slightly smaller than a slot length to provide guard time. In unslotted networks, packets enter the switch without being aligned and packet sizes may or may not be fixed. Because they do not require advanced synchronization methods, unslotted networks are more flexible and easier and cheaper to build. On the other hand they require more advanced scheduling algorithms in the switch as packets arrive in a less controlled manner [47, 48].

The first scientific papers that envisioned optical packet switching as the future backbone technology appeared more than 30 years ago [49]. As OPS is currently still in its research phase justifiable criticism regarding the feasibility of OPS for real-life commercial backbone use has emerged in the last decade [50–53]. Major hurdles for OPS backbone implementation are scalability to a huge number of wavelength and spatial channels, optical buffering (see below) and most importantly increasing the switching speeds to enable packet-based traffic at the required small time scale [54–56]. Moreover as all-optical implementations of these OPS elements cause major nonlinear impairments such as attenuation, noise, dispersion, crosstalk and jitter to the optical signals, reliable and high-speed optical signal regeneration solutions are required. As electronic switching continuously sets the bar higher, achieving OPS implementation on the required scale and in a cost competitive way remains a challenge to date. Despite these considerable hurdles interest in OPS as a future switching technology remains alive [57–61]. In this the rapidly increasing energy consumption of the Internet is one of the major motives to back up this vision [58, 62, 63]. Although core networks account for only a small part of this energy consumption, OPS can greatly enhance their energy efficiency. Moreover as optical packet switching can reduce the latency and thus the need for energy devouring CDNs, their implementation could act as a lever for global IT energy reduction.

1.7 Contention resolution

So far we deliberately ignored a key aspect of both optical packet and burst switching. By dropping the requirement of establishing a dedicated communication channel between origin and destination node, the network links are shared among different communication sessions. Although this results in a more efficient use of the available bandwidth and a lower latency, it also raises the concern that in these packet-based switching techniques contention may arise in the network nodes when more than one packet (or burst[1]) heads for the same output port at the same time [64, 65]. In this OBS and OPS do not differ from other packet-switched techniques in the extent to which contention occurs but rather in the way it is resolved. In electronic packet switching contention is simply resolved by storing one of the contending packets in the RAM while the other packet is sent immediately. When the output line is available again, the packet that was put aside is retrieved from the RAM and sent. As the use of electronic storage in OBS or OPS would nullify the efforts of avoiding O/E/O conversions for the data payload, alternatives have to be employed [66, 67].

A rather straightforward first way to resolve contention in an optical setting could be to simply drop one of the contending packets. Although this is possible, this would require the higher layer network protocols to be able to handle a possibly large number of packets lost along the way. This can be done by retransmitting the lost packets or by proactively sending duplicates. Both solutions however would greatly increase the effective load of the overall network (resulting in even more lost packets) and are not suitable for real-time applications in which packets have to arrive in order and with a low tolerance for latency.

A second way to resolve contention is what sometimes is referred to as space deflection [48, 67] or the hot-potato strategy [68] and what we will call deflection routing [69]. In deflection routing one of the contending packets is redirected by sending it to another node than the one stated in the header. The effect of this strategy is that, despite a detour, packets will eventually arrive at their proper destinations. Because of these detours the distance traveled and number of hops covered per packet increase, the network resources are used more extensively and the effective load of the network rises. Although some performance gains can be achieved by deflection routing [70], these are limited to cases with a low load, and performance depends

1. Unless the distinction is made explicitly, in what follows "packet" may refer to either an OPS packet or an OBS burst.

heavily on network topology and routing strategy [71]. When the load is too high and deflection routing is used excessively, the network simply saturates immediately. It is therefore advised to only use it when the network load is low [68].

A third way to deal with contention encompasses both wavelength and spatial multiplexing. Of these two, wavelength multiplexing has been studied the longest and DWDM is already used extensively in today's core networks [72, 73]. In wavelength multiplexing multiple wavelengths can be sent coincidentally on a single fiber. The concept of spatial multiplexing (SDM) has been around almost as long as wavelength multiplexing but has not been explored in as much detail up until the last decade. Different approaches for realizing SDM exist but all result in an increased number of available spatial channels within a single fiber [74–77]. Both DWDM and SDM increase the number of channels and more importantly allow altering the spatial and wavelength channel of an optical packet on the fly. As a result when contention arises both packets are sent on the same output fiber at once but on a different wavelength or spatial channel.

A fourth and last way to handle contention is some type of optical buffering. Although at first sight it seems infeasible to "store" light, several full-size and miniature implementations to delay light locally have been proposed throughout the years. In general these can be categorized in one of the two following types:

- Full-size optical buffers rely on increasing the physical length that optical signals have to traverse by sending it through a sufficiently long piece of coiled fiber to delay it for the time needed. A frequently occurring design is one with Fiber Delay Lines (FDLs), in which a packet that needs to be buffered is sent through one of a set of fibers with different lengths [78]. In the alternative recirculation design of so called fiber loops on the other hand a packet is recirculated some non-predetermined number of times in the same smaller fiber loop [79]. As the decision to send or buffer a packet for a longer time can be reevaluated recirculation after recirculation, fiber loops are more flexible and their overall performance is better than that of FDL buffers. The downsides of this setup are the severe physical impairments of the optical signal by continuous recirculation (and switching) [67] and the fact that the loops can only accommodate packet sizes smaller than their limited loop length. Because of this the packet length also directly limits the resolution of possible delays [64]. Both fiber delay line and

fiber loop designs can only provide a discrete set of delays. As opposed to electronic memory (RAM) they can thus not retrieve packets at will, which severely limits their flexibility. Moreover, although the fibers are coiled their physical size remains significant. If we take a typical OBS setting as an example (10 Gbit/s link, 100 kbit burst size) delaying a burst for 10 µs (which equals its own size) requires a fiber length of approximately 2 km [12]. Although ongoing advances in optical fiber technology allow to reduce the cross section of optical fibers and thus the footprint of coiled fiber [80], the total buffer size will, as opposed to electronic memory (RAM), always be severely limited and packet loss not avoidable.

- So called slow-light devices have a microscopic size (on-chip) and rely on decreasing the group velocity to delay packets [81, 82]. The group velocity of a wave is the velocity with which the overall shape of the wave, i.e., the envelope, propagates through space. This is different from the phase velocity, which is the velocity at which individual photons travel through the medium. The phase velocity is linked to the refraction index, i.e., the ratio between the speed of light in a vacuum and the phase velocity, which has a typical value around two for glass (fibers). When the refraction index changes rapidly over a small range of frequencies, this can be exploited to achieve a very low group velocity, thousands or millions times less than the speed of light. The mechanisms that exist to generate slow light can be divided in material dispersion and waveguide dispersion, both of which succeed in decreasing the group velocity by creating a narrow spectral range with a high dispersion peak. A key aspect of these slow-light devices is that the product of the delay and the bandwidth for a given device length is roughly constant, i.e., by delaying a packet, its bandwidth at the output will decrease. Increasing the delay (for a given device length) will thus always come at the cost of an even lower bandwidth. This means that before the packet re-enters the system its bandwidth has to be restored (by the same mechanism). It is usually assumed that slow light devices will use the same principle design as delay line buffers, i.e., either an FDL type buffer with a choice of multiple device lengths or a fiber loop recirculation buffer with an adjustable number of recirculations [83]. As opposed to delay line buffers, slow-light devices have the desirable advantage of a continuously variable storage time as the delay of a device can be changed in between packets. Nevertheless they do not harness the power of retrieving data at will, as most are not able to change the storage time when a packet occupies the device [82]. To date only very smalls storage capacities are achievable, which

amount to typically less than a single burst. Together with their high cost, signal dispersion and loss they are practically limited in their use as buffers and have not yet made it to OPS or OBS testbeds [82, 83].

1.8 Demarcating our playground

In the publications [84–91] that provided the material for this dissertation we mainly focused on contention resolution in individual network nodes. As an effective implementation of deflection routing is managed on a network level, we thus did not include it in our analysis. Nevertheless deflection routing is not incompatible with our implementation of contention resolution in individual nodes. Indeed, even if local contention resolution is present, packet loss is not avoidable. Deflection routing can thus be used additionally, as a last resort to avoid packet loss of all or certain QoS packet classes.

Within individual nodes we focus on fiber loops (Chapter 5), FDLs (Chapter 4) and the combined use of wavelength converters and FDLs to resolve contention (Chapters 2-4). As wavelength multiplexing (DWDM) shows more technological maturity than spatial multiplexing (SDM), we assumed wavelengths converters (WCs) to be the underlying technology used to achieve multiplexing over multiple channels within a single fiber. Nevertheless, our findings and results can be largely translated to a setting in which SDM, or both DWDM and SDM combined, are used to achieve multiple channels on a single fiber. Because of this generality of our results, the generic term “channels” is preferred over “wavelengths”.

As in fiber loop buffers the number of recirculations, and thus the number of times a packet is sent over a (small) switch, is not determined in advance, different packets exit the buffer stage with different signal-to-noise ratios. As this signal distortion is moreover a highly nonlinear effect, this complicates the signal regeneration stage after the buffering stage. This may be a main reason why an FDL buffer setting is encountered the most in literature and in testbeds [92]. Nevertheless, because of their smaller footprint and bigger flexibility, fiber loop buffers remain of interest and are therefore, although to a smaller extent, also studied in this thesis.

Both packet (OPS) and burst (OBS) networks can use different means of contention resolution to avoid packet/burst loss. As bursts have a significantly longer length, optical buffering in the network nodes requires significantly more space for an OBS setting. In current prototypes, OBS is therefore often

implemented without optical buffering; i.e., only relying on deflection routing and wavelength and spatial multiplexing to avoid contention [93]. Despite this, results indicate that for both OBS and OPS networks a combined use of wavelength conversion and optical buffering provides the best means to avoid packet loss [67, 94–97]. Indeed, even a strictly limited number of FDLs can greatly reduce the loss probability (LP) and delay of certain QoS burst classes. Because both OPS and OBS are within the scope of our setting, both fixed and variable sized packets are used throughout the analysis. Although both slotted and unslotted operations exist within OPS, we will assume an unslotted setting. Similar to the OBS setting, they do not require advanced synchronization methods, and are thus more flexible and easier and cheaper to build.

Below we go into more detail on the possible types and physical structure of fiber loops, fiber delay lines (FDLs) and wavelength converters (WCs). We also devote special attention to how the combined use of FDLs and WCs is implemented most effectively within a network node.

1.8.1 Optical buffer and wavelength converter position

Similar to the electronic counterpart where RAM is used in electronic switches, optical buffering to resolve contention can be done in three distinctive locations: at the input fibers, output fibers or shared among several output fibers. The biggest drawback of buffering at the input fibers is that a temporarily high number of packets for a single output fiber can cause congestion in one or more input buffers and as such block packets destined for other, idle output fibers. This phenomenon is known as head-of-line-blocking (HOL-blocking) and is the main reason why output buffering is preferred when there is sufficient internal bandwidth available in the switch. In Fig. 1.6 a general example of HOL-blocking is shown. In (dedicated) output buffering buffer capacity is dedicated to each output fiber. Shared buffering can be seen as a modified version of output buffering in which the buffer is shared among multiple (often all) output fibers. As the buffering capacity is shared in this setting, it has a lower overall cost but also results in an unavoidable lower performance than dedicated output buffering.

Different wavelengths on a single input fiber are (de)multiplexed before/after they enter/exit the spatial switch by optical wavelength sensitive (de)couplers. These are often implemented by means of prisms and are depicted as triangles in Fig. 1.7, which shows an optical switch with a shared optical buffer and wavelength converter pool. Wavelength converters and

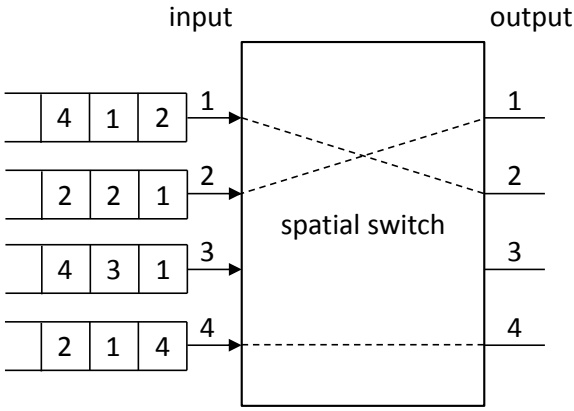


Figure 1.6: A generic example of head-of-line blocking in a spatial switch with FCFS input buffering. The second packet in the queue at input 3 cannot reach output 3 as the packet in front of it is blocked because of contention for output 1 [4].

optical buffer are more commonly used in the same manner, i.e., either both are used in a shared manner (as shown in Fig. 1.7) or both are dedicated to single output fibers (as shown in Fig. 1.8). Irrespective of the shared or dedicated nature of the optical buffer and wavelength converter pool, they can be probed in different order [71]. This order, with either conversion before or after buffering, is often set based on the node dimensions, i.e., the size of the optical buffer and the number of wavelength converters. Changing the order in which both are used dynamically, i.e., on a packet to packet basis, does not seem inconceivable at first sight but has not been encountered in literature. After all allowing the order to change dynamically does not only complicate the control logic but also the physical realization of the internal node architecture.

This consideration can be generalized more broadly as in all-optical nodes functionality is closely related to architecture and physical realization [71]. As a consequence a bewildering amount of, sometimes slightly, different architectures and accompanying performances can be found throughout literature [98–109]. We will therefore always strive to make our assumption regarding node (component) architecture as general as possible, allowing to shift the focus from the physical realization to the actual control algorithms. This allows to develop algorithms of which the fundamental groundworks are applicable to a wide variety of node architectures in which optical buffers and wavelength converters are used jointly to resolve contention. It is against

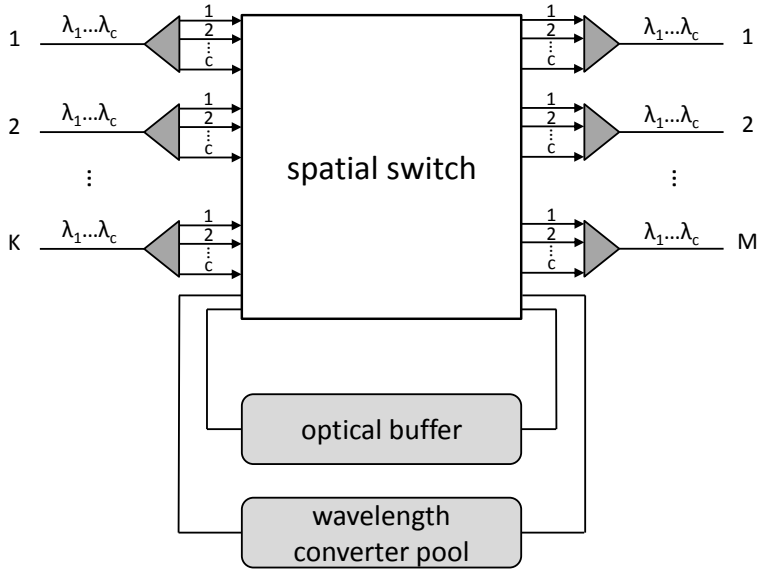


Figure 1.7: A K x M switch with c wavelengths on each input fiber. Both the optical buffer and wavelength converter pool are shared among the M output fibers. Input fibers are demultiplexed before entering the spatial switch [4].

this background that throughout this work a setting in which optical buffers and wavelength converter pools are dedicated per output fiber is chosen over a setting in which these are shared over multiple output fibers. The shared setting can indeed be seen as a mere constricted case of the dedicated setting. Resource constraints because of the shared nature and network routing concerns will very likely lead to deviating values of the optimal algorithm parameters but often will likely follow the same underlying principle. Our analysis will thus be confined to a single output fiber. In the example of Fig. 1.8, which shows a dedicated conversion before buffering setting, this is marked by the dashed-line box.

1.8.2 Fiber Delay Lines (FDLs)

In the above we analyzed the broader context of an optical buffer, i.e., the location within the switch and its interaction with the wavelength converters. We now go into more detail on the physical structure and different types of FDL buffers.

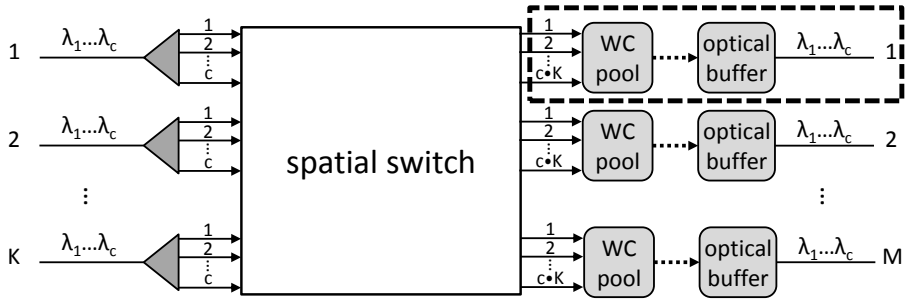


Figure 1.8: A K x M switch with c wavelengths on each input fiber. A wavelength converter pool and optical buffer are dedicated to each output fiber.

In Fig. 1.9 an abstract illustration of a WC pool and FDL buffer corresponding to the dashed line box from Fig. 1.8 is shown. Irrespective of the order of the WC pool and FDL buffer, an output switch is provided after the FDL buffer to group the optical signals on a single or the appropriate fiber. In Fig. 1.9 a single output fiber is shown. This corresponds with the conversion before buffering and single fiber per port setting of Fig. 1.8 in which case the spatial multiplexer can be implemented by a simple optical coupler. Similarly at the input of the FDL buffer an input spatial switch allows the packets to be directed to the correct fiber delay line requested by the control logic.

Intrinsically fiber delay lines can accommodate multiple overlapping packets on different wavelengths simultaneously. In the case of conversion after buffering additional decouplers may be needed to allow overlapping different wavelength signals, exiting a single delay line, to enter the wavelength converters separately. In the conversion after buffering setting the demultiplexing done at the input fibers of the overall switch is thus partially nullified in the buffering stage resulting in redundant equipment and operations.

Although in theoretical studies, an infinite number of fiber delay lines is sometimes assumed to reduce mathematical complications, in practice and real life settings the number of fibers is always limited and, therefore, an important parameter regarding in-node and overall-network packet loss and latency. As a simulation methodology is used in this work and assuming infinite size provides no benefit in the modeling, we will always work with a limited number of fiber delay lines. The number of fiber delay lines is denoted by $N+1$ as in general N non-zero delay lines and a single zero delay line are present in a single FDL buffer.

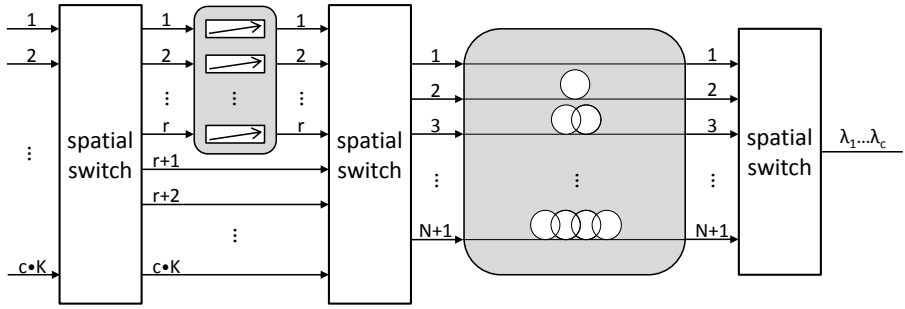


Figure 1.9: A WC pool (left) with r wavelength converters and an FDL buffer (right) with N non-zero delay lines as part of a setting in which both are dedicated to an output fiber in a conversion before buffering setting. This corresponds to the dashed-line box in Fig. 1.8.

Although the major subdivisions of optical buffers are fiber delay lines and fiber loops, based on a few key aspects a further refinement of different types of fiber delay line buffers is given in literature [4, 48, 67]:

- Feedforward or feedback: In a feedback FDL buffer packets can be sent back to the input after their exit from the FDL buffer by means of a feedback line. For the passage of a single packet multiple delay lines can thus be combined in order to achieve a wider choice of assignable delays. This is as opposed to a feedforward FDL buffer, in which each packet can only be sent through a single delay line and the buffer should therefore use an equal number of delay lines as the desired variation in delay times. Because, similarly to fiber loops, the number of recirculations is not determined in advance, comparable issues regarding a varying amount of amplifier noise and signal regeneration are present in feedback FDL settings.

- Number of stages: In multi-stage FDL buffers, several single-stage FDL buffers (as shown in Fig. 1.9) are connected in series to provide a similar set of possible combinations as in an FDL feedback setting. When traversing the multi-stage FDL buffer, packets go through a single FDL in each stage resulting in more fine-tuned delays at the end. Because, as opposed to the feedback setting, different packets always go through an equal number of lines, i.e., the number of stages, less issues regarding amplifier noise and signal regeneration are present in multi-stage buffers compared to feedback buffers. Nevertheless multi-stage FDL

buffers are rarely come across in testbeds as they are more expensive and have a larger physical footprint. Moreover, they suffer from blocking, i.e., they do not provide full "non-blocking switch" functionality.

- Degenerate or non-degenerate: Although the lengths of the fiber delay lines can be chosen in any way [110], typically they are chosen consecutive multiples of a basic value D called the granularity [111]. In these buffers incoming packets sent through the j-th $(j = 0 \dots N)$ delay line thus encounter a delay of $j \cdot D$. This type of FDL buffer is often referred to as a degenerate buffer although other equivalent terms as equidistant [112] and uniform [110] can be found in literature. Although non-degenerate FDL buffers, which evidently have a different set of fiber lengths, can offer significant performance benefits [110] the optimal set of delay lines is intimately related to the nature of the arrival process and cannot be generalized. They are thus less focused on in literature than the more general applicable degenerate setting.

Because of the drawbacks of their alternatives and for sake of universality, in this work all of our settings are assumed to be dedicated single-stage degenerate feedforward FDL buffers. Although many of the new algorithms, and especially their underlying strategies, could also prove to be useful in other settings, a straightforward extension is often not possible and is therefore considered to be out of the scope of this work.

1.8.3 Fiber loops

Fig. 1.10 shows a detailed figure of a possible fiber loop implementation [113]. This consists of the input fiber, the output fiber, four semiconductor optical amplifiers (SOAs), two optical duplicators and two optical couplers. Depending on the situation packets can enter, recirculate or exit the fiber loop by switching different semiconductor optical amplifiers (SOA) in the pass-through or block state. Packets can also circumvent the fiber loop altogether by putting SOA_{line} in the pass through state and SOA_{in} in the block state. As opposed to the FDL buffer no additional fiber line (the uppermost delay line in the FDL buffer of Fig. 1.9) is thus needed to provide a zero delay opportunity to a packet.

Although multi-wavelength operation of a fiber loop is possible and has been shown in practice [114], it requires a set of SOAs for each wavelength and thus a lot of overhead components. This is as opposed to an FDL buffer in

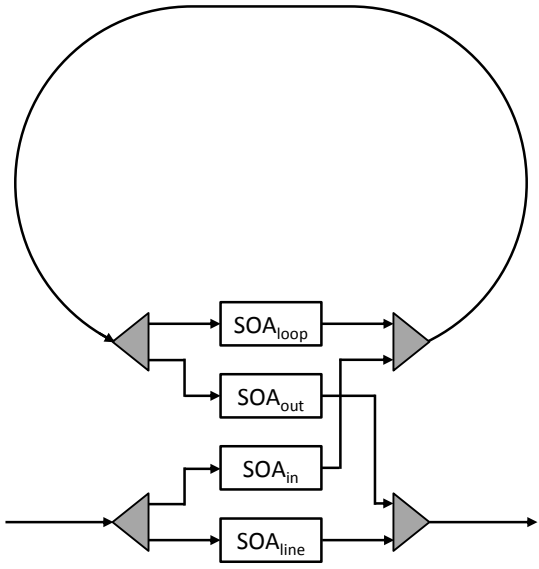


Figure 1.10: Possible Fiber loop implementation using four semiconductor optical amplifiers (SOAs), two optical duplicators and two optical couplers.

which packets on different wavelengths on the same fiber line experience the same delay and can simply be multiplexed. As the delay for a packet is not necessarily decided upon arrival of the packet but can be reevaluated after each recirculation, fiber loops however, are more flexible than FDL buffers. Algorithms using this strategy are called post-reservation schemes and differ from the so-called pre-reservation schemes, a term coined by [115], that reserve the outgoing channel upon arrival of the packets. Pre-reservation algorithms are used for FDL buffers but also for fiber loop buffers as they require a less complicated control. In that case a fiber loop buffer can be seen as merely a more compact implementation of an FDL buffer (without the added flexibility).

With each recirculation packets experience signal distortion passing through the SOAs and optical couplers. As the number of recirculations is different for each packet, the signal regeneration process is further complicated. Although increasing the fiber loop length (and thus reducing the number of recirculations) can partially alleviate this concern, it directly impacts the possible set of delays that can be assigned to a packet. As the maximum packet size is also directly linked to the fiber loop length, all three parameters (packet size distribution, fiber loop length and maximum number of recirculations) are interconnected and can thus not be set independently.

Different variations of fiber loops as for example dual loops [116] exist but are not discussed in detail. Indeed, although they have a different physical structure, these alternative implementations have similar fundamental characteristics and limitations such as finite packet length and a discrete set of delays. Because of this, the design principles of the scheduling algorithms designed for the classic fiber loop setting can often be tailored to these alternative designs.

1.8.4 Wavelength converters

Today the main (if not only) commercially deployed wavelength converters are of the optical-electronic-optical (OEO-WC) type [117]. These are straightforward in implementation and consist of a detector followed by a laser that retransmits the incoming signal on the new wavelength. As OEO-WCs convert the signal to and from the electronic domain, they are not preferred in all optical network applications as OPS and OBS. Moreover they suffer from complexity and energy consumption issues [118, 119].

As a consequence interest has shifted to all-optical wavelength converters (AO-WC) for future networks [117]. The simplest way to perform all-optical wavelength conversion is by means of an optically controlled laser converter in which the input signal to be converted is launched into a laser to cause gain saturation. By doing so the oscillation of the laser, which has a different wavelength, is changed to the same signal. The output wavelength can be either fixed or tunable by an electronic signal from the control logic. Although conceptually simple, these laser converters are limited in speed and are therefore less interesting for optical network applications.

A second type of all-optical wavelength converter is the four wave mixing (FWM) converter using semiconductor optical amplifiers (SOA). Although these converters have no speed limitations, they do result in high levels of noise on the output signal which is a particular concern when they are cascaded. An additional drawback is that the output wavelength depends on both the pump (=control) wavelength and the input wavelength. As a consequence the pump must be controllable even for converters with fixed output wavelength.

A third, and most widely researched, type of AO-WC is based on optically controlled gates. This type of AO-WCs, which is also based on semiconductor optical amplifiers (SOA), has a cross gain modulation (XGM) or cross phase

modulation (XPM) architecture. Because of this, they do not suffer some of the drawbacks of FWM all-optical converters and are found regularly in current testbeds.

Independent of the exact implementation, all AO-WCs suffer from similar concerns and bottlenecks that prevent them to be applied in commercial and industry-level settings: low bit rate, low signal-to-noise ratio, moderate input power levels, limited wavelength span (both input and output), and setup time.

Although current research heavily focuses on alleviating these bottlenecks, they nevertheless show how in all-optical nodes functionality is closely related to physical realization. Because the focus of this work is on control logic we will assume most of these limitations to be softened to a level where they pose no constraints on our scheduling algorithms. For example we will always assume that a wavelength converter can convert a packet, irrespective of its wavelength, to any of the available output wavelengths. We will thus not consider the case of converters with a limited wavelength conversion capability in which incoming packets can only be converted to nearby wavelengths on the spectrum despite the popularity of this type of converter [120–122]. By relaxing this and other restrictions on the wavelength converters we strive to make our scheduling algorithms as generally applicable as possible. Indeed, it makes more sense to adapt algorithms designed for hypothetical ideal conditions to constrained situations than the other way around. Although we do not take into account the physical limitations of the wavelength converters themselves, we will consider the number of WCs to be a constraining factor (r wavelength converters in Fig. 1.9). In general we assume tunable wavelength converters (TWCs) to be present although a certain number of TWCs in general can be replaced with a larger set of (cheaper) fixed output wavelength converters [123].

We already mentioned that a disadvantage of the conversion after buffering case is that additional decouplers may be needed after buffering to allow packets to enter the wavelength converters separately. However, an advantage of this setting is that the available wavelength converters are used more effectively. When, for simplicity, for example fixed output wavelength converters are used, the number of required WCs in the conversion after buffering case is limited to the number of wavelengths on the output fiber to achieve full wavelength conversion capability. Indeed, as packets on the same output wavelength no longer overlap after the buffer stage, a single WC per output wavelength suffices. This is not the case for the conversion

before buffering case in which packets destined for the same output wavelength may still overlap at the conversion stage. As a result more than c wavelength converters may be required simultaneously. On the other hand in the conversion after buffering setting, two packets on the same incoming wavelength can never enter the same FDL or fiber loop simultaneously, even though they end up on a different wavelength at the output. We can thus say that in general, the second stage, whether it is the optical buffer or WC pool, can be used more effectively than the first stage. As our focus is on improving the effective use of optical buffers we will always assume a conversion before buffering setting.

1.9 This dissertation

1.9.1 Overview and structure

In this doctoral dissertation we focus on contention resolution in individual network nodes by means of optical buffers (both fiber loops and fiber delay lines) and wavelength converters. In this we are mainly interested in the way these components are controlled rather than their physical structure and limitations. With the exception of the prior sections of the Introduction chapter we will thus largely ignore physical constraints and implementations and focus on the so-called scheduling algorithms.

Scheduling algorithms control both the delay and wavelength assignment of the arriving packets by taking into account both the availability of wavelength converters and optical buffer capacity. As, because of physical limitations, the capacity of both wavelength converters and optical buffers is always severely limited, select contention resolution algorithms make a substantial difference in performance and lie at the heart of effective contention resolution. Throughout this dissertation several different performance measures are used to evaluate the performance of the algorithms in its various aspects. Of these, given the importance of limiting data loss and latency, loss probability (LP) and average packet delay are the most important ones.

From the first sections of the Introduction chapter it is clear that the assumed physical implementation and physical restrictions can have a major impact on the achievable performance of the scheduling algorithms. Indeed, in the case of, for example, a fiber loop buffer, the consecutive amount of signal distortion and the efficacy of the subsequent signal regeneration directly limits the exact number of recirculations a packet can take. When using wavelength

converters, the wavelength span differs for each physical implementation. These and many other (interacting) physical boundary conditions strictly confine the playground of any real-life or prototype scheduling algorithm. As our focus is on the actual control aspect of the algorithms, we therefore choose to make these boundary conditions as general as possible. Although this limits the direct implementation of these algorithms, it allows to develop algorithms of which the fundamental groundworks are applicable to a wide variety of node architectures and physical implementations.

In section 1.8 this guiding mantra was used to motivate specific assumptions. Below the most important assumptions are repeated in bullet point form together with an overview of the content of each chapter (chapters 2-5). As such we wish to give the reader a quick and uncluttered overview of this dissertation. More details on the assumptions can be found in section 1.8 and the corresponding chapters.

General assumptions:

- no deflection routing
- no spatial multiplexing (within a single fiber)
- applicable to both OBS and OPS
- continuous time (i.e., arrivals are not synchronized to a clock)
- optical buffer and wavelength converters dedicated per output fiber
- both fixed and variable sized packets
- FDL buffer:
  - degenerate
  - single stage
  - feedforward
- fiber loop buffer:
  - fixed length set
  - post reservation
- wavelength converters (WCs):
  - unlimited wavelength span (both input and output)

  - zero setup time

In **chapter 2** we first give an introduction to the basics of scheduling when FDL buffers and wavelength converters are present. Next we focus on parametric cost-based scheduling algorithms that take into account both gap and delay characteristics to schedule packets. By varying the gap-delay balance, we study and optimize the achievable performance improvement in terms of packet loss compared to existing benchmark algorithms.

- FDL buffer + wavelength converters
- cost-based algorithms: gap + delay

In **chapter 3** the cost-based scheduling algorithms of Chapter 2 are extended with an extra term to penalize the use, and thus energy consumption, of the wavelength converters. Although energy consumption reduction is the main motivation for this extra cost, in the case of a limited number of wavelength converters, this also results in a more effective use of a scarce resource and thus an improved performance (in terms of packet loss) compared to the algorithms of Chapter 2. By means of a detailed performance analysis, the exact performance improvement and energy consumption reduction are investigated and quantified.

- FDL buffer + wavelength converters
- cost-based algorithms: gap + delay + wavelength converter cost

In **chapter 4** we focus on a type of algorithms that optimizes the creation and filling of idle periods in between packets; i.e., the voids. By doing so, these so-called void-creating algorithms, can significantly improve performance in terms of packet loss and packet delay. As these algorithms are a completely new type of algorithms, we focus on the mathematical framework in order to maximize the insight in their operation.

- FDL buffer
- void-creating algorithms

In **chapter 5** new performance improving post-reservation algorithms are developed for a setting with fiber loops and a uniform distributed packet size. As the focus is put on the control for the fiber-loop buffer, wavelength converters are not taken into account. By delaying packets longer than strictly

necessary, the newly developed algorithms are able to outperform existing best in class algorithms in terms of different performance measures. The exact performance gains are evaluated for different settings.

- fiber loop buffer
- post-reservation algorithms

1.9.2 List of publications

The following publications have provided the material for this doctoral dissertation. Note that the bibliographic data is followed by a reference to the relevant chapters.

International journals (A1):

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Improving the energy efficiency of scheduling algorithms for OPS/OBS buffers", *Photonic Network Communications*, vol. 29, no. 2, pp. 183-197, April 2015. **Chapters 2 and 3**
- K. Van Hautegem, W. Rogiest and H. Bruneel, "Improving performance and energy consumption of shared wavelength converters in OPS/OBS", *Optical Switching and Networking*, vol. 17, pp. 15-38, July 2015. **Chapters 2 and 3**
- K. Van Hautegem, W. Rogiest and H. Bruneel, "Selective void creation/-filling for variable size packets and multiple wavelengths", *Journal of Industrial and Management Optimization*, vol. 13, no. 5, pp. 1-18, January 2017. **Chapter 4**

Conference proceedings (full paper):

- K. Van Hautegem, W. Rogiest and H. Bruneel, "OPS/OBS scheduling algorithms: incorporating a wavelength conversion cost in the performance analysis", *Proceedings of the IEEE 32nd International Performance Computing and Communications Conference (IPCCC)*, pp. 1-9, San Diego, USA, December 2013. **Chapter 3**
- K. Van Hautegem, W. Rogiest and H. Bruneel, "Scheduling in optical switching: Deploying shared wavelength converters more effectively", *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 3418-3424, Sydney, Australia, June 2014. **Chapters 2 and 3**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Void-creating algorithm in OPS/OBS: mind the gap", *Proceedings of the AIP International Conference of Numerical Analysis and Applied Mathematics (ICNAAM)*, Rhodes, Greece, September 2014. **Chapter 4**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Fill the void: improved scheduling for optical switching", *Proceedings of the 27th International Teletraffic Conference (ITC)*, pp. 82-88, Ghent, Belgium, September 2015. **Chapter 4**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Optical Switching for Variable Size Packets: Improved Void Filling through Selective Void Creation", *Proceedings of the 11th International Conference on Queueing Theory and Network Applications (QTNA)*, pp. 1-8, Wellington, New Zealand, December 2016. **Chapter 4**

- K. Van Hautegem, M. Pinto, W. Rogiest and H. Bruneel, "Analysis of VAS, WAS and XAS scheduling algorithms for fiber-loop optical buffers", *Proceedings of the 13th International Conference on Queueing Theory and Network Applications (QTNA)*, Tsukuba, Japan, July 2018. **Chapter 5**

Conference proceedings (abstracts):

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Using shared wavelength converters effectively in optical switching", *Book of abstracts of the BESTCOM meeting - Fall 2013*, Leuven, Belgium, October 2013. **Chapters 2 and 3**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "OPS/OBS scheduling algorithms: incorporating a wavelength conversion cost in the performance analysis", *Book of abstracts of the 7th Young European Queueing Theorists (YEQT) Workshop 'Scheduling and priorities in queueing systems'*, Eindhoven, Netherlands, November 2013. **Chapters 2 and 3**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Void-creating algorithms for fixed packet length in OPS/OBS", *Book of abstracts of the first European Conference on Queueing Theory (ECQT)*, p. 45, Ghent, Belgium, August 2014. **Chapter 4**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Void-creating algorithms for fixed size packets in optical switching", *Book of abstracts of the 15th FEA Research Symposium*, Ghent, Belgium, December 2014. **Chapter 4**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Mind the gap: void-creating algorithms for optical switching", *Book of abstracts of the BEST-COM meeting - Fall 2015*, Leuven, Belgium, October 2015. **Chapter 4**

- K. Van Hautegem, W. Rogiest and H. Bruneel, "Void creation: reculer pour mieux sauter", *Book of abstracts of the BESTCOM meeting - Spring 2016*, Ghent, Belgium, April 2016. **Chapter 4**

1.9.3 Methodology: Discrete Event Simulation

To evaluate the performance of scheduling algorithms used in OPS and OBS networks many different methods can be used. If possible, analytical methods resulting in closed-form expressions are preferred, as these allow to evaluate the performance of the system, or at least an approximation thereof, instantaneously and for all parameter combinations. This approach, which can be used for both continuous time and discrete time settings, may capitalize on techniques developed for classic (non-optical) buffers such as transform functions and allows to fit the analysis within the general framework of queueing theory [12]. Despite their clear advantages, these analytical techniques are limited to basic scheduling algorithms, e.g., FCFS, in the setting of optical networks. Alternatively analytically or numerically solvable Markov chains may be used to model the embedded transitions of various system state variables.

Despite the wide range in analytical solution methods, many real-life systems are so complex that their behavior is too intricate to analyze with analytical techniques. This is the case for the systems investigated in this work as the proposed scheduling algorithms overstep the simplicity of more basic scheduling algorithms suitable for the above methods. Although a simulation methodology is used throughout this dissertation, wherever possible we try to back our algorithms by a mathematical framework that helps construct our algorithms in a sensible way. This framework deepens the insight in the mechanism of the algorithms and makes it possible to apply them in other settings without (a lot of) additional simulations.

Although few systems in practice are completely discrete or continuous, most systems can be categorized as either the one or the other [124]. In continuous systems the state of the system, i.e., the condition in which the system is, continuously changes over time (often described by a set of differential equations). Because of this, in continuous simulation time is broken into

small pieces called time slices. At each ending of a time slice the system state is possibly changed based on the events happened in the last time slice. Both the accuracy of the results and the time it takes to complete the simulation (a.k.a. the computational complexity) highly depend on how small and dense these time intervals are chosen.

Discrete systems on the other hand have a state that can only change at a discrete set of points in time. In so-called discrete event simulation (DES) the system can thus be modelled as a sequence of events marked by their particular instant in time, i.e., the simulation is event-based. The system state changes from one "event" to the next and does not change in between these events. Because DES simulations do not simulate every time slice, they are far more efficient in terms of computational resources than continuous simulations. Queueing systems are textbook examples of discrete systems as the number of customers or packets in line only changes when a new customer arrives or the service of a customer is completed. Together with the system state variables, there are a few other key components that constitute a DES. Below we briefly summarize the function of these components for optical queue systems as discussed in this dissertation. The exact implementation for each setting will be discussed in more detail in the relevant chapter.

- **(System) state variables** contain all necessary information to describe and characterize the system unambiguously at any point in time. The state variable of queue simulations will almost always contain variables related to the number and type of customers in the queue. For optically buffered nodes the complexity of the state description highly depends on the type of scheduling algorithm. Sometimes a single value can describe the state of the buffer, sometimes multi-dimensional matrices are needed if the relative positions of all packets matter.
- **The clock** keeps track of simulated time in whatever unit suitable for the system being modelled. For real-life examples of queues this can be seconds or another standardized unit. For our simulated optical nodes time units will often be chosen in relation to (average) packet sizes. The clock changes from one event to the other with no changes in between and need not to be linked to the internal clock of the machine on which the simulation is run.
- **An event** is an instantaneous occurrence that changes the state of the system. In queueing systems this is often the arrival or departure of a new packet/customer. Besides changing the system state, events can

also generate new events. The arrival of new packet will for example often create the event corresponding with the next arrival by adding a random inter-arrival time to the current arrival. Each event also has a time-stamp that changes the system clock at the moment it is processed.

- **The events list** or **agenda** lists all future events together with their corresponding time-stamp. The events in the agenda are processed in chronological order. After the system state and clock are changed and the new events are added to the agenda, the current event is crossed out from the agenda and the agenda is reordered chronologically. In a similar way the earliest event in the agenda is processed again and again (change system state, create new events and reorder agenda). This is repeated until a fixed number of arrivals have occurred. The number of simulated arrivals, together with the number of runs, is often chosen in such a way that the obtained confidence intervals on the target statistics are small enough while maintaining the simulation time within reasonable bounds.
- **A (pseudo)random number generator** is used when variables are not fixed but rather sampled from specific distributions. Although sequences that are closer to truly random can be achieved by using other random number generating techniques, a standard pseudorandom number generator such as the popular Mersenne Twister suffices for the task at hand and is preferred for its speed and reproducibility.

Besides the above static description of the components of a DES, the dynamic relationships and interactions between these components are perhaps even more important to correctly describe the analyzed system. Some important questions that lie at the heart of designing discrete event simulations include [125]:

- How does each event affect the system state?
- What events should trigger the creation of other events?
- Can new events be created regardless of the system state or is its creation conditioned on the system being in a certain state?
- Which combinations of events and system state trigger packet loss?
- Under what condition does a delay begin or end?
- How is the system initialized at time 0?

- What statistics need to be tracked at which moments?
- ...

To come up to and, most of all, answer all of the relevant questions often multiple iterations in the design process are needed. Moreover well-chosen test cases are used to assure the dynamics of the system are captured correctly by the above components and their interactions.

Chapter 2

Scheduling basics and cost-based algorithms: gap + delay

X-rays will prove to be a hoax.
Lord Kelvin, President of the Royal Society, 1883

In this chapter we first give an introduction to the basics of scheduling when an FDL buffer and WCs are present. This includes reviewing the assumptions for this chapter, discussing existing algorithms and going into more detail on their dynamics by means of a basic example. Next we focus on parametric cost-based scheduling algorithms that take into account both gap and delay characteristics to achieve a superior performance.

2.1 Scheduling basics

2.1.1 Assumptions

As mentioned in the introduction, the combined use of an FDL buffer and WCs is the most effective solution to avoid contention. As our prime focus is on a more effective use of the FDL buffer we assume a dedicated conversion before buffering setting corresponding to the setting of Fig. 1.8 and 1.9. This setting assumes a $K \times M$ optical switch configuration. Packets arrive on a finite number of incoming ports K, on c different wavelengths $\lambda_1, \ldots, \lambda_c$, also called channels. Each packet arrives on a certain wavelength and is

switched (still on this wavelength) to one of the M output ports according to the packet header destination information. Each output port thus accepts packets from K ports, on c wavelengths. The output port is connected to a single fiber with the same c different wavelengths. The scheduling algorithm, which operates at output port level, controls the entrance to the converters (first stage) and the FDLs (second stage).

We assume that at each of the M output ports the global packet arrival process is a Poisson process with the wavelength w of each arriving packet randomly and uniformly distributed among the c wavelengths. A Poisson arrival process on each wavelength at the K input ports combined with a non-blocking switch architecture is an example of a set-up that results in this packet arrival process at the output ports. In this chapter an arbitrary single output port is analyzed, marked by the dashed-line box in Fig. 1.8. The length of arriving packets, B, is assumed exponentially distributed with an average of $E[B]$ time units. With the assumption of a Poisson process for the arrival process at the output port, the inter-arrival time T is thus exponentially distributed with an average $E[T]$. Given the nature of the Poisson process and the random and uniform distribution of the wavelength w among the c possibilities the inter-arrival times on each wavelength are also exponentially distributed but with an average of $c \cdot E[T]$. Related, the overall incoming traffic load at the output port is given by $\rho = E[B]/(c \cdot E[T])$.

For the FDL buffer the above setting corresponds to the output-buffered switch architecture in [109] named "dedicated FDL buffers per fiber". Each output port has an optical buffer in which $N+1$ FDLs are available to schedule incoming packets. The lengths of the FDLs are consecutive multiples of a basic value D called the granularity. As mentioned this is called a degenerate delay buffer, in which incoming packets sent through the j-th $(j = 0 \ldots N)$ delay line encounter a delay of $j \cdot D$. As was motivated in Section 1.8 we moreover assume our FDL buffer to be single-stage, i.e., with only one set of FDL lines, and feedforward, i.e., each packet can only be sent through a single FDL. All other physical limitations regarding the FDL buffers are disregarded, making the boundary conditions as general as possible to focus on the actual control aspect of the scheduling algorithms.

Besides an FDL buffer wavelength converters are available at the output port to change the wavelength of an arriving packet if demanded so by the scheduling algorithm. Either full wavelength conversion capability or a limited but dedicated set of r wavelength converters is assumed. With full wavelength conversion capability, whenever a packet is scheduled on a dif-

ferent wavelength than the incoming wavelength, wavelength conversion is possible. This is as opposed to the case when only a limited set of wavelength converters are available in which case at least one of the r WCs has to be available before scheduling a packet on another wavelength. Independently of the limited or unlimited nature of the WC pool, all limitations of the WCs are softened to a level where they pose no constraints on our scheduling algorithms. For example, we assume WCs can convert any of the c incoming wavelengths to all other wavelengths of the set of c wavelengths. Both the input and output span of the WCs is thus assumed to cover the complete wavelength spectrum of the setting. As both the target and origin wavelength can change from packet to packet, the WCs are thus moreover assumed to be tunable (TWC).

2.1.2 The provisional schedule

At the output port under study, scheduling is done upon arrival of each subsequent packet. Scheduling amounts to the assignment of two variables (i and j) to the given packet, corresponding to the outgoing wavelength ($i = 1 \dots c$) and the delay line ($j = 0 \dots N$). This is best done by means of a provisional schedule which shows the outgoing future line's occupation upon and in reference to each new packet arrival. An example of a provisional schedule is shown in Fig. 2.1. The already scheduled packets upon arrival of an arbitrary packet that is yet to be scheduled (and is thus not displayed on the provisional schedule) are shown as grey boxes. The arrival instant of a packet corresponds to the zero delay reference line and the provisional schedule of each outgoing wavelength ($i = 1 \dots c = 4$) relative to this arrival is represented horizontally, line by line. The vertical lines represent the delays of the FDLs ($j = 0 \dots N = 5$). The granularity in the example is assumed as unity ($D = 1$) to ease notation.

In this representation, the provisional schedule evolution can be seen as a choppy (observed from arrival to arrival) but uniform (all packets move alike) movement of all packets to the left, with packets disappearing (because they are being transmitted) when crossing the zero delay reference line. The provisional schedule, although similar at first sight, is not to be confused with a slotted arrival process. The inter-arrival times are continuously distributed and the vertical lines represent the delays of the FDLs and not slot boundaries. Each intersection between a horizontal line and a vertical line is marked with a dot and represents a *potential scheduling point* (PSP) for the new packet. Within this set of points, a *scheduling point* (SP) is one allowing the packet to be scheduled with the used algorithm and the vacant WCs (if any),

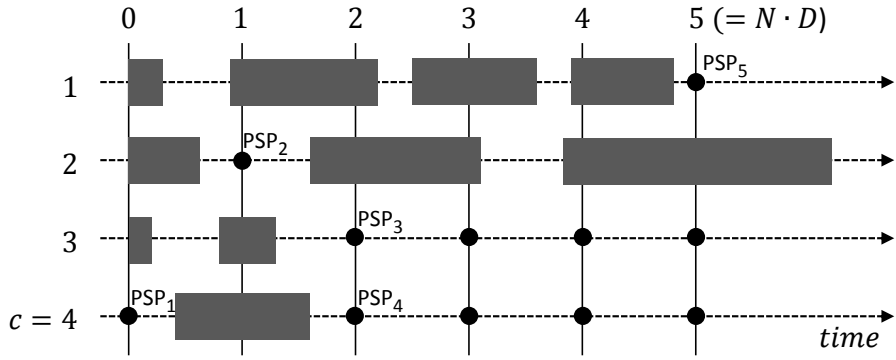


Figure 2.1: Example of a provisional schedule at an output port with the number of outgoing wavelengths (c), the number of FDLs ($N+1$) and the granularity (D) equal to 4, 6 and 1, respectively.

without overlapping with any of the packets already present in the system. Whether or not a given combination of i and j is a PSP thus depends on the system's provisional schedule. Whether a PSP is a SP depends upon both the size of the packet to be scheduled, the used scheduling algorithm and whether a WC is vacant. If no SPs are available, the current packet is lost, otherwise the scheduling algorithm decides which SP the packet is assigned to.

As mentioned, scheduling is done according to a scheduling algorithm, which either aims purely for minimal loss, or also pursues other performance measures as for example energy consumption in Chapter 3. Regardless of their exact design aim, scheduling algorithms can be split up in two main categories: void-filling algorithms and non-void-filling algorithms.

- The former allows packets to be scheduled on any PSP. This implies the possibility of filling up Voids, defined on a given wavelength as unscheduled periods followed by one or more packet already scheduled beyond it. Here, only voids overlapping a PSP can actually be used. Voids with lengths smaller than D do not always contain a PSP, rather it depends on the arrival instant whether these voids contain a PSP upon arrival of an arbitrary packet. In the example of Fig. 2.1, seven voids occur, of which two can potentially be filled (i.e., one on wavelength 2 and one on wavelength 4).
- Algorithms of the latter type only allow packets to be scheduled on a PSP if no packets are already scheduled beyond it; all other PSPs are excluded from being a SP. Packets can only join at the back and voids

between already scheduled packets cannot be filled. In this way the algorithm is not obliged to keep track of all voids in the output channels but merely of the channels' horizon, defined as the latest time at which the channel is currently scheduled to be in use. Graphically, on each channel this corresponds to the right edge of the rightmost packet. In practice, Non-Void-Filling (NVF) algorithms only allow packets to be scheduled on the first PSP to the right of the horizon. PSPs further to the right are excluded from being a SP, resulting in at most one SP per channel. Non-void-filling algorithms are less complex and easier to implement than Void-Filling (VF) algorithms but suffer larger packet loss.

2.1.3 Existing algorithms

To schedule a packet, a scheduling algorithm always bases its decision on the availability of the WCs, the provisional schedule and the incoming wavelength of the packet. Specifically, the algorithms determine the set of wavelengths to choose a SP by considering the availability of a WC. If no WC is available, the set is limited to the incoming wavelength, if one or more WCs are vacant (=available) or full wavelength conversion capability is assumed, the set comprises all wavelengths. Upon this set of wavelengths the algorithm applies the *scheduling criteria* (SCs) to choose a SP. Scheduling criteria are properties regarding the provisional schedule, the incoming wavelength and the number of available WCs. Table 2.1 shows an overview of all scheduling criteria (SCs) (see below) of the algorithms of this chapter. All algorithms, both existing and new, are given an abbreviated name in accordance with the used logic of the concerned algorithm.

The most obvious scheduling algorithm is a non-void-filling scheduling algorithm, which schedules the packet on the SP with the shortest wavelength horizon of the set of considered wavelengths, provided that at least one wavelength with a horizon smaller than $N \cdot D$ is present in this set. If no such feasible wavelength is present, there are no SPs and the packet is lost. Ties are chosen arbitrarily, unless the packet's original wavelength is part of the tie, in which case the original wavelength is chosen. This algorithm is called JSQ (Join-the-Shortest-Queue), and is discussed in detail in [126]. The used SCs are the horizon of the channels and the wavelength of the channels in case of a tie. For this algorithm the availability of the WCs is only used to determine the set of wavelengths to choose from and not as a SC.

Table 2.1: Overview of the considered scheduling algorithms of this chapter, current and new, and their scheduling criteria (SCs).

algorithm	scheduling criterium (SC)	scheduling criterium (SC) in case of a tie		
		1st order tie	2nd order tie	3rd order tie
JSQ	horizon	wavelength	random	
D-G	delay	gap	wavelength	random
G-D	gap	delay	wavelength	random
D-G-VF	delay	gap	wavelength	random
G-D-VF	gap	delay	wavelength	random
C	cost *C*	wavelength	random	
C-VF	cost *C*	wavelength	random	

In [127], two non-void-filling algorithms with better performance are presented:

- In D-G (Delay-Gap) the packet is scheduled on the SP with lowest delay, i.e., once the set of considered wavelengths is determined. The *delay* of a SP is the length of the FDL that together with the corresponding channel defines the SP. If two or more SPs have an equal delay (first-order tie), the SP with the minimal gap is chosen, which is defined as the length of the unscheduled period preceding the SP. In this way these unscheduled periods in front of the scheduled packets are minimized as a second priority. Gaps, only defined for SPs, are thus not to be confused with voids. As a third priority (second-order tie) the original wavelength is chosen again if still available. The SCs are the delays of the SPs and the gaps and wavelengths in case of a tie.

- In G-D (Gap-Delay) the gap in front of the scheduled packet is minimized as a first priority for the considered set of wavelengths. So the SP which minimizes the difference between the delay and the horizon is chosen to service the incoming packet. As a second priority (first order tie) the delay is minimized. Second-order ties are again decided in favour of the original wavelength if possible. The SCs are the gaps of the SPs and the delays and wavelengths in case of a tie.

Both D-G and G-D have a counterpart with void-filling. D-G-VF is similar to D-G with the only difference that void-filling is implemented. The D-G rules are applied on all PSPs for which the available time window between the PSP and the next scheduled packet is wide enough to fit in the current packet. Analogously, G-D-VF is G-D with void-filling.

As an illustration, we apply these strategies to the specific case of Fig. 2.1. We assume at least one WC is vacant which implies all wavelengths are part of the considered set. The scheduled packets result in a number of consecutive voids on each channel which are defined completely by their onsets and offsets. The voids with both beginning and ending equal to ∞ are called dummy voids and are used to have the same number of voids for each channel in the void matrix (see further). The horizon of each channel is marked in bold:

channel 1: $[(0.3\,;\,0.9),\ (2.2\,;\,2.5),\ (3.6\,;\,3.9),\ (\mathbf{4.8}\,;\,\infty)]$
channel 2: $[(0.7\,;\,1.6),\ (3.1\,;\,3.8),\ (\mathbf{6.2}\,;\,\infty),\ (\infty\,;\,\infty)]$
channel 3: $[(0.2\,;\,0.8)\,,\ (\mathbf{1.3}\,;\,\infty),\ (\infty\,;\,\infty),\ (\infty\,;\,\infty)]$
channel 4: $[(0\,;\,0.4)\,,\ (\mathbf{1.6}\,;\,\infty),\ (\infty\,;\,\infty),\ (\infty\,;\,\infty)]$

JSQ results in position PSP_3 as the chosen SP since wavelength 3 has the lowest horizon. D-G chooses position PSP_4 (and thus wavelength 4) since wavelengths 3 and 4 have the same delay but position PSP_4 results in a smaller gap in front of the scheduled packet. When we use G-D, position PSP_5 on wavelength 1 will be chosen because the gap is only 0.2, which is the lowest in this example. D-G-VF results in position PSP_1 if the packet length is smaller than 0.4. If bigger than 0.4 but smaller than 0.6, it will be scheduled on position PSP_2; if longer than 0.6 it will be scheduled on position PSP_4, the same position as D-G. For G-D-VF, the first choice also is position PSP_1 if the packet length is smaller than 0.4. If the packet length is larger than 0.4 it will be scheduled on the same position as G-D, position PSP_5.

2.2 Cost-based algorithms: gap + delay

2.2.1 Approach and concept

While the above algorithms all yield high performance (in terms of mitigating loss), especially the void-filling ones, they are not strictly optimal, and can be outperformed by a cost-based approach, as argued in [84, 87] for the case of unlimited wavelength conversion and in [85, 88] for the case of a limited set

of wavelength converters. In both settings this cost-based approach assigns a cost C to all SPs which equals a weighted average of the SCs delay and gap associated with the SP:

$$C = \alpha \cdot gap + (1 - \alpha) \cdot delay \ . \tag{2.1}$$

Here, we introduced α, an algorithm parameter called the *gap-delay balance*, real-valued in the interval $[0, 1]$. To obtain a "fair" representation of the relative importance of the SCs delay and gap, the weighing coefficients α and $1 - \alpha$ will be varied (see below) but always sum up to one. A first algorithm using the above cost function is named C (Cost) and is a non-void-filling algorithm which assigns the cost C to all SPs. As non-void-filling algorithms have at most one SP per channel this results in a cost C value per wavelength/channel. The second algorithm, named C-VF (Cost void-filling), is similar to the first but provides for void-filling. The class of SPs is thus larger than in the first case. However, the associated cost function is identical to that of the non-void-filling case, (2.1). For both C and C-VF, the SP with the lowest cost is chosen. Ties are broken arbitrarily, unless SPs on the packet's original wavelength are still available, in which case the SP on the original wavelength is chosen.

The operation of the algorithms critically depends on the cost function, which in turn depends on the choice of the algorithm parameter α. This can be illustrated by means of the example in Fig. 2.1. The C algorithm, assuming a WC is available, will schedule on PSP_4 for $\alpha < 0.9375$ and on PSP_5 for $\alpha > 0.9375$. When $\alpha = 0.9375$ the resulting tie will be broken based on the packet's original wavelength or randomly. The C-VF algorithm will schedule a packet smaller than 0.4 on PSP_1 as both gap and delay, and thus cost C as well, of this PSP are equal to 0. If longer than 0.4 but smaller than 0.6, it will be scheduled on position PSP_2 if $\alpha < 0.9756$ and on PSP_5 if $\alpha > 0.9756$. Packets longer than 0.6 will be scheduled on PSP_4 for $\alpha < 0.9375$ and on PSP_5 for $\alpha > 0.9375$.

2.2.2 Performance results

When considered independently of energy efficiency, the key performance measure is loss probability (LP). The focus of this chapter is therefore on the comparison of the LPs of the algorithms. The number of FDLs $(N + 1)$ is fixed at 10 and the load ρ is fixed at 80 %. The number of available channels c is assumed to equal 4 and 8 in two different sets of simulations. It is known from literature (e.g., [110]) that the LP is highly dependent on

the granularity (D), therefore in simulations D will be varied from 10 to 200 time units with steps of 10. The average packet length ($E[B]$) is assumed 100 time units. For the non-void-filling algorithms the arrival of $10 \cdot 10^7$ packets (i.e., 10 traces of 10^7) is simulated. For the void-filling algorithms the arrival of $10 \cdot 10^6$ packets is simulated to keep run-time acceptable. Although obtained in the simulations, we chose to plot confidence intervals only when they add value to the graph because they are very narrow in most instances and their inclusion would reduce the graphs' clarity.

2.2.2.1 Unlimited wavelength conversion

In Fig. 2.2 the LPs of JSQ, D-G, G-D, D-G-VF and G-D-VF with unlimited wavelength conversion are plotted as a function of D for $c = 4$. The same graphs for $c = 8$ with equal ρ (and thus, doubled arrival rate) are shown in Fig. 2.3. These are existing algorithms (proposed in literature, by other authors, see above). Among the non-void-filling algorithms G-D has the lowest loss. All three non-void-filling algorithms have a specific granularity for which minimal loss is achieved. This optimal value of D is larger for D-G than for JSQ, and largest for G-D. It is known that these optimal granularities rise with decreasing load. The existence of an optimal granularity for non-void-filling algorithms can be explained intuitively: if D is too small, also the maximum achievable delay $N \cdot D$ is small, and packets cannot be delayed long enough to avoid contention with previously scheduled packets, and hence are lost. However, if D becomes too large, the gaps (which are in the range $[0, D]$) become too large. As such, the optimal D is found for some intermediate value. This is as opposed to the void-filling algorithms, for which the LP consistently decreases with increasing D in the considered range. As such, void-filling algorithms typically outperform their counterparts without void-filling. Both D-G-VF and G-D-VF perform similarly for low values of D. An optimal granularity is not present for the void-filling algorithms because gaps do not grow along with increasing D, but rather are further mitigated by the void-filling capabilities of the scheduling algorithms. We can also see that in general for $c = 8$ the LPs are a lot smaller. This is because for unlimited wavelength conversion the burstiness of the packet arrival streams on the separate wavelengths can be dealt with in a better way as more wavelengths are present to reschedule contending packets and there is no limitation on the necessary wavelength conversion.

Results of C and C-VF with unlimited wavelength conversion are summarized in Fig. 2.4 and 2.5 and Tables 2.2 and 2.3 for $c = 4$. Fig. 2.4 shows the LP of the C scheduling algorithm for different D as a function of α. The algorithm

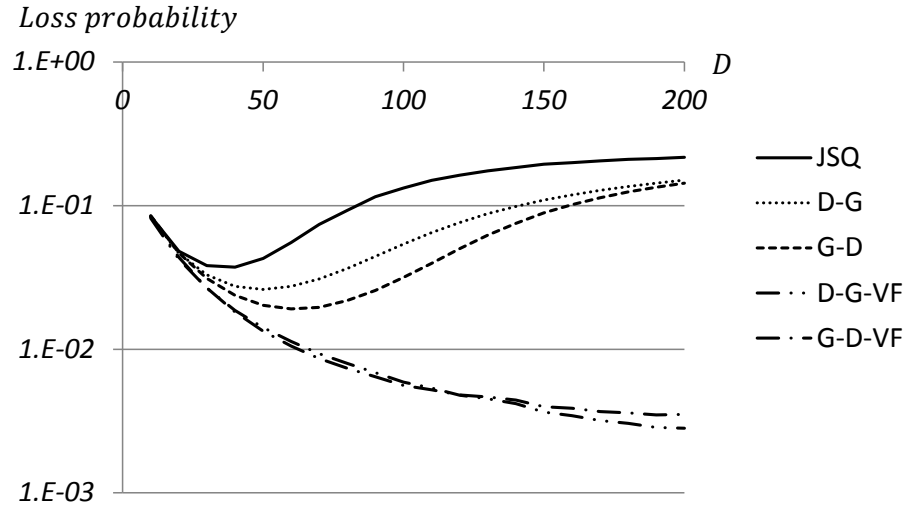


Figure 2.2: Loss probability (LP) as a function of D with unlimited wavelength conversion and $c = 4$. To improve visualisation, data points and confidence intervals are not displayed.

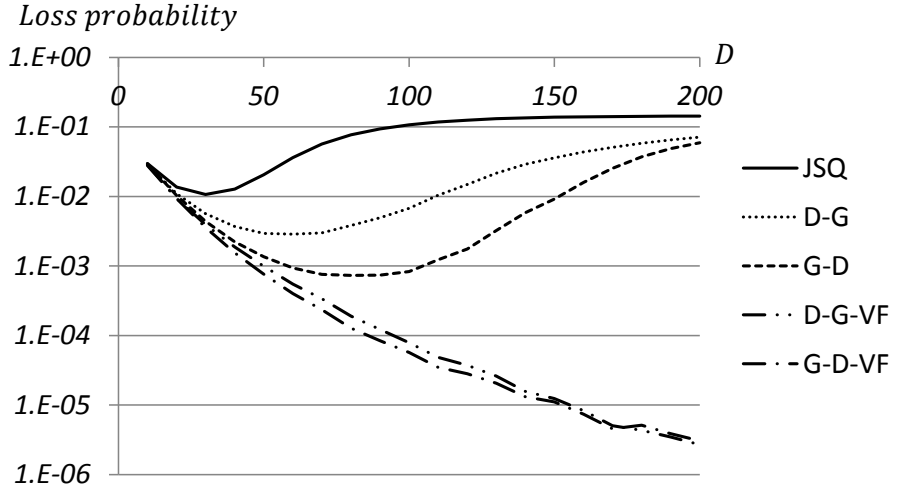


Figure 2.3: Loss probability (LP) as a function of D with unlimited wavelength conversion and $c = 8$. To improve visualisation, data points are not displayed.

parameter α is varied from 0 to 1 in steps of 0.1. For $\alpha = 0$, C coincides almost completely with D-G (and thus performs almost equally), for $\alpha = 1$, it coincides with G-D. For $0 < \alpha < 1$, C outperforms both D-G and G-D

for some intermediary interval of α. Table 2.2 shows the reduction of the LP of C relative to the LP of G-D, for different values of D, maximized with respect to α. The corresponding optimal α value is given in the second row. Similarly Fig. 2.5 and Table 2.3 are generated for C-VF. Table 2.3 calculates the reduction of LP of C-VF with respect to D-G-VF. Using G-D and D-G-VF as reference in calculating the reduction is the "fairest" choice for $c = 4$, since these are for most values of D the best performing existing non-void-filling and void-filling algorithms respectively. Tables 2.4 and 2.5 show the corresponding results for $c = 8$. For $c = 8$ the reductions of LP are calculated with respect to G-D and G-D-VF (instead of D-G-VF) as these perform best for most values of D.

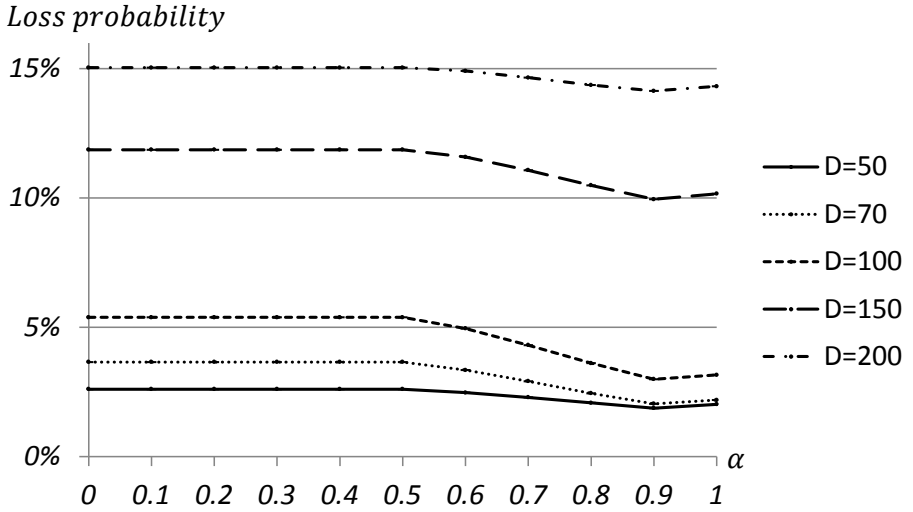


Figure 2.4: Loss probability (LP) of C as a function of the gap-delay balance (α) for different values of the granularity (D), $c = 4$ and with unlimited wavelength conversion.

On Tables 2.2 and 2.3 we can see that both algorithms achieve significant loss reduction, the exact amount, however, is highly dependent on the choice of D. The optimal value of α remains rather constant across different values of D. Averaged over the simulated values of D ($D = i \cdot 10,\ i = 1 \ldots 20$), for both C and C-VF the optimal α is 0.9. The loss probability of C is minimal for $D = 60$; for this value of D, also the largest reduction over G-D is realized (7.2 %). The LP of C-VF keeps decreasing with increasing D, similar to D-G-VF and G-D-VF, with reductions of up to 22.6 % in comparison with D-G-VF ($D = 100$). Similar results can be seen in Tables 2.4 and 2.5 for $c = 8$. The optimal α is also 0.9 for both C and C-VF but the maximum perfor-

Table 2.2: Maximum reduction of the loss probability (LP) of C relative to the loss probability (LP) of G-D for different values of the granularity (D), $c = 4$ and with unlimited wavelength conversion. The second row gives the corresponding optimal value of the gap-delay balance (α).

D	10	20	30	40	50	60	70	80	90	100
LP reduction	2.1%	4.2%	5.7%	6.7%	7.2%	7.2%	7.2%	6.5%	6.0%	5.3%
α	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

D	110	120	130	140	150	160	170	180	190	200
LP reduction	4.5%	3.8%	3.2%	2.8%	2.4%	2.1%	1.9%	1.6%	1.4%	1.2%
α	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

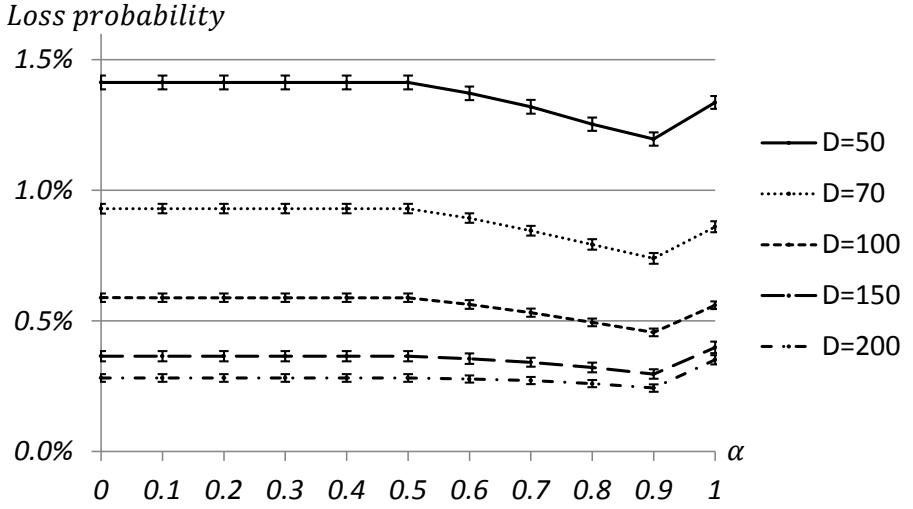


Figure 2.5: Loss probability (LP) of C-VF as a function of the gap-delay balance (α) for different values of the granularity (D), $c = 4$ and with unlimited wavelength conversion.

Table 2.3: Maximum reduction of the loss probability (LP) of C-VF relative to the loss probability (LP) of D-G-VF for different values of the granularity (D), $c = 4$ and with unlimited wavelength conversion. The second row gives the corresponding optimal value of the gap-delay balance (α).

D	10	20	30	40	50	60	70	80	90	100
LP reduction	0.2%	2.1%	6.6%	11.1%	15.4%	18.5%	20.5%	21.4%	22.6%	22.6%
α	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

D	110	120	130	140	150	160	170	180	190	200
LP reduction	22.2%	21.3%	20.9%	19.7%	18.8%	16.8%	16.4%	15.8%	14.7%	13.7%
α	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

Table 2.4: Maximum reduction of the loss probability (LP) of C relative to the loss probability (LP) of G-D for different values of the granularity (D), $c = 8$ and with unlimited wavelength conversion. The second row gives the corresponding optimal value of the gap-delay balance (α).

D	10	20	30	40	50	60	70	80	90	100
LP reduction	4.2%	8.1%	11.1%	12.5%	11.5%	11.5%	11.9%	11.2%	10.8%	12.4%
α	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

D	110	120	130	140	150	160	170	180	190	200
LP reduction	9.2%	9.6%	9.6%	10.0%	9.7%	9.0%	9.3%	8.2%	6.7%	5.9%
α	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

mance improvements are larger. For both C and C-VF the largest reduction is achieved for $D = 100$ (12.4 % and 37.7 % respectively). For C-VF results are only shown for D up to 120 as for larger values of D too many simulations, and accompanying very long simulation run time, are necessary to obtain reliable results. We expect that for larger values of D a similar trend as in Table 2.3 will be seen, i.e., a decreasing performance improvement for increasing D. Moreover this trend is already visible for $D = 110$ and $D = 120$.

Table 2.5: Maximum reduction of the loss probability (LP) of C-VF relative to the loss probability (LP) of G-D-VF for different values of the granularity (D), $c = 8$ and with unlimited wavelength conversion. The second row gives the corresponding optimal value of the gap-delay balance (α).

D	10	20	30	40	50	60
LP reduction	4.3 %	9.0 %	13.0 %	15.6 %	18.1 %	20.6 %
α	0.9	0.9	0.9	0.9	0.9	0.9

D	70	80	90	100	110	120
LP reduction	20.7 %	23.3 %	27.2 %	37.7 %	31.0 %	24.7 %
α	0.9	0.9	0.9	0.9	0.9	0.9

2.2.2.2 Limited number of wavelength converters

Tables 2.6 and 2.7 present the maximum performance improvement that C and C-VF can obtain (in terms of LP reduction) when compared with G-D and D-G-VF, respectively, for a varying number of WCs r and $D = 100$. Results for other values of D (different from 100) are not displayed but similar in nature. For each value of r, the value of α that yielded the performance improvement is indicated. The number of WCs, r, is varied from 1 to 4; this is supplemented with the case of unlimited wavelength conversion.

The case of unlimited wavelength conversion is indicated with ∞ in Table 2.6 and 2.7 although its implementation requires only $c \cdot (N+1)$ WCs. Unlimited wavelength conversion implies the presence of $c \cdot (N + 1)$ WCs since this is the maximum number of simultaneously arriving packets (i.e., during a certain time interval their arrivals all overlap) that can be stored in the FDL

buffer following the WCs. With this number of simultaneously arriving packets the entrance of each FDL is occupied for all wavelengths during the time interval their arrivals overlap. Any additional simultaneous arrivals cannot be sent to a FDL on any wavelength without overlapping with another packet, they are thus lost, even if more WCs would be present.

Inspecting Tables 2.6 and 2.7, we note that the obtained improvement of the C and C-VF algorithms in general decreases as less WCs are present. The corresponding optimal α value decreases with decreasing number of WCs for C-VF as opposed to C, for which the optimal α value stays constant when the number of WCs changes. When looking at the differences between the cases $c = 4$ and $c = 8$, we can see that a bigger improvement is possible for $c = 8$ in the case of unlimited wavelength conversion. When only a small number of WCs are present, the performance improvements that can be achieved by the C and C-VF algorithms are larger when less wavelengths are present. The optimal values of α are very similar for $c = 4$ and $c = 8$ for both a limited number of WCs and unlimited wavelength conversion.

Table 2.6: Maximum reduction in LP and corresponding α of C and C-VF relative to G-D and D-G-VF respectively for $c = 4$, $D = 100$ and a varying number of WCs r.

# wavelength converters *r*	0	1	2	3	4	∞
LP reduction C	0.0 %	2.5 %	3.5 %	4.1 %	5.2 %	5.3 %
optimal α		0.9	0.9	0.9	0.9	0.9
LP reduction C-VF	0.0 %	0.8 %	3.9 %	6.5 %	9.1 %	22.6 %
optimal α	0.7	0.8	0.8	0.8	0.8	0.9

When looking at the actual LPs of the algorithms in Table 2.8 we can see that when the number of WCs is limited the algorithms perform worse when $c = 8$. The same is observed for other values of D and the existing algorithms for which both results are not shown in Table 2.8. This is as opposed to the case of unlimited wavelength conversion, where the algorithms perform a lot better when more wavelengths are present.

When the number of WCs is severely limited ($1 \leq r \leq 4$ in Table 2.8) the rescheduling is bounded by the small number of WCs. As the same

Table 2.7: Maximum reduction in LP and corresponding α of C and C-VF relative to G-D and D-G-VF respectively for $c = 8$, $D = 100$ and a varying number of WCs r.

# wavelength converters *r*	0	1	2	3	4	∞
LP reduction C	0.0%	2.8%	2.6%	2.8%	5.2%	12.4%
optimal α		0.9	0.9	0.9	0.9	0.9
LP reduction C-VF	0.0 %	0.2%	1.9%	4.6%	7.3%	26.7%
optimal α	0.7	0.7	0.8	0.8	0.8	0.9

number of WCs are available to solve the contention on twice the number of wavelengths, this can be done less effectively and the performance is worse. Note that for $c = 8$ and $c = 4$ the same load ($\rho = 0.8$) and the same average packet length ($E[B] = 100$) are used. The average inter-arrival time $E[T]$ for $c = 8$ thus is half of that for $c = 4$. The average inter-arrival time on the separate wavelengths $c \cdot E[T]$ however is the same. Overall twice the number of packets arrives thus in the same time for $c = 8$ resulting in a lot more requests per time unit to use the small numbers of WCs. As more WCs are added, the WCs become less and less a limiting factor and the number of wavelengths c determines how good contention is resolved and thus the number of packets lost. For a certain value of r (larger than 4) and above the C and C-VF algorithm will perform better for $c = 8$. While there is no simulation data for $4 < r < c \cdot (N + 1)$, looking at the values in Table 2.8 one can suspect that this inversion will happen for C-VF for a value of r only slightly larger than 4. For the C algorithm this is likely to happen for a larger value of r.

2.2.3 Methodology

As mentioned in Section 1.9 the performance of the scheduling algorithms is evaluated by means of Monte Carlo simulation and discrete event simulation (DES). We now go into more detail on the exact implementation of all variables, system components and their interactions.

For both the setting of unlimited wavelength conversion and a limited number of wavelength converters, the first part of the system state includes the

Table 2.8: Optimal LPs of C and C-VF for $c = 4$ and $c = 8$ with $D = 100$ and a varying number of WCs r.

# wavelength converters *r*	0	1	2	3	4	∞
LP C, optimal α, c=4	18.14%	12.20%	8.37%	5.80%	4.21%	2.99%
LP C, optimal α, c=8	18.19%	13.56%	10.36%	7.56%	5.11%	0.07%
LP C-VF, optimal α, c=4	9.02%	4.69%	2.75%	1.70%	1.10%	0.46%
LP C-VF, optimal α, c=8	9.12%	5.56%	3.40%	1.99%	1.11%	0.004%

description of the provisional schedule. However, when the number of WCs is limited, a second additional part of the system state includes information about the occupation of the WCs. Both parts are discussed in more detail below. As an illustration the Matlab-like based pseudocode of two algorithms is added in Appendix A and B. Appendix A contains the pseudocode of the non-void-filling C algorithm for a setting of unlimited wavelength conversion. Appendix B contains the pseudocode of the void-filling CWC-VF algorithm for the setting of a limited number of wavelength converters. CWC-VF is an algorithm discussed in more detail in Chapter 3. By adding these two specific appendices not only the difference between the non-void-filling and void-filling algorithms can be illustrated but also between the setting of unlimited wavelength conversion and a limited number of wavelength converters and between the C and C_W costs (see Chapter 3 for more details on the C_W cost and its implementation).

2.2.3.1 First part of system state

A first part of the system state, concerning the provisional schedule, differs for non-void filling algorithms and void-filling algorithms.

For the **non-void-filling algorithms** the simulation keeps track of the horizon of each channel. A first part of the system state is thus a c-dimensional vector of scalar real values (*horizons* in Appendix A), and is the first part of the Markov state of a Markov chain, evolving as follows. The events list consists of the arrivals of packets. In each iteration the arrival of a packet is simulated, i.e., the inter-arrival time (T_{packet}) (the time between the current packet and the next), the packet length of the current packet (B_{packet}) and the wavelength w ($w = 1 \ldots c$) on which it arrives, are generated using a

random number generator. The packet is lost if no feasible channel exists, or the channel with wavelength w is not feasible and no WC is available. Else the correct channel is chosen accordant with the used algorithm by assigning a cost to each outgoing channel.

For the C algorithm this cost takes into account both the gap and the delay in varying degrees as α, the gap-delay balance, is changed in different simulation runs. The channels that are not feasible and all channels with a different wavelength than w in case no WC is available are given a cost of ∞. Next the channel with the lowest cost is chosen as the scheduling wavelength (with preference for the channel with wavelength w in case of a tie). The horizon of the scheduling wavelength is updated to a new value:

$$horizon_{scheduling\ wavelength} = delay_{scheduling\ wavelength} + B_{packet}\ .$$

The horizons of the other channels are not changed in this step. Next, the horizons of all channels are updated by subtracting the inter-arrival time and taking the maximum of this value and zero as the new horizons.

$$horizon_{all\ wavelengths} = max(horizon_{all\ wavelengths} - T_{packet}, 0)\ .$$

This process is repeated iteratively from arrival to arrival, for the predetermined (large) number of packets.

Simulations of **void-filling algorithms** are more complex than their non-void-filling counterparts. Yet the events list still consists only of the simulated arrivals (inter-arrival time, packet length and arriving wavelength). The algorithms are implemented as described in [128]. The simulation keeps track of onsets and offsets of all voids on each channel in the three-dimensional void matrix V, representing the first part of the Markov state of the involved Markov chain. The dimensions of V are $(c,\ L_{max},\ 2)$; here, L_{max} is the maximum of $\{L_i,\ i = 1 \ldots c\}$ with L_i the number of non-dummy voids on channel i. For a random point (i, l, o) in matrix V, the first dimension $(i = 1 \ldots c)$ denotes the wavelength of the void and the second dimension $(l = 1 \ldots L_{max})$ indicates the sequence number of the void on this channel. The third dimension is set to one $(o = 1)$ if the value gives the beginning of a void and is set to two $(o = 2)$ for the end of a void. If a channel i is idle, a single void with $V(i,\ 1,\ 1) = 0$ and $V(i,\ 1,\ 2) = \infty$ is stored in V. Because the system starts completely idle, V is initialised with a single void $(0,\ \infty)$ for each wavelength.

After the arrival of a packet is simulated, the cost matrix is initialised as a matrix of size $c \cdot (N+1)$ with all elements equal to ∞. Each combination of a wavelength (row) and an FDL (column) corresponds to a position in this matrix. Next, knowing how much and which WCs are available for each void, it is checked whether it overlaps an FDL at least an amount equal to the packet length to the right of this FDL. If so the resulting cost, for some algorithms taking into account the number of available WCs, is calculated for this position in the cost matrix. After all costs have been calculated the chosen SP (i.e., the scheduling wavelength and the scheduling FDL) is the one with the lowest cost. Again preferably the channel with wavelength w is chosen to not needlessly use the WCs. If the lowest value in the cost matrix equals ∞, no eligible SP exists and the packet is lost. Next, if the packet is not lost, V is updated by splitting the void that is overlapping the chosen position in one void in front of the newly scheduled packet and one behind it. The number of voids in V thus increases by one.

When updating V we however have to take into account that for each channel the last non-dummy void ($l = L_i$) always ends at ∞ ($V(i,\, L_i,\, 2) = \infty$) and begins before ∞ ($V(i,\, L_i,\, 1) < \infty$). The values of V ($i,\, l = L_{i+1} \ldots L_{max},\, o = 1 \ldots 2$) for which $L_i < L_{max}$ have to be set to ∞, so creating dummy voids with both ending and beginning equal to ∞. This is achieved by increasing the size of V only when the scheduling wavelength has no dummy voids. When the scheduling wavelength has at least one dummy void the extra void created by scheduling a packet will be nullified by deleting a dummy void. Subsequently the simulated inter-arrival time is subtracted from all values in V to take into account progression of time. All voids in the updated V that start and end at time equal to zero are removed. The next arrival is now simulated, which again is repeated iteratively.

The void-filling algorithms have to check all voids on all wavelengths to calculate the cost matrix, whereas the non-void-filling algorithms have to check each wavelength only once to calculate the cost. This implies that the former ones are computationally more complex and more difficult to implement in a switch. Within both classes of algorithms, however, no significant difference exists in complexity of implementation. Because existing algorithms also use gap and delay to choose among the SPs, a simple additional calculation of a cost based on the same gap and delay does not slow down the algorithm. This is important: any improvement achieved within a class of algorithms is achieved without complicating implementation.

2.2.3.2 Second part of system state

The second part of the system state describes the availability of the WCs. Since possible wavelength conversion takes place before the packets enter the FDLs we only need to know whether a WC is vacant right now, i.e., upon arrival of the packet. Similar to the model of the non-void-filling algorithms, we thus keep track of a horizon for each WC, representing the latest time the WC is currently scheduled to be in use. Interwoven with the calculation of the correct SP, the availability of WC is checked in the algorithm. To enable a packet's wavelength to be changed, the horizon of one or more WCs has to be equal to zero. Next the chosen WC's horizon is then altered to the length of the packet. As a last, the inter-arrival time is subtracted from all horizon values, after which the maximum of this value and zero is chosen as the new horizon.

2.3 Conclusions

In this chapter we introduced the C and C-VF scheduling algorithms for optical switching in OPS/OBS networks in settings of both a limited number of wavelength converters and unlimited wavelength conversion. As opposed to existing scheduling algorithms as JSQ and others, the newly proposed algorithms use a parametric cost-based approach to choose among the potential scheduling points. This cost function assigns a cost C to all scheduling points which equals a weighted average of the scheduling criteria delay $(1-\alpha)$ and gap (α) associated with the scheduling point. By varying the weighing factors (α and $1-\alpha$) of this cost function the performance in terms of loss probability was optimized for a variety of settings. This was done using Monte Carlo discrete event simulation (DES).

It was shown that for the setting of unlimited wavelength conversion performance could be improved by up to 38 % for the C-VF algorithm for 8 wavelengths, the granularity equal to 100 and the load equal to 0.8. For the C algorithm or when the number of wavelength converters is limited, the obtainable improvement decreases. Overall the optimal α value hovers around 0.90 for both settings and different parameter combinations. The only minor exception to this rule is for C-VF in the setting of a limited number of WCs, for which the optimal α decreases with a decreasing number of WCs.

When the number of WCs is strictly limited both existing and the C and C-VF algorithms perform (slightly) worse when more wavelengths are present

for an equal overall load. This is as opposed to the setting of unlimited wavelength conversion, in which more wavelengths result in a higher multiplexing gain and thus highly (multiple orders of magnitude) improved performance. For a limited number of WCs, the rescheduling is bounded by the number of WCs which have to be shared among a higher number of requests per time unit to change the wavelength. As more WCs are added, the WCs become less and less a limiting factor and the number of wavelengths determines how good contention is resolved. There is thus an intermediate value of WCs for which a higher number of wavelengths becomes beneficial. The value for which this inversion occurs is higher for non-void-filling algorithms.

Chapter 3

Cost-based algorithms: wavelength converter cost

Louis Pasteur's theory of germs is ridiculous fiction.
Pierre Pachet, Professor of Physiology at Toulouse, 1872

In this chapter we continue on the path of parametric cost-based scheduling algorithms we introduced in Chapter 2. In order to further improve performance and reduce the energy consumption of the wavelength converters we extend the cost function with a term that takes into account the use of the wavelength converters.

3.1 Energy consumption of wavelength converters

Given that some studies indicate that 4 % of the primary energy worldwide is consumed by the IT sector, the evaluation of and reduction in energy consumption of IT applications has been a popular subject of academic research for more than a decade [129]. In this, the main drivers for so called "green networking" are not only economical and environmental but also technical (reduced heat dissipation). Moreover, if IT equipment and applications consume less energy, it further strengthens the case for using IT-focused solutions to achieve cost (and energy) savings in other sectors and domains.

Focusing on backbone telecommunication networks we can say that, in general, using all-optical wavelength conversion technology results in less energy

consumption than using (semi-)electronic switches with equal switching capability, see, e.g., [130]. This said, the energy consumption of optical switches remains substantial. It is therefore desirable that the energy efficiency of all switch elements is focused on explicitly.

WCs enable high performance switching but are known to contribute significantly to the energy consumption of the switching element. As pointed out in [128], switching the WCs off when they are not used already enables a reduction of the overall power consumption. However, one may also design with the aim of reducing the usage of the WCs, taking into account the trade-off between performance and energy consumption explicitly. This is the aim of the current work, in which scheduling is preferably done without wavelength conversion, unless in the cases where conversion yields significant performance improvement, justifying the (momentary) increase in energy consumption by an increase in performance. Since the number of both wavelengths and FDLs in a practical implementation is limited, the major concern in scheduling indeed remains the reduction of packet loss using easily implementable scheduling algorithms.

For the specific problem, to the best of the authors' knowledge, we are the first to consider performance and energy consumption of wavelength conversion in a unified manner. A parametric, cost-based approach to this problem is presented in [84, 87] for the setting of unlimited wavelength conversion and in [85, 88] for a limited number of wavelength converters. Both lead to algorithms which are of the same implementation complexity as existing cost-based algorithms and can easily be implemented in practice, motivating the usefulness of our results for optical switch design.

3.2 Assumptions

For both the case of unlimited wavelength conversion and a limited number of wavelength converters similar assumptions as in Section 2.1.1 are made. The overall framework is a dedicated conversion before buffering setting corresponding to the setting of Fig. 1.8 and 1.9. This setting assumes a $K \times M$ optical switch configuration. Packets arrive on a finite number of incoming ports K, on c different wavelengths $\lambda_1, \ldots, \lambda_c$, also called channels. Each packet arrives on a certain wavelength and is switched (still on this wavelength) to one of the M output ports according to the packet header destination information. Each output port thus accepts packets from K ports, on c wavelengths. The output port is connected to a single fiber with the

same c different wavelengths and we assume at each of the M output ports the joint packet arrival process is a Poisson process with the wavelength w of each arriving packet randomly and uniformly distributed among the c wavelengths. Again, an arbitrary single output port is analyzed. The length of arriving packets, B, is assumed exponentially distributed with an average of $E[B]$ time units. With the assumption of a Poisson process for the arrival process at the output port, the inter-arrival T time is thus exponentially distributed with an average $E[T]$. Given the nature of the Poisson process and the random and uniform distribution of the wavelength w among the c possibilities the inter-arrival times on each wavelength are also exponentially distributed but with an average of $c \cdot E[T]$. Related, the overall incoming traffic load at the output port is given by $\rho = E[B]/(c \cdot E[T])$.

We assume each output port has the same single stage and feedforward optical buffer in which $N + 1$ FDLs are available to schedule incoming packets. The lengths of the FDLs are consecutive multiples of a basic value D called the granularity. As mentioned this is called a degenerate delay buffer, in which incoming packets sent through the j-th $(j = 0 \ldots N)$ delay line encounter a delay of $j \cdot D$. Besides an FDL buffer wavelength converters are available at the output port to change the wavelength of an arriving packet if demanded so by the scheduling algorithm. Either full wavelength conversion capability (Section 3.3) or a limited but dedicated set of r wavelength converters (Section 3.4) is assumed. With full wavelength conversion capability, whenever a packet is scheduled on a different wavelength than the incoming wavelength, wavelength conversion is possible. This is as opposed to the case when only a limited set of wavelength converters are available, in which case at least one of the r WCs has to be available before scheduling a packet on another wavelength. Similar to Section 2.1.1 we assume the WCs can convert any of the c incoming wavelengths to all other wavelengths of the set of c wavelengths on a packet to packet basis (TWC).

Given that the assumptions and boundary conditions for the new algorithms are the same, a similar visual representation, i.e., the provisional schedule, can be used to display the already scheduled packets upon arrival of an arbitrary packet that is yet to be scheduled. On this provisional schedule, each intersection between a horizontal line and a vertical line is marked with a dot and represents a *potential scheduling point* (PSP) for the new packet. Within this set of points, a *scheduling point* (SP) is one allowing the packet to be scheduled with the used algorithm and the vacant WCs (if any), without overlapping with any of the packets already present in the system. Whether or not a given combination of wavelength and FDL is a

PSP thus depends on the system's provisional schedule. Whether a PSP is an SP depends upon both the size of the packet to be scheduled, the used scheduling algorithm and whether a WC is vacant. If no SPs are available, the current packet is lost, otherwise the scheduling algorithm decides which SP the packet is assigned to. Upon this set of SPs the algorithm applies the *scheduling criteria* (SCs) to choose an SP. Scheduling criteria are properties regarding the provisional schedule, the incoming wavelength and the number of available WCs. In this the big difference between the algorithms discussed below and those from Chapter 2 is that the latter only take into account the number of WCs indirectly, i.e., they only schedule on the same wavelength if no WC is available. The algorithms below take into account the use of a WC explicitly, possibly scheduling a packet on the same wavelength it arrived on to avoid wavelength conversion. Table 3.1 extends the overview of Table 2.1 and shows an overview of all scheduling criteria (SCs) of the algorithms of Chapter 2 and the ones discussed below. Similar to Chapter 2 all algorithms are given an abbreviated name in accordance with the used logic of the concerned algorithm.

The number of FDLs $(N+1)$ is fixed at 10 and the load ρ is fixed at 80 %. The number of available channels c is assumed to equal 4 and 8 in two different sets of simulations. As the LP is highly dependent on the granularity (D), the granularity is therefore varied from 10 to 200 time units with steps of 10. The average packet length $(E[B])$ is assumed 100 time units. For the non-void-filling algorithms the arrival of 10^7 packets is simulated 10 times. For the void-filling algorithms the arrival of 10^6 packets is simulated 10 times to keep run-time acceptable. Both values result in narrow confidence intervals for each setting.

3.3 Unlimited wavelength conversion

As mentioned before, scheduling is done according to a scheduling algorithm, which either aims purely for minimal loss (LP), or also pursues reduced energy consumption of the WCs. As in the unlimited wavelength conversion case, the wavelength converters are not a scarce resource, reducing the WCs' use only affects their energy consumption and does not allow to improve the LP performance. In this section we will thus focus on trading off the performance improvement of the cost-based algorithms from Chapter 2 for a significant reduction in up-time of the wavelength converters by introducing a conversion cost in the involved cost function.

Table 3.1: Overview of the considered scheduling algorithms of this chapter and Chapter 2 and their scheduling criteria (SCs).

algorithm	scheduling criterium (SC)	scheduling criterium (SC) in case of a tie		
		1st order tie	2nd order tie	3rd order tie
JSQ	horizon	wavelength	random	
D-G	delay	gap	wavelength	random
G-D	gap	delay	wavelength	random
D-G-VF	delay	gap	wavelength	random
G-D-VF	gap	delay	wavelength	random
C	cost C	wavelength	random	
C-VF	cost C	wavelength	random	
CW	cost C_w	wavelength	random	
CW-VF	cost C_w	wavelength	random	
CW (A, B, C, D)	cost $C_{w,\,(A,\,B,\,C,\,D)}$	wavelength	random	
CW-VF (A, B, C, D)	cost $C_{w,\,(A,\,B,\,C,\,D)}$	wavelength	random	

3.3.1 Approach and concept

In Section 2.2.1 we developed the C and C-VF algorithms based on the cost C which equals a weighted average of the SCs delay and gap associated with the SP:

$$C = \alpha \cdot gap + (1 - \alpha) \cdot delay \ . \tag{3.1}$$

Here α is an algorithm parameter called the *gap-delay balance*, real-valued in the interval $[0, 1]$. The weighing coefficients α and $1 - \alpha$ were varied to maximize performance in different settings but always sum up to one. Based on the cost C we now introduce the altered cost function C_W that by means of an extra term incorporates (and reduces) the usage of the wavelength converters:

$$C_W = \alpha \cdot gap + (1 - \alpha) \cdot delay + \beta \cdot D \cdot \left[1 - \delta_{wi}\right] \ . \tag{3.2}$$

Here, w is the packet's original wavelength and β is a second, and newly introduced, algorithm parameter called the *conversion cost parameter*, which (like α) is a real-valued parameter, however, not restricted to the interval $[0, 1]$, but always positive ($\beta > 0$). Furthermore, δ_{wi} denotes Kronecker's delta (1 if $w = i$ and 0 otherwise.) All SPs on an outgoing channel (i) different from the packet's incoming wavelength (w) are thus penalized with an extra cost. For appropriate scaling of parameters, this cost is the product of a weighing factor involving β and the granularity D. The granularity is introduced because both the gap and delay term in the cost function scale along with D. In this way values of the algorithm parameters are kept in the same order of magnitude, facilitating insight after parameter optimization.

Based on this C_W cost we propose a non-void-filling and void-filling scheduling algorithm called respectively CW (Cost and Wavelength) and CW-VF. The SCs for the CW and CW-VF algorithms are the delays, the gaps and the wavelengths of the SPs (all reflected in the cost C_W). Ties are broken arbitrarily.

3.3.2 Performance results

As we will focus on the trade off between performance and energy consumption for the setting of unlimited wavelength conversion we need, besides LP as our measure of performance, a measure to keep track of the energy consumption of the wavelength converters. To do this we monitor the packets' payload (packet length) rather than the number of converted packets, since conversion time is proportional to the length of the packet. The ratio of the payload of converted packets to the payload of all packets that are not lost, is therefore our energy consumption measure of the wavelength converters. Although this is not a direct measure of energy consumption it obviously relates to it. Since we assume wavelength converters are only switched on when actually used [130], energy consumption is proportional to the payload of the converted packets.

3.3.2.1 Algorithms of Chapter 2

In Fig. 3.1 and 3.2 our energy consumption measure is displayed for $c = 4$ and $c = 8$ as a function of D for the algorithms analyzed in Chapter 2. As it was shown that, for the case of unlimited wavelength conversion, typically $\alpha = 0.9$ is the best choice for both C and C-VF, the displayed graphs of C and C-VF assume this value of α. From Fig. 3.1 and 3.2 we can see that for all algorithms the percentage of converted packets' payload increases with

D. For a tie between scheduling with and without conversion to occur, the presence of at least two idle channels is required. This is due to the continuous nature of the arrival process. Increasing D, the time necessary to clear a channel completely increases, which decreases the chance, or probability, of finding at least two idle channels. In the limit for D going to infinity, $\frac{c-1}{c}$ of all payload is converted to a different channel since no ties occur, and with a chance of 1 in c the algorithm schedules without conversion.

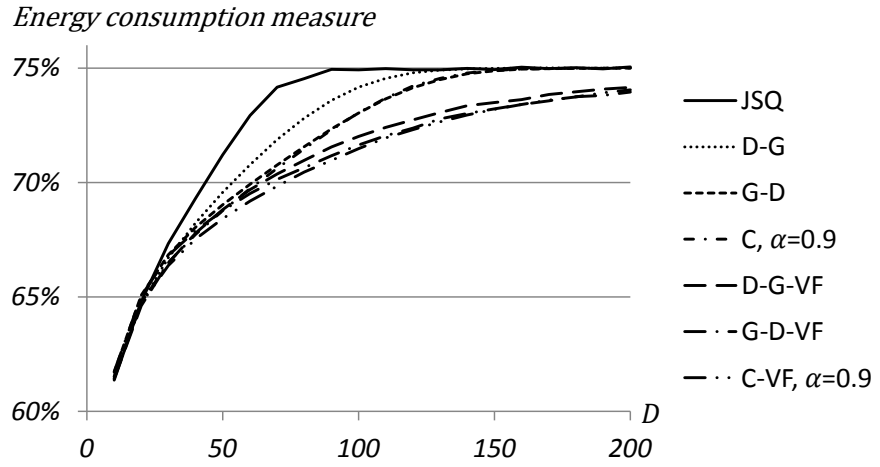


Figure 3.1: Ratio of payload of converted packets to the payload of all packets as a function of the granularity (D) for all algorithms analyzed so far for $c = 4$. To improve visualisation, data points are not displayed.

One can also optimise the gap-delay balance α of C and C-VF for energy consumption and analyze performance under these conditions. Fig. 3.3 and 3.4 show the energy consumption measure for $c = 4$ of C and C-VF respectively as a function of the gap-delay balance α. Although the maximum improvements in energy consumption that can be achieved by C and C-VF are always very limited (<1 %), the values of α for which the lowest energy consumptions are achieved, are always close to those for optimal performance. For $D \geq 100$ G-D ($\alpha = 1$) has the lowest energy consumption of all non-void-filling algorithms and for $D \geq 190$ G-D-VF ($\alpha = 1$) has the lowest energy consumption of the void-filling algorithms. For $c = 8$ no graphs are displayed but the results are similar. Again the α values for which the lowest energy consumptions are achieved are close to those for optimal performance and the achievable improvements are very small (<1 %). For $D \geq 50$ and $D \geq 80$ respectively G-D and G-D-VF have the lowest energy consumption ($\alpha = 1$).

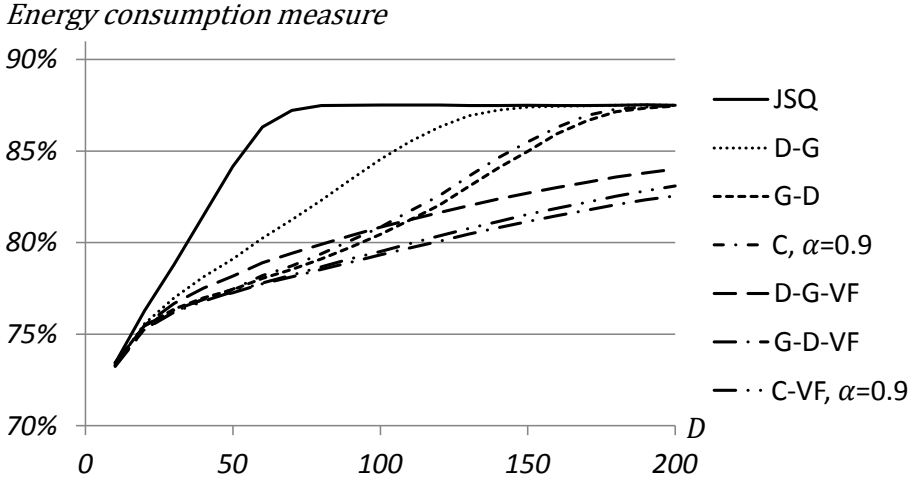


Figure 3.2: Ratio of payload of converted packets to the payload of all packets as a function of the granularity (D) for all algorithms analyzed so far for $c = 8$. To improve visualisation, data points are not displayed.

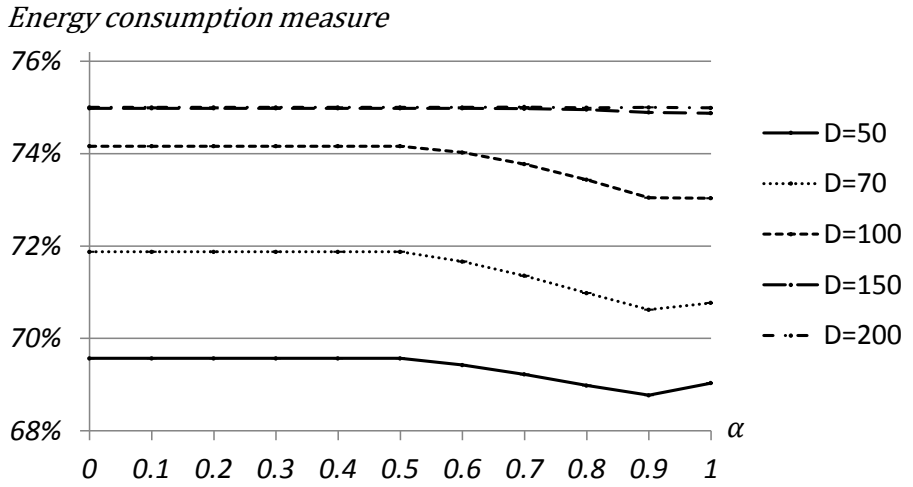


Figure 3.3: Energy consumption of C as a function of the gap-delay balance (α) for different values of the granularity (D) and $c = 4$.

3.3.2.2 CW and CW-VF

To now evaluate the performance and energy consumption of CW and CW-VF, α is fixed to 0.9, which is on average the optimal value of α for both C and C-VF and for both $c = 4$ and $c = 8$. With α fixed we can now change β and investigate the effect. Here, β comes about as an energy reduction parameter: increasing the value of β, the contribution of the energy cost of using a wavelength conversion in the total cost increases proportionally. As a result, fewer wavelength conversions are made, at the price of (slight) reduction in loss performance.

Fig. 3.5 compares the energy consumption reduction and performance decrease (loss probability increase) of CW with respect to C (both with $\alpha = 0.9$) for $c = 4$. As can be seen, reducing energy consumption by a limited amount, e.g., 10 %, results in less-than-proportional decrease in performance, with 1 - 4 %, for $D = 50$, 100 and 150. For energy consumption reduction of up to 20 %, the performance decrease is still proportionally smaller. As such, results suggest that the region of 0 - 20 % energy consumption reduction yields improved energy efficiency (i.e., more energy consumption reduction than performance decrease) with the given method. For larger reductions, a different approach is needed. Fig. 3.6 shows complementary results, with CW-VF compared to C-VF also for $c = 4$. The interpretation is largely the same as for Fig. 3.5: 0 - 20 % (or even 0 - 30 %) energy reduction allows for improved energy efficiency (i.e., more energy consumption reduction than performance decrease). For the case of $c = 8$ however, Fig. 3.7 clearly shows that, with exception for low values of D, the performance decrease is always a lot higher than the energy consumption reduction for CW. Although too much simulations are necessary to obtain reliable results for CW-VF in the case $c = 8$, similar results as for CW are expected: with exception for low values of D, a much higher performance decrease than energy consumption reduction.

Finally, merging results of Chapter 2, Fig. 3.5 and 3.6, we can also determine the value of β for which G-D and CW, and D-G-VF and CW-VF, respectively perform equally (for $c = 4$). These values are shown in Fig. 3.8 for CW and in Fig. 3.9 for CW-VF. Since these values of β are obtained by linear interpolation between actually simulated values of β (β was varied in steps of 0.1), actual simulated β values are also shown in these figures. The actual simulated β values displayed, are those simulated values of β for which the algorithm (CW or CW-VF) performs at least as good as its benchmark algorithm (G-D or D-G-VF) and, the energy consumption is minimal. Because β was varied in steps of 0.1 and energy consumption rises monotonically with increasing β, actual simulated β values are the interpolated values of β rounded down to

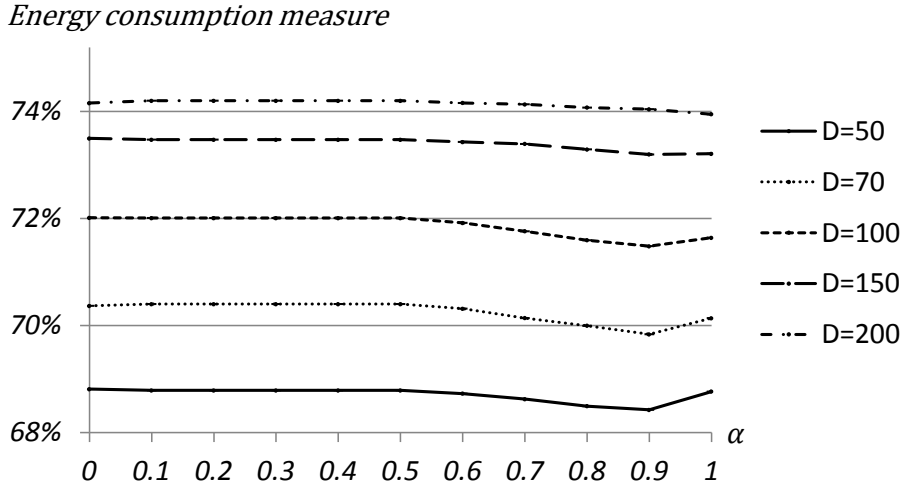


Figure 3.4: Energy consumption of C-VF as a function of the gap-delay balance (α) for different values of the granularity (D) and $c = 4$.

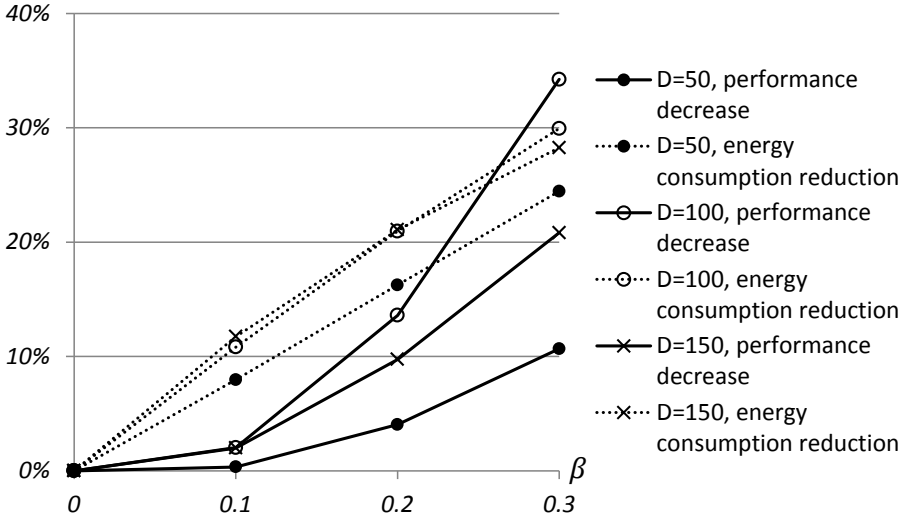


Figure 3.5: Energy consumption and performance for CW ($\alpha = 0.9$) with respect to C ($\alpha = 0.9$) as a function of the conversion cost parameter (β) with granularity $D \in \{50, 100, 150\}$) and $c = 4$.

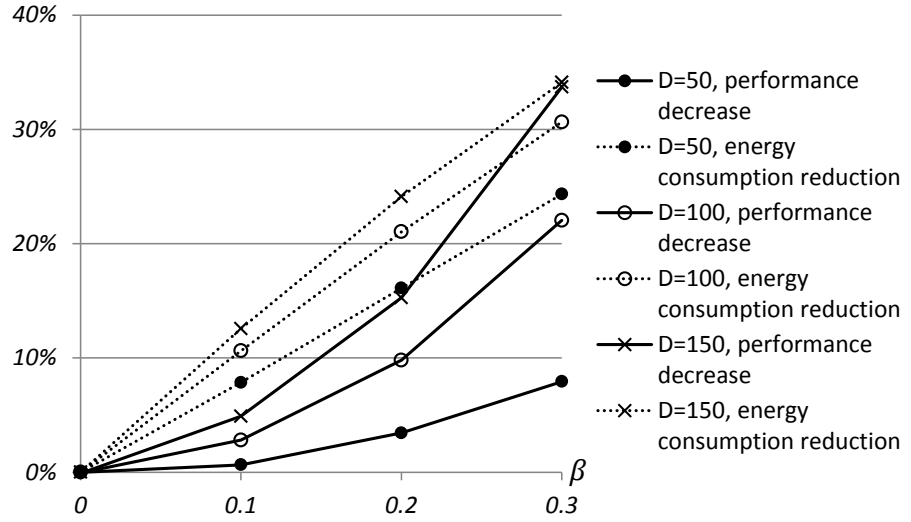


Figure 3.6: Energy consumption and performance for CW-VF ($\alpha = 0.9$) with respect to C-VF ($\alpha = 0.9$) as a function of the conversion cost parameter (β) with granularity $D \in \{50, 100, 150\}$) and $c = 4$.

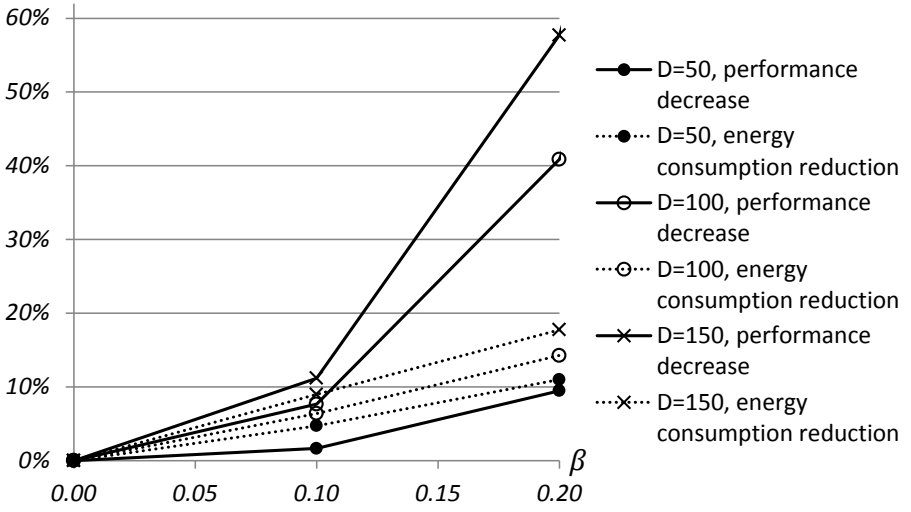


Figure 3.7: Energy consumption and performance for CW ($\alpha = 0.9$) with respect to C ($\alpha = 0.9$) as a function of the conversion cost parameter (β) with granularity $D \in \{50, 100, 150\}$) and $c = 8$.

a multiple of 0.1. Notice that the data point of CW-VF for $D = 10$ is missing. This is because for $\alpha = 0.9$ and $D = 10$ C-VF does not improve performance compared to D-G-VF and thus the trade-off between performance and energy consumption is absent for CW-VF. The corresponding energy consumption reduction of CW and CW-VF with respect to G-D and D-G-VF respectively is shown in Fig. 3.10 and Fig. 3.11 for both the interpolated values of β and the actual simulated values of Fig. 3.8 and Fig. 3.10 respectively (matching line styles with Fig. 3.8 and Fig. 3.9). It turns out that CW provides up to 28 % energy reduction over G-D for matching LP ($\alpha = 0.9$, $\beta = 0.54$ and $D = 20$) and CW-VF up to 37 % when compared to D-G-VF ($\alpha = 0.9$, $\beta = 0.41$ and $D = 70$). In those cases, energy efficiency is thus improved significantly while the performance of existing benchmark algorithms is matched. In Fig. 3.12 and 3.13 the corresponding results for CW in the case $c = 8$ are shown. We can see that although the performance decrease is almost always a lot higher than the energy reduction for CW (Fig. 3.7), similar energy consumption reductions for CW are obtained when performance is equal to that of G-D as in the case for $c = 4$. This is because the worse performance-energy trade-off of CW is compensated by the bigger performance improvement obtained by the C algorithm with $\alpha = 0.9$ for $c = 8$ (Table 2.4). Moreover the corresponding values of β for which this equal performance is obtained are also similar to those of the case of $c = 4$. Results for CW-VF when $c = 8$ are not shown as a too long simulation time is necessary to obtain reliable results. Similar results are expected though: slightly lower energy consumption reductions for equal performance than in the case of $c = 4$ due to the higher performance decrease (partly compensated by the better performance of C-VF).

To optimise CW and CW-VF for energy consumption, β is chosen large enough to schedule all packets on their arriving wavelength. By doing so the WCs are not used at all and the energy consumption measure is zero. Performance of CW is then equal to the performance of JSQ when no WCs are present and performance of CW-VF equals performance of C-VF with the same value of α when no WCs are present. These values of the LP are not shown but are up to 30 times larger than the corresponding optimal performance values.

3.4 Limited number of wavelength converters

In Section 3.3, the main aim was to validate the usefulness of a cost-based approach, with results obtained under the assumption of unlimited wavelength

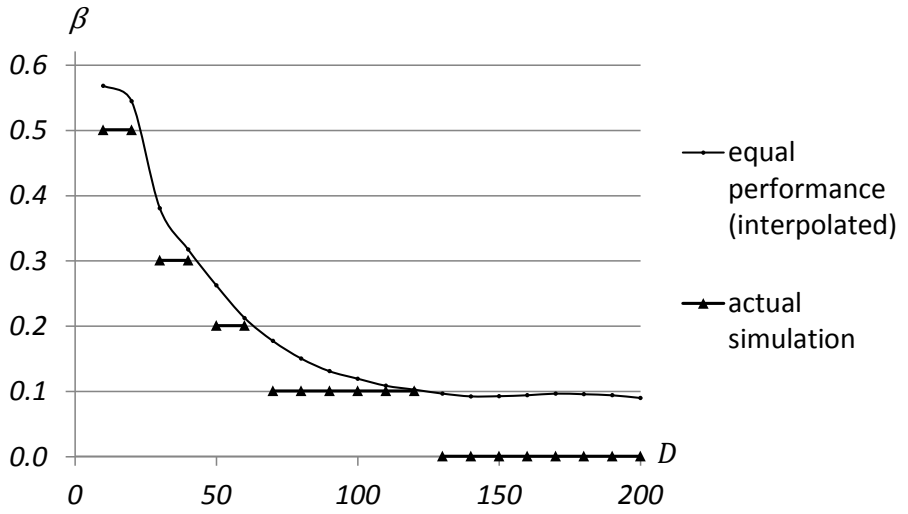


Figure 3.8: Interpolated values and actually simulated values of β for which CW ($\alpha = 0.9$) performs (at least) as good as G-D, as a function of the granularity D and for $c = 4$.

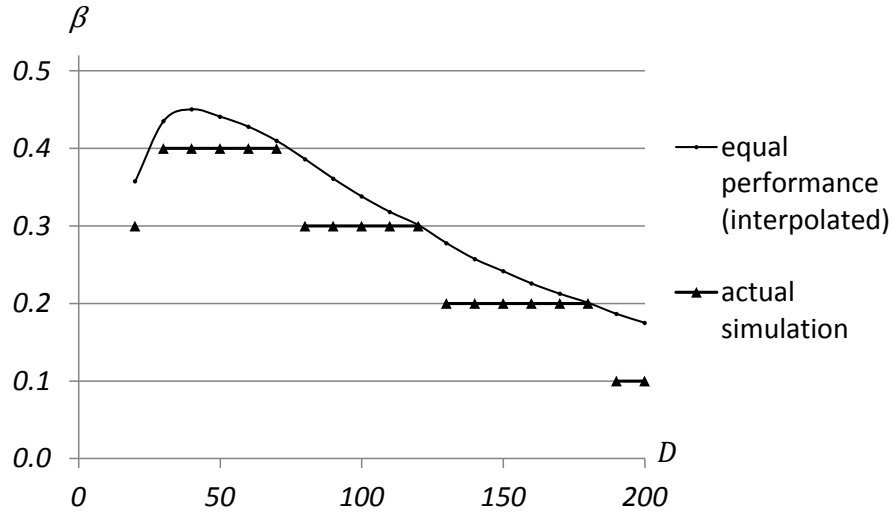


Figure 3.9: Interpolated values and actually simulated values of β for which CW-VF ($\alpha = 0.9$) performs (at least) as good as D-G-VF, as a function of the granularity D and for $c = 4$.

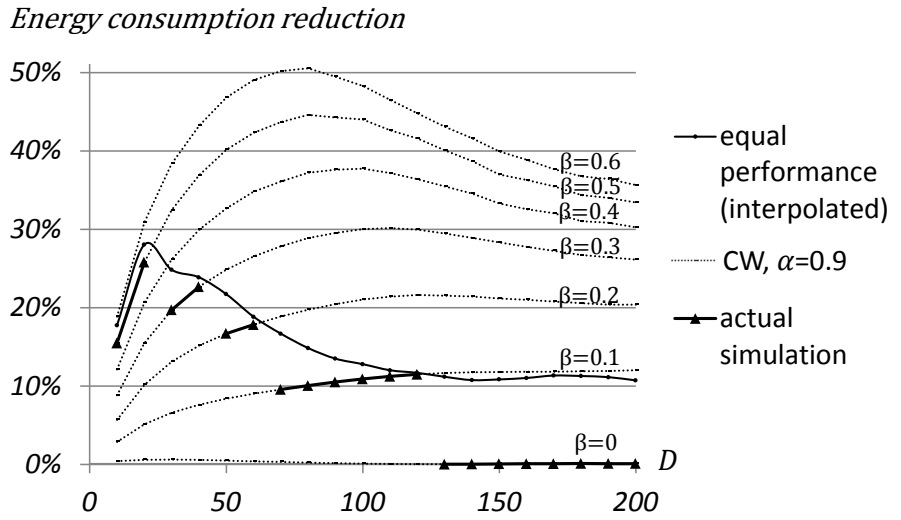


Figure 3.10: Energy consumption reduction for CW ($\alpha = 0.9$ and β values of Fig. 3.6) when compared to G-D, as a function of the granularity D and for $c = 4$.

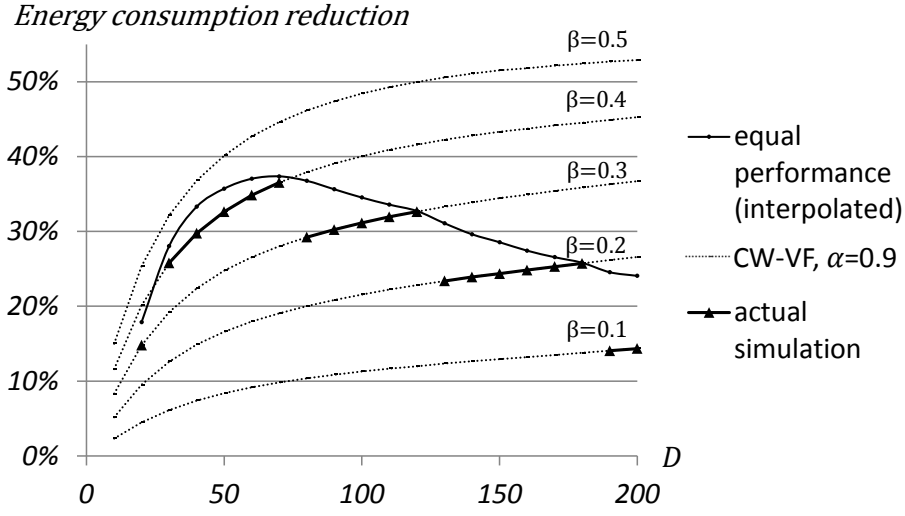


Figure 3.11: Energy consumption reduction for CW-VF ($\alpha = 0.9$ and β values of Fig. 3.7) when compared to D-G-VF, as a function of the granularity D and for $c = 4$.

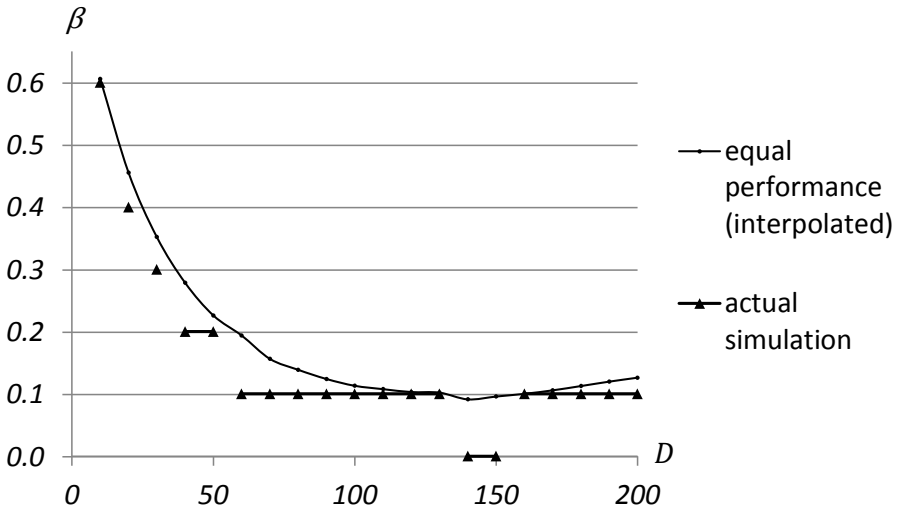


Figure 3.12: Interpolated values and actually simulated values of β for which CW ($\alpha = 0.9$) performs (at least) as good as G-D, as a function of the granularity D and for $c = 8$.

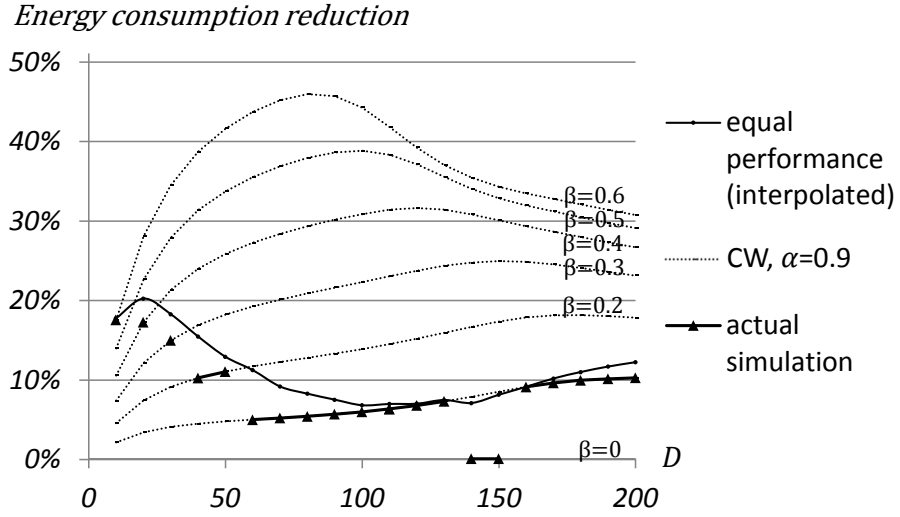


Figure 3.13: Energy consumption reduction for CW ($\alpha = 0.9$ and β values of Fig. 3.10) when compared to G-D, as a function of the granularity D and for $c = 8$.

conversion capability. There, this reduction comes at the cost of a decrease in the performance of the algorithms (increase in LP). Here, we consider the case of a limited number of shared WCs, as it occurs in practical implementations. As shown below, the scarcity of this resource results in substantially different behaviour for the underlying system, and consequently, in different requirements for the involved scheduling algorithms, motivating a dedicated performance analysis. Here, it is important to note that the new scheduling algorithms presented below prevent indiscriminate use of a scarce resource (WCs). Aiming purely for minimal loss may thus result in reduced energy consumption, as is the case in several instances studied (see below).

3.4.1 Approach and concept

Based on the cost C_W we propose four new cost functions: $C_{W,A}$, $C_{W,B}$, $C_{W,C}$ and $C_{W,D}$, resulting in eight new cost-based algorithms for a setting with a limited number of WCs: CWA, CWA-VF, CWB, CWB-VF, CWC, CWC-VF, CWD and CWD-VF. The four cost functions (also referred to as variants A, B, C and D) assign the same cost if only one WC is still vacant but let this cost decay in a different manner as more WCs are vacant. As such we can investigate how this influences performance and energy consumption reduction.

- $\mathbf{C_{W,A}}$: the $C_{W,A}$ cost is equal to the C_W cost, the A is added to emphasize the difference with the other variants. The $C_{W,A}$ cost adds a cost to a SP if one of the r WCs has to be used to schedule the arriving packet on the SP:

$$\begin{aligned} C_{W,A} &= C_W \\ &= \alpha \cdot gap + (1-\alpha) \cdot delay + \beta \cdot D \cdot \left[1 - \delta_{wi}\right] \ . \end{aligned} \tag{3.3}$$

- $\mathbf{C_{W,B}}$: variant B is similar to variant A but the conversion cost is only added if the last vacant WC of the set of r WCs has to be used to schedule the packet on this SP. If more than one WC is vacant, the extra cost is omitted. If v ($v = 1...r$) WCs are vacant the assigned cost is given by:

$$\begin{aligned} C_{W,B} = \alpha \cdot gap + (1-\alpha) \cdot delay \\ + \beta \cdot D \cdot \left[1 - \delta_{wi}\right] \cdot \delta_{1v} \ . \end{aligned} \tag{3.4}$$

- $\mathbf{C_{W,C}}$: variant C assigns a linearly rising extra cost as less WCs are vacant:

$$C_{W,C} = \alpha \cdot gap + (1-\alpha) \cdot delay + \beta \cdot D \cdot [1 - \delta_{wi}] \cdot \frac{r - v + 1}{r} . \quad (3.5)$$

- $\mathbf{C_{W,D}}$: variant D assigns an exponentially rising extra cost as less WCs are vacant. Correspondingly, an extra algorithm parameter ϵ ($0 < \epsilon < 1$) is introduced:

$$C_{W,D} = \alpha \cdot gap + (1-\alpha) \cdot delay + \beta \cdot D \cdot [1 - \delta_{wi}] \cdot \epsilon^{v-1} . \quad (3.6)$$

  Like the algorithm parameters α and β, ϵ can be varied, with overall performance varying along. As such, also this parameter is taken into account in the performance analysis presented below.

Packets are assigned to the SP with the lowest cost. The above costs are used if at least one WC is vacant, i.e., if the set of considered wavelengths consists of all wavelengths. If no WC is vacant ($v = 0$), only the arriving wavelength is in the available set and all C_W cost variants are reduced to the C cost. For $C_{W,A}$ the SCs are the same as for C_W: the delays, the gaps and the wavelength of the SPs (all reflected in the cost $C_{W,A}$). For $C_{W,B}$, $C_{W,C}$ and $C_{W,D}$ the number of vacant WCs is also used as a SC (as reflected in the cost).

3.4.2 Performance results

We compare the LP of the 8 new scheduling algorithms in Tables 3.2 and 3.3. Keeping the value of the algorithm parameter α fixed to the optimal values indicated in Table 2.6 and Table 2.7 respectively, we vary β to achieve improved performance. The corresponding optimal values of β are also shown in Tables 3.2 and 3.3. The LPs are compared with the LPs of C for the non-void filling algorithms and C-VF for the void-filling algorithms with the same values of α, c and r. From Tables 3.2 and 3.3 one finds that introducing a term that penalises the use of the WCs improves performance significantly, especially when the number of WCs is limited. If the use of a WC is penalised for a limited number of WCs, the WCs are used only for those conversions

that give a significant reduction in the assigned cost, in the sense of reducing the LP significantly. Over-restricting the use of the WCs, however, results in an under-utilisation of the WCs. An optimal β thus emerges for intermediate values. For both CWD and CWD-VF, values $\epsilon = 1/3, 1/2, 2/3$ have been considered but only $\epsilon = 2/3$ is shown because this high value of ϵ results in the largest performance improvement. Moreover, the difference in achievable LP between the different types of C_W costs is small. When $c = 4$ both for non-void-filling algorithms (9.4 %) and void-filling algorithms (25.6 %) the largest improvement is realized in the case of three WCs. When $c = 8$ the largest improvement is achieved for a higher number of WCs. With exception of the non-void-filling CW algorithms when $r = 1$, the optimal improvements for $c = 8$ are larger.

As for the optimal value of β, the value decreases with increasing number of WCs. This is intuitive: indeed, as more WCs are present, the use of a WC can, and should be, less restricted. The β values for $c = 8$ are higher because for these small numbers of WCs ($1 \leq r \leq 4$) their usage has to be even more restricted as they have to be shared among twice as many arrival streams (arriving wavelengths). Again similar results are obtained for other values of D.

The larger improvement of the CW algorithms with respect to the C algorithms for $c = 8$ does not compensate the higher values for the LP of the C algorithms for $c = 8$. This can be seen in Table 3.4, which shows the actual optimal LP values of the CWC and CWC-VF algorithms. When comparing $c = 4$ and $c = 8$ for $1 \leq r \leq 3$ optimal performance of the CWC algorithm is worse for $c = 8$ than for $c = 4$. Also, for the CWC-VF algorithm the LP for $c = 8$ is higher for any $1 \leq r \leq 2$. Again when the number of WCs is severely limited, this restriction weighs heavier on performance than the number of wavelengths c. The introduction of a conversion cost term does shift the inversion in optimal performance to a lower number of WCs; for 4 WCs and above for CWC, and for 3 WCs and above for CWC-VF, performance is better for $c = 8$. In contrast, for both the optimal performance of C and C-VF algorithm this shift was not seen for $r < 5$.

Besides reducing the LPs, the introduction of a conversion cost term in the cost functions limits the utilization (and thus the energy consumption) of the WCs, just as in the case of unlimited wavelength conversion. To model the energy consumption of the WCs we again keep track of the packets' payload (packet length) rather than the number of converted packets, since conversion time is proportional to the length of the packet. The ratio of the

Table 3.2: Maximum reduction in LP and corresponding β of new algorithms relative to C and C-VF respectively for $c = 4$ and $D = 100$.

# wavelength converters *r*	0	1	2	3	4	∞
LP reduction CWA			9.3%	8.6%	4.0%	
LP reduction CWB	0%	6.5%	7.2%	6.7%	3.1%	0%
LP reduction CWC			8.9%	9.4%	5.5%	
LP reduction CWD, ε=2/3			9.2%	9.0%	5.5%	
optimal β CWA			0.3	0.2	0.1	
optimal β CWB		0.4	0.3	0.3	0.2	0
optimal β CWC			0.3	0.3	0.2	
optimal β CWD, ε=2/3			0.3	0.3	0.2	
LP reduction CWA-VF			22.0%	23.3%	19.8%	
LP reduction CWB-VF	0%	13.8%	19.1%	18.8%	15.5%	0%
LP reduction CWC-VF			22.7%	25.6%	22.6%	
LP reduction CWD-VF, ε=2/3			23.2%	25.5%	23.0%	
optimal β CWA-VF			0.6	0.5	0.4	
optimal β CWB-VF		0.8	0.7	0.7	0.7	0
optimal β CWC-VF			0.8	0.7	0.7	
optimal β CWD-VF, ε=2/3			0.7	0.7	0.7	

payload of converted packets to the payload of all packets that are not lost, is our energy consumption measure of the WCs. Results are shown in Fig. 3.14 for $c = 4$ and in Fig. 3.15 for $c = 8$. Results are displayed for $D = 100$ and the applicable optimal algorithm parameters. As expected the energy consumption increases with an increasing number of wavelength converters present in the system. This increment is lower for each additional wavelength converter as more wavelength converters are already present in the system. Besides this it is clear that as the CWC and CWC-VF algorithms penalize the usage of the wavelength converters, the energy consumption is lower for these algorithms.

Table 3.3: Maximum reduction in LP and corresponding β of new algorithms relative to C and C-VF respectively for $c = 8$ and $D = 100$.

# wavelength converters *r*	0	1	2	3	4	∞
LP reduction CWA			13.3%	21.4%	29.8%	
LP reduction CWB	0%	6.3%	11.2%	16.4%	20.7%	0%
LP reduction CWC			12.9%	21.5%	30.7%	
LP reduction CWD, ε=2/3			13.4%	21.6%	30.4%	
optimal β CWA			0.5	0.5	0.4	
optimal β CWB		0.7	0.5	0.5	0.5	0
optimal β CWC			0.6	0.5	0.5	
optimal β CWD, ε=2/3			0.6	0.5	0.5	
LP reduction CWA-VF			33.3%	46.9%	55.2%	
LP reduction CWB-VF	0%	16.2%	29.0%	37.7%	44.1%	0%
LP reduction CWC-VF			32.2%	49.0%	60.2%	
LP reduction CWD-VF, ε=2/3			34.1%	49.4%	59.9%	
optimal β CWA-VF			1.1	0.9	0.8	
optimal β CWB-VF		1.6	1.1	1.0	1.0	0
optimal β CWC-VF			1.1	1.1	1.1	
optimal β CWD-VF, ε=2/3			1.2	1.1	1.1	

In Fig. 3.16 the achieved marginal reductions in LP for non-void-filling algorithms for $c = 4$ by adding increasingly more WCs are shown. As we can see this marginal reduction decays (almost exponentially) with increasing number of WCs; i.e., the marginal benefit of a WC reduces as more WCs are already present. For example we can see that adding a fourth WC gives a larger reduction in LP than adding 36 (the fifth through the fortieth) additional WCs. Here the number of WCs (40) is obtained as $c \cdot (N + 1)$ (see above). From Fig. 3.16 we can also observe that adding a first WC has a larger

Table 3.4: Optimal LPs of CWC and CWC-VF for $c = 4$ and $c = 8$ with $D = 100$.

# wavelength converters *r*	0	1	2	3	4	∞
LP CWC, optimal α & β, c=4	18.14%	11.40%	7.62%	5.26%	3.98%	2.99%
LP CWC, optimal α & β, c=8	18.19%	12.71%	9.02%	5.93%	3.55%	0.07%
LP CWC-VF, optimal α & β, c=4	9.02%	4.04%	2.15%	1.26%	0.85%	0.46%
LP CWC-VF, optimal α & β, c=8	9.12%	4.66%	2.28%	1.01%	0.44%	0.004%

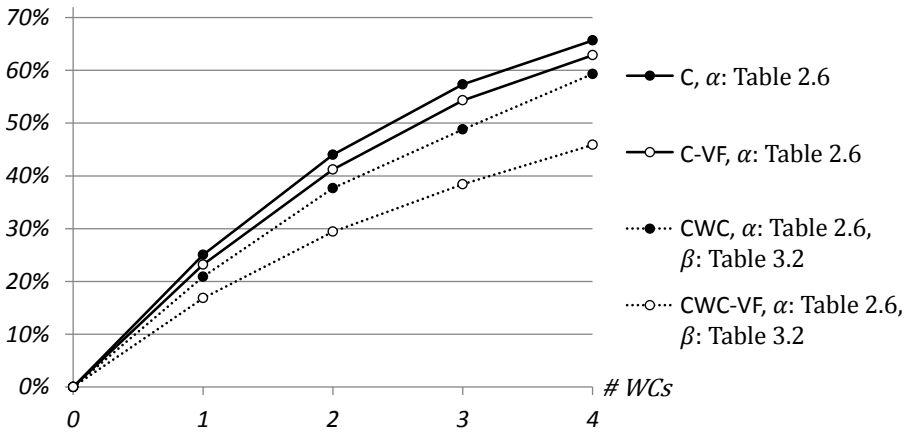


Figure 3.14: Energy consumption measure as a function of the number of WCs for $c = 4$ and $D = 100$.

effect for the C and CWC algorithms than for the G-D algorithm. in other words, the marginal benefit of adding the first WC is largest for the newly proposed algorithms. In Fig. 3.17 similar results are shown for void-filling algorithms. In Fig. 3.18 and Fig. 3.19 the corresponding results for $c = 8$ are shown. Although adding the first WCs give a smaller improvement in LP than it does for $c = 4$, it is clear that the marginal benefit of a WC also reduces (be it less steep) as more WCs are already present. As a consequence the marginal benefit of adding the fifth through the eightieth is not negligible.

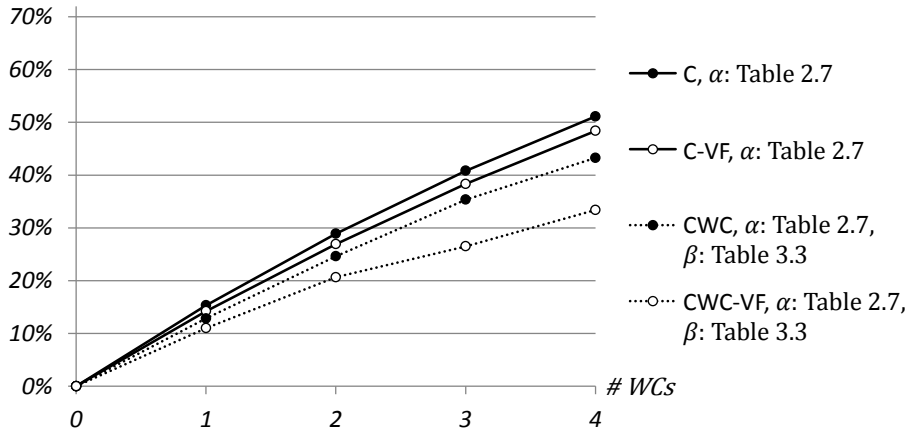


Figure 3.15: Energy consumption measure as a function of the number of WCs for $c = 8$ and $D = 100$.

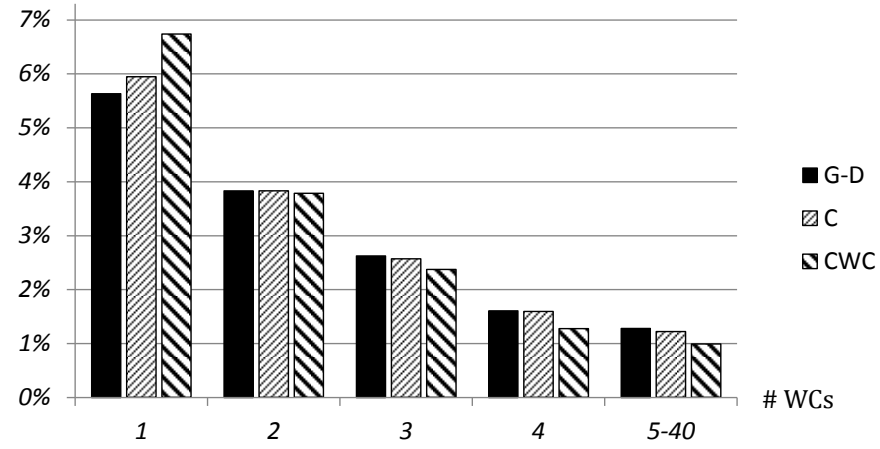


Figure 3.16: Marginal LP reduction achieved by adding the i-th WC ($i = 1 \ldots 4$ and $i = 5 - 40$: cumulative reduction) for G-D, C and CWC. $c = 4$, $D = 100$, α as in Table 2.6 and β as in Table 3.2.

3.5 Methodology

Similar to the cost-based algorithms of Chapter 2, the performance of the CW, CW-VF, CWA, CWA-VF, CWB, CWB-VF, CWC, CWC-VF, CWD and CWD-VF algorithms is evaluated by means of Monte Carlo simulation and discrete event simulation (DES). As an illustration the Matlab-like based pseudocode of two algorithms is added in Appendix A and B. Appendix A contains the pseudocode of the non-void-filling C algorithm for a setting of unlimited

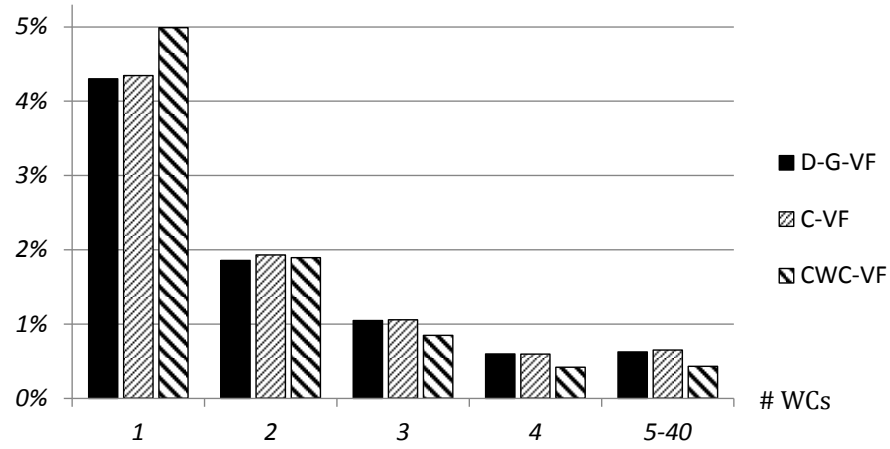


Figure 3.17: Marginal LP reduction achieved by adding the i-th WC ($i = 1 \ldots 4$ and $i = 5-40$: cumulative reduction) for D-G-VF, C-VF and CWC-VF. $c = 4$ $D = 100$, α as in Table 2.6 and β as in Table 3.2.

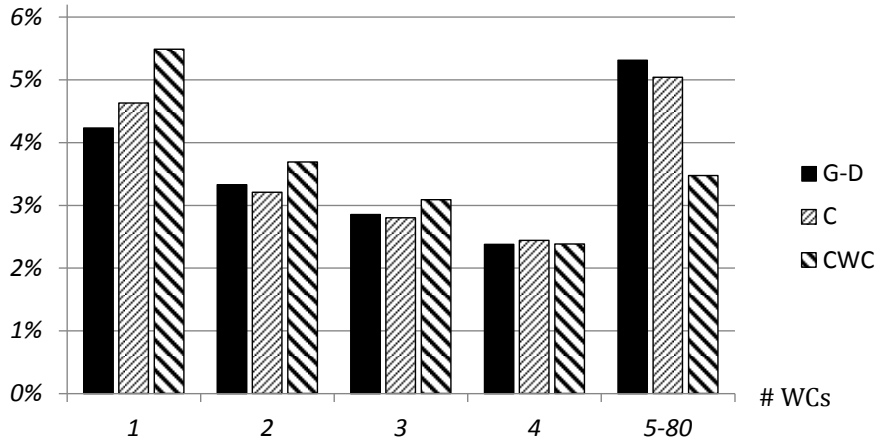


Figure 3.18: Marginal LP reduction achieved by adding the i-th WC ($i = 1 \ldots 4$ and $i = 5-80$: cumulative reduction) for G-D, C and CWC. $c = 8$ $D = 100$, α as in Table 2.7 and β as in Table 3.3.

wavelength conversion. Appendix B contains the pseudocode of the void-filling CWC-VF algorithm for the setting of a limited number of wavelength converters. In Section 2.2.3 more details can be found on the exact implementation of all variables, system components and their interactions. Below we focus on the most important aspects and those aspects distinctive to cost-based algorithms taking into account the number of available wavelengths.

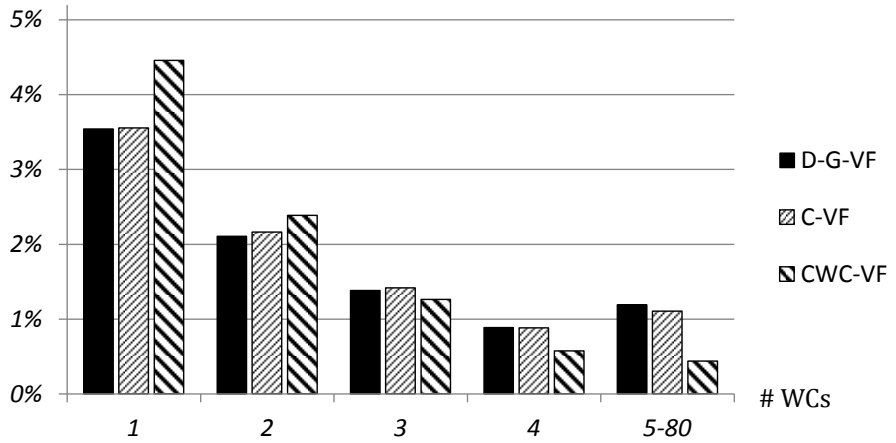


Figure 3.19: Marginal LP reduction achieved by adding the i-th WC ($i = 1 \ldots 4$ and $i = 5-80$: cumulative reduction) for D-G-VF, C-VF and CWC-VF. $c = 8$ $D = 100$, α as in Table 2.7 and β as in Table 3.3.

Similar as for other algorithms, the events list consists of the arrivals of packets. In each iteration the arrival of a packet is simulated, i.e., the inter-arrival time (T_{packet}) (the time between the current packet and the next), the packet length of the current packet (B_{packet}) and the wavelength w ($w = 1 \ldots c$) on which it arrives, are generated using a random number generator. As opposed to simulations for the C cost-based algorithms of Chapter 2, α, the gap-delay balance, is kept constant, i.e., equal to the optimal values of Chapter 2. Other parameters as β, the conversion cost parameter, and ϵ , an extra parameter of the CWD and CWD-VF algorithms, are changed in different simulation runs.

The system state again consists of one (unlimited number of wavelength converter capacity) or two (limited number of wavelength converters) parts. The first part of the system state always includes the description of the provisional schedule, i.e., a c-dimensional vector for the non-void-filling algorithms and a three-dimensional void matrix for the void-filling algorithms. Taking into account the gap, the delay and the number of available WCs (in a certain way), the algorithms assign a cost to each SP. If no eligible SP exists, the packet is lost. Otherwise, the algorithms choose the SP with the lowest cost to schedule the newly arrived packet on. After all system state variables are updated, the procedure is then repeated until a certain number of arrivals have been processed.

If the number of WCs is limited, the system state additionally includes information about the occupation of the WCs in the second part. When moreover the cost takes into account the actual number of available WCs, the for loop checking the availability of the WCs therefore cannot break after a first available WC has been found but has to check all WCs. This however takes little time as the number of WCs is, in general, severely limited.

Again the void-filling algorithms are computationally more complex and more difficult to implement in a switch than the non-void-filling algorithms. Within both classes of algorithms, however, no significant difference exists in complexity of implementation. Because existing algorithms also use gap and delay to choose among the SPs, a simple additional calculation of a cost-based on the same gap and delay (and possibly the number of wavelength converters) does not slow down the algorithm. This is important as we can thus say that any improvement achieved within a class of algorithms is achieved without complicating implementation.

3.6 Conclusions

In this chapter we continued on the path of parametric cost-based scheduling algorithms we introduced in Chapter 2 by extending the cost function with a term that takes into account the use of the wavelength converters.

In the setting of unlimited wavelength conversion it was shown that, within certain boundaries, reduction in energy consumption can be traded off for performance decrease by means of a single parameter β. Results suggest that the region of 0 - 20 % energy consumption reduction yields improved energy efficiency (i.e., more energy consumption reduction than performance decrease) with the given method. For larger reductions, a different approach is needed. When α and β are chosen in such away that performance in terms of LP of existing benchmark algorithms is matched, the energy consumption is reduced up to 28 % for non-void-filling algorithms and 37 % for void filling algorithms.

In the setting of a limited number of wavelength converters, introducing a term in the cost function that penalises the use of the wavelength converters, can improve both the performance and the energy consumption simultaneously, i.e., without the trade-off. This is because for a limited set of wavelength converters, introducing a term that penalises their use results in a more effective use of a scarce resource. Over-restricting the use of the WCs

however, results in an under-utilisation of the WCs. An optimal intermediate value of β for which LP is optimized thus emerges. This optimal value increases for a higher number of wavelengths and a lower number of WCs. Indeed, in these cases the usage of the WCs has to be even more restricted as they are shared more. Performance of four different cost functions that take into account the use of the WCs in a different manner, was evaluated. In general a wavelength conversion cost that rises as less WCs are vacant gives the best LP improvement in the setting of a limited number of WCs. As little performance difference exists between different rising costs, a linear rising cost should be preferred because of simplicity.

Similar to the algorithms evaluated in Chapter 2, the algorithms in this chapter perform (slightly) worse when more wavelengths are present for an equal overall load and when the number of WCs is strictly limited. This is as opposed to the setting of unlimited wavelength conversion, in which more wavelengths result in a higher multiplexing gain and thus highly (multiple orders of magnitude) improved performance. For a limited number of WCs the rescheduling is bounded by the number of WCs which have to be shared among a higher number of requests per time unit to change the wavelength. As more WCs are added, the WCs become less and less a limiting factor and the number of wavelengths determines how good contention is resolved. There is thus an intermediate value of the number of WCs for which a higher number of wavelengths becomes beneficial. The value for which this inversion occurs is higher for non-void-filling algorithms. Moreover, the introduction of a wavelength conversion cost term shifts this value to a lower number of WCs when compared to the algorithms of Chapter 2.

Looking at the marginal reductions in LP for an increasing number of WCs, we can see that this marginal reduction decays (almost exponentially) with an increasing number of WCs; i.e., the marginal benefit of adding a WC decreases as more WCs are already present. This is true for all algorithms and settings. Besides this it was clear that for the cost-based algorithms, especially those that take into account a wavelength converter cost, the first WC brings a larger reduction in LP. The marginal benefit of adding a first WC is thus largest for the newly proposed algorithms.

Chapter 4

Void-creating algorithms

There is not the slightest indication that nuclear energy will ever be obtainable. It would mean that the atom would have to be shattered at will.

Albert Einstein, 1932

The bomb will never go off. I speak as an expert in explosives.

Admiral William Leahy, U.S. Atomic Bomb Project, 1945

In Chapter 2 and 3 we were able to validate the usefulness of a cost-based approach to achieve an increased performance (decreased LP) and decreased WC energy consumption. Although these algorithms can improve the performance significantly, their structure is very similar to existing algorithms. Indeed, they can also be split up in two big categories: void-filling and non-void-filling algorithms. In contrast to the latter, the former allow packets to be scheduled before already scheduled packets, filling the so-called voids (unscheduled periods between already scheduled packets), thereby improving the performance (in terms of LP) significantly. As these void-filling algorithms keep track of all voids, also those that are not likely to be filled, this performance improvement comes at the cost of an increased computational complexity. As no trade-off between performance and computational complexity is possible in these algorithms, this is especially an issue in a typical setting with small FDLs (to minimize the average packet delay and/or the footprint of the buffer) in which voids are unlikely to be filled or even unfillable. In this chapter we therefore explore a new type of algorithm that selectively creates larger voids only when they will likely be filled in the future. This type of algorithm, which was also analyzed in [86, 89–91] does not

only fill the available voids, it also, based on the system conditions, controls the creation of the voids. In this way algorithms are made more powerful, enabling better switch performance while at the same time it allows to trade-off performance for computational complexity.

4.1 Assumptions

For both the fixed and general packet size setting mostly the same assumptions as in Chapter 2 and 3 are made. The overall framework is a dedicated conversion before buffering setting corresponding to the setting of Fig. 1.8 and 1.9. This setting assumes a $K \times M$ optical switch configuration. Packets arrive on a finite number of incoming ports K, on c different wavelengths $\lambda_1, \ldots, \lambda_c$. Each packet arrives on a certain wavelength and is switched (still on this wavelength) to one of the M output ports according to the packet header destination information. Each output port thus accepts packets from K ports, on c wavelengths. The output port is connected to a single fiber with the same c different wavelengths and we assume at each of the M output ports the joint packet arrival process is a Poisson process with the wavelength w of each arriving packet randomly and uniformly distributed among the c wavelengths. Again, an arbitrary single output port is analyzed.

As opposed to Section 4.4, where full wavelength conversion capability is assumed, we assume no wavelength converters are present at the output port in Sections 4.2 and 4.3. While other wavelengths may thus be used for packet switching within the same switch, they operate independently of each other and all packets are processed on the same wavelength upon which they arrive. We can thus confine our analysis of the output port to a single wavelength λ_1 (Fig. 4.1) in Sections 4.2 and 4.3.

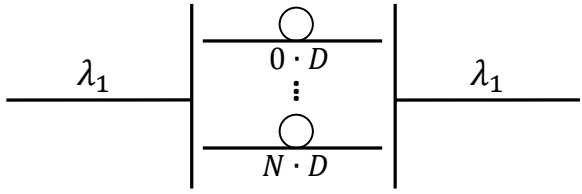


Figure 4.1: The modelled output port as analyzed in Sections 4.2 and 4.3: with FDLs and a single wavelength λ_1.

Independently of the analyzed number of wavelengths, each output port has an optical buffer in which $N + 1$ FDLs are available to schedule incoming

packets. The lengths of the FDLs are consecutive multiples of a basic value D called the granularity. Packets arrive at the output port according to a Poisson process, with exponentially distributed inter-arrival times T and average $E[T]$. The packet size is denoted B, its average is denoted $E[B]$. The overall incoming traffic load at the output port is thus equal for all packet size distributions and given by $\rho = E[B]/(c \cdot E[T])$. The nature of the arrival process implies possible overlap of packets at the entrance of the output port. It is this overlap that causes contention and is resolved by sending one of the contending packets on a different wavelength or through one of the FDLs (or a combination of both). A packet is lost if none of the FDLs on any of the wavelengths can be chosen without causing contention with one of the other packets present in the system. It is therefore necessary to schedule packets as wisely as possible among the wavelengths and FDLs. In general, LP (loss probability) and average packet delay are the main performance measures. Besides these, we will also, if relevant, consider the average packet gap, i.e., the time between packets on the outgoing line, and the LPsize, i.e., the sum of all lost packets multiplied with their size divided by the sum of all arrived packet sizes.

4.2 Fixed packet size on a single wavelength

The practical importance of the case of fixed packet size, not in the least for OPS testbeds, motivates our choice to focus on the case of fixed packet sizes (in combination with multiple wavelengths) in this and the next section. Particularly, the case is considered of a fixed packet size B equal to the granularity, $B = D$. Matching the granularity and the (average) packet size is a natural and recommendable choice in switch design in view of performance (see [131]). As many results are available for this value of the granularity, results can be easily compared.

4.2.1 Approach and concept: gap thresholds

At the output port under study, scheduling is done separately for each packet upon its arrival. In this setting this amounts to assigning a single variable (j) to the packet, corresponding to the delay line $(j = 0 \ldots N)$ the packet is scheduled on. This is again done by means of a provisional schedule, of which an example for this setting is given in Fig. 4.2. The provisional schedule is set out horizontally and the vertical marks represent the delays of the FDLs $(j = 0 \ldots N = 5)$. The granularity in the example is taken as unity $(D = 1)$ to ease notation.

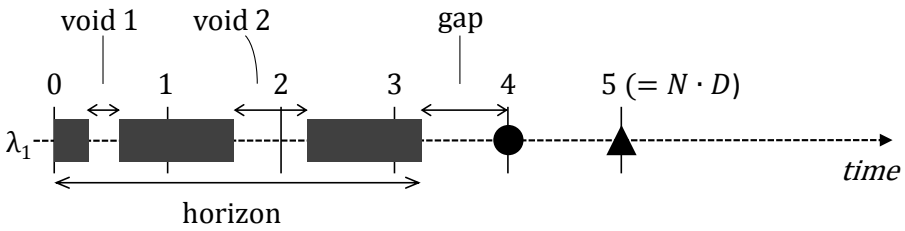
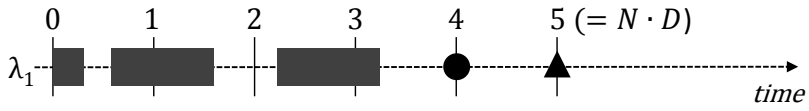


Figure 4.2: An example of a provisional schedule as analyzed in Section 4.2: fixed size packets and a single wavelength λ_1.

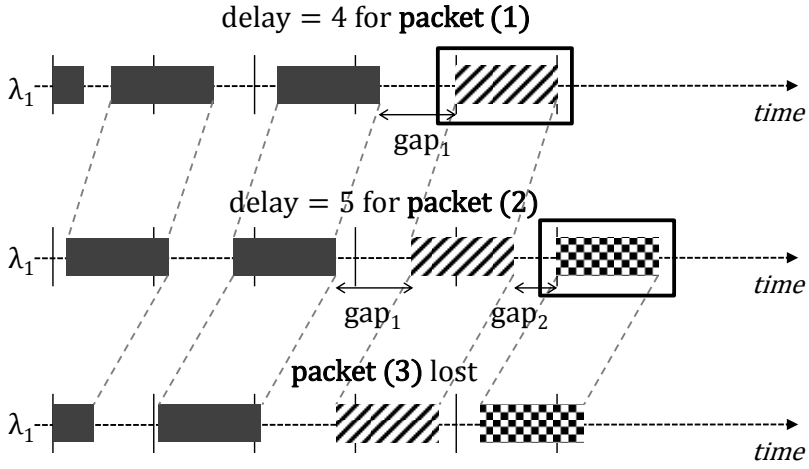
A packet has to be scheduled on a PSP (● or ▲) without overlapping with any of the already present packets. If no such PSP is available, the packet is lost. As $D = B$ in the example of Fig. 4.2 no voids equal to or larger than the packet length (*fillable voids*) are created and thus none can be filled. Only voids smaller than the packet length (*unfillable voids*) are created. In this setting of a single wavelength, and $D = B$, all void-filling and non-void-filling algorithms schedule the packet on the first PSP to the right of the horizon (D-G algorithm). This is unless the horizon is larger than $N \cdot D$, in which case the packet is lost. Performance (in terms of LP and average packet delay) will thus be the same for all void-filling and non-void-filling algorithms in this setting. Hence, regardless of the algorithm chosen for implementation, the actual scheduling always boils down to the non-void-filling D-G algorithm, and the horizon determines all.

In order to improve the performance of this system, fillable voids are created by scheduling the packet on the second PSP (triangle in Fig. 4.2) instead of the first PSP (circle in Fig. 4.2) to the right of the horizon. As $D = B$ this creates voids larger than B (fillable voids) which, if favorably positioned with respect to a PSP, can be filled. A favorable position occurs when the void overlaps the PSP at least a packet length to the right of the PSP, i.e., the (fillable) void is *reachable*. Analogously, a (fillable) void that is not positioned favorably with respect to a PSP is called *unreachable*. Note that an unfillable void can never be reachable. When a fillable void created by a first packet can be filled by any subsequent packet the average gap as well as the average delay encountered per packet will typically be lower than in the case when a fillable void was never created.

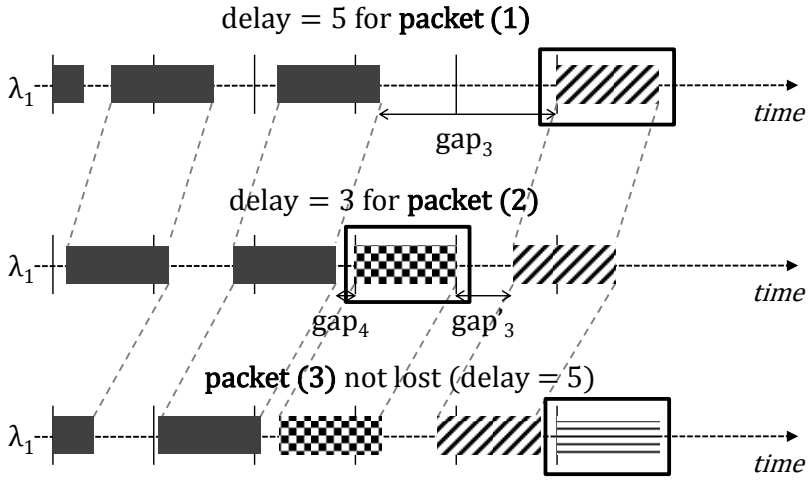
This is illustrated by means of an example in Fig. 4.3 in which the same scenario of three subsequent packet arrivals is analyzed. Fig. 4.3 compares the evolution of the provisional schedules from arrival to arrival without (Fig. 4.3b) and with (Fig. 4.3c) void creation. Packet (1) (diagonally hatched)



(a) A scheduling dilemma: ● or ▲



(b) Three packets scheduled without void creation



(c) Three packets scheduled with void creation

Figure 4.3: Evolution of the provisional schedule for the considered example of three arriving packets.

arrives at $time = 0$ (reference time). Packet (2) (checkerboarded) and packet (3) (horizontally hatched) arrive a time of 0.44 and 1.19 respectively hereafter (assuming $D = B = 1$). These arrival instants again coincide with the zero reference time in the matching provisional schedule (second provisional schedule for packet (2) and third provisional schedule for packet (3) in both Fig. 4.3b and Fig. 4.3c), as this is how the provisional schedule works.

Without void creation. In Fig. 4.3b both packet (1) and (2) are scheduled on the first FDL to the right of the horizon of their specific provisional schedule, corresponding to the strategy used by the existing algorithms (both void-filling and non-void-filling) in this setting (D-G). This results in an average delay of 4.5 for packets (1) and (2) (delay of 4 for packet (1) and 5 for packet (2)). The total gap assigned to packets (1) and (2) equals $gap_1 + gap_2$. Moreover at the arrival instant of packet (3) (third provisional schedule in Fig. 4.3b) there is no FDL available to schedule packet (3), which thus results in a lost packet.

With void creation. In Fig. 4.3c packet (1) is scheduled on the second FDL to the right of the horizon, in this way creating a fillable void. Upon its arrival packet (2) is able to fill this fillable void as it is reachable (it overlaps the FDL that corresponds with a delay of 3 in a favorable way). The scenario with void creation has two clear benefits:

- Lower delays: the average delay for packets (1) and (2) equals 4 (delay of 5 for packet (1) and 3 for packet (2)) as opposed to an average delay of 4.5 for packets (1) and (2) without void creation.
- Reduced loss: the total gap assigned to packets (1) and (2) equals $gap'_3 + gap_4$ which equals gap_1 and thus is always smaller than in the case without void creation: $(gap'_3 + gap_4 = gap_3 - D = gap_1) < (gap_1 + gap_2)$. Because of this, the stacking of packets (1) and (2) is more dense in Fig. 4.3c and the last FDL is available to schedule packet (3) when it arrives. In this example thus no packet is lost when void creation is used.

It is clear from the above example that filling a fillable void may lower the average delay of the packets and the average gap size. Moreover the example shows that reducing the average gap size may mitigate the number of packets lost as it makes the stacking of the packets dense and reduces the chance the horizon exceeds $N \cdot D$, the only case in which a packet that arrives is lost. On the other hand, creating but not filling voids is disadvantageous for performance as the stacking becomes less dense. The chance the horizon

exceeds $N \cdot D$ (and, likewise, the average packet delay) seemingly increases. Whether or not a void is reachable, depends on the choppy evolution of the provisional schedule and thus on the stochastic arrival process. We can however maximize the chance that a void is reachable by creating only voids that are likely to be filled. As a first condition, we only allow a single fillable void to be present in the system. As soon as the ending of the fillable void is lower than $B = D$ it is smaller than the packet length $B = D$. It will never be reachable again in the future and the fillable void *expires*. An *expired void* is thus a void that was fillable but has turned unfillable. Because it is unfillable, it is omitted and a new fillable void is allowed to be created. A fillable void can of course also disappear because it is reachable the moment a new packet arrives. This packet will then fill this void, and a next arrival may be used to create a new fillable void. Besides allowing only a single fillable void in the provisional schedule we demand for two additional conditions to be met before creating a fillable void:

- A first condition relates to the size of the created fillable voids. As the void-creating algorithm chooses the second FDL to the right of the horizon, the created fillable voids will have a length between D and $2 \cdot D$. A void only slightly larger than D will have a small chance of being reachable. To fill a void, it indeed has to overlap an FDL at least a length of $D = B$ to the right of the FDL on a set of future, yet unknown, arrival instances. For a void almost equal to $2 \cdot D$ it is very likely to have a long period of reachability, although it is still possible that an unfavorable arrival pattern occurs. The first condition therefore states that the created void has to be larger than a certain value $D + y_n \cdot D$, i.e., $D + y_n \cdot D < fillable\ void$. Here y_n is called the *gap threshold*, an algorithm parameter varied in simulations ($0 < y_n < 1$ in steps of 0.01) with n the horizon index (see next).
- A second condition to create a fillable void states that the horizon has to be between two specific and consecutive FDLs: $(n-1) \cdot D < horizon < n \cdot D$ in which n is called the *horizon index* and is varied in simulations ($n = 1 \ldots N-1$). With this condition we want to investigate the effect of the position of void creation. Fillable voids created when n is closer to $N-1$ will stay longer in the system and thus given the Poisson arrival process, have a higher chance of being filled. This condition is used to create auxiliary results that allow us to obtain the optimal gap thresholds of the actual algorithm in which void creation is allowed for all horizon indexes.

If one of these conditions is not met or another fillable void is already present in the provisional schedule, the packet is used to fill this void or otherwise is

scheduled on the first FDL to the right of the horizon, creating an unfillable void. Similar to regular void-filling algorithms, this void-creating algorithm allows packets to be scheduled out of order.

4.2.2 Performance results

To obtain the LP and average packet delay of the void-creating algorithm, the following method is used. A first and auxiliary set of simulations only allows to create a void when both conditions are met, i.e., $D+y_n \cdot D < fillable\ void$ and $(n-1) \cdot D < horizon < n \cdot D$. By varying y_n ($0 < y_n < 1$ in steps of 0.01) for a fixed n we determine the optimal gap threshold when voids are only allowed to be created on horizon index n. This is done separately for each value of n ($n = 1 \ldots N-1$) and gives us two sets (one for LP and one for average packet delay) of both $N-1$ optimal gap threshold values for which either LP or average packet delay is minimized. These are called the *single creation point thresholds*.

The actual void-creating algorithm allows voids to be created on all horizon indexes but still optimizes the gap threshold y_n for each horizon index: i.e., depending on the position of the horizon, the condition on the gap size can either be more or less strict. Note that the restriction of only one fillable void is maintained. Allowing void creation for each horizon index results in a very large parameter space for optimization. With $0 < y_n < 1$ in steps of 0.01 and $N-1$ different horizon indexes the parameter space contains 100^{N-1} combinations. Even for a small value of N this results in an unreasonably long simulation time.

We therefore use the following iterative approach to approximate the behavior of the actual void-creating algorithm. In a first iteration the fillable void (only one allowed) can be created on all horizon indexes, with all gap thresholds but one fixed to the optimal single creation point thresholds (auxiliary set of simulations). The one gap threshold that is not fixed is varied and optimized ($0 < y_n < 1$ in steps of 0.01). Consecutively this is done once for each horizon index ($n = 1 \ldots N-1$) in a different simulation trace. In this way the gap threshold of each horizon index is optimized, with the other thresholds fixed. This gives us a set of $N-1$ new optimal gap thresholds which are called the *first iteration thresholds*. In a second iteration we fix all gap thresholds but one to these first iteration thresholds and again optimize the one that is not fixed. Again only the gap threshold that is optimized is varied ($n = 1 \ldots N-1$). This results in a set of $N-1$ *second iteration thresholds*. These iterations can be repeated as many times as

desired until an acceptable convergence is noticed in the iteration thresholds and their corresponding performance, marking a good approximation of the actual void-creating algorithm. This iterative approach is used for both the LP and delay thresholds. In general only a few (2-3) iterations were necessary to obtain a complete convergence of the thresholds in the examples of this chapter.

As in the given setting of $D = B$ the void-filling algorithms perform exactly the same as the non-void-filling algorithms, we only use the latter as a comparison for the performance of the void-creating algorithm. We thus first simulate the system without void creation, i.e., all packets are scheduled on the first FDL to the right of the horizon (D-G). If the horizon is larger than $N \cdot D$ on arrival of a packet, it is lost. The number of FDLs was assumed ten ($N + 1 = 10$) and $D = B = 1$. For a fixed load ρ of 80 % this gives an LP of 14.46 % of the arriving packets. When the load is fixed to 60 % the LP is 2.08 %. The average packet delay under this algorithm is 6.14 time units for a load of 80 % and 2.98 time units for a load of 60 %. The number of simulated packets is $10 \cdot 10^6$ for a load of 80 % and $10 \cdot 10^7$ for a load of 60 %. For all simulations this yields confidence intervals too narrow to be displayed on the figures.

4.2.2.1 Optimized for loss probability

The results of the auxiliary set of simulations, i.e., the single creation point thresholds and their corresponding performance, are shown in Fig. 4.4 and 4.5. The curves marked as "single creation point" in Fig. 4.4 show the optimal gap thresholds for the different horizon indexes, when void creation is only allowed for that index and the LP is minimized. For example when a fillable void is allowed to be created if (and only if) $D < horizon < 2 \cdot D$, then the void has to be larger than $1.6 \cdot D$ ($y > 0.6$) for a load of 60 % to achieve the largest reduction in LP. The LPs with the threshold values of Fig. 4.4 are shown in Fig. 4.5. The LPs are shown relative to the LPs when no void creation is allowed, i.e., 2.08 % and 14.46 % for a load of 60 % and 80 % respectively. The graphs marked as "second iteration" on Fig 4.4 and 4.5 show the second iteration thresholds and the corresponding relative LPs obtained in the second iteration when the LP is optimized. Again the LPs are shown relative to the LPs when no void creation is allowed (D-G).

Looking at Fig. 4.4 we can see that for an equal horizon index, the single creation point threshold is larger, and thus stricter, for the lower load. This is because for lower load the number of expected packets to arrive, before the void disappears by crossing the zero delay line, is lower. This lack of high

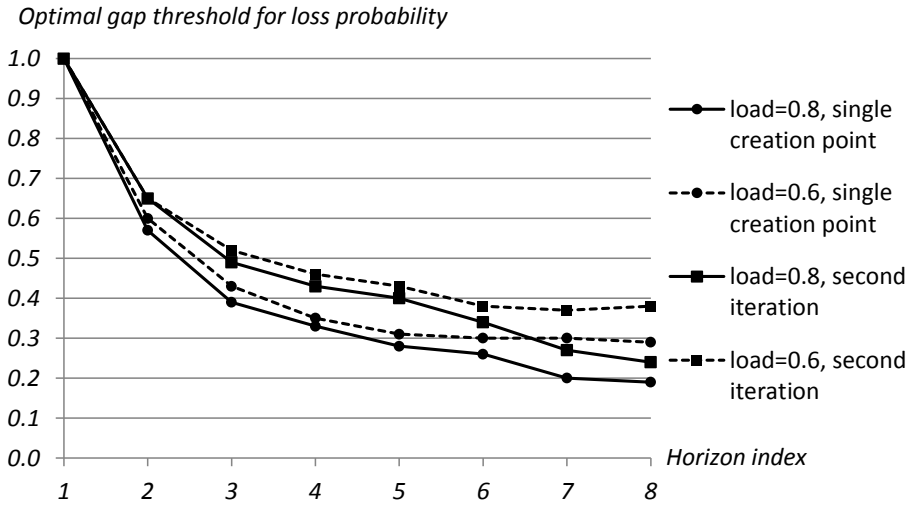


Figure 4.4: Gap threshold optimized for loss probability for fixed packet size on a single wavelength.

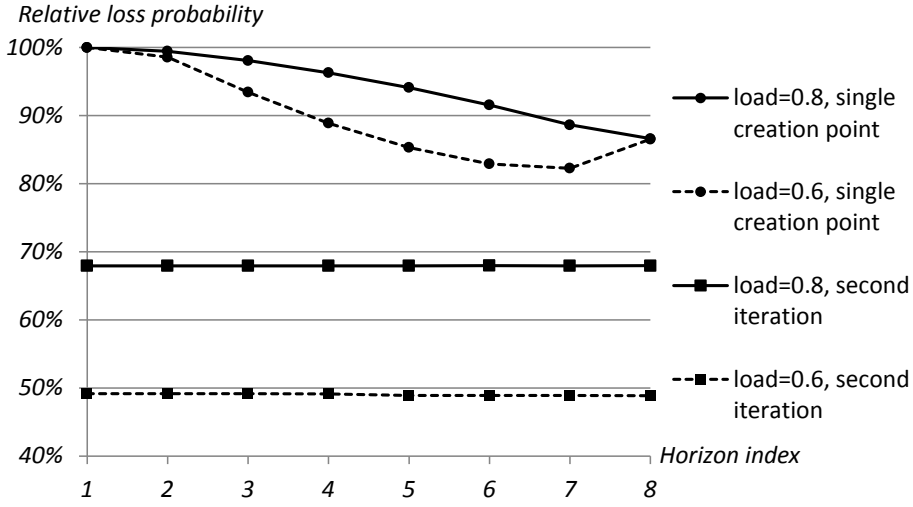


Figure 4.5: The optimized relative loss probability for fixed packet size on a single wavelength.

arrival density thus has to be compensated by a larger size of the created void. When the fillable void is only allowed to be created when $n = 1$ (i.e., the horizon is between 0 and D on the provisional schedule), no improvement in LP is achieved, and this for both loads. This is indicated by a single creation point threshold of 1 for $n = 1$, as this never occurs. For both loads the single creation point threshold decreases as the horizon index increases. As the horizon index on which fillable voids are allowed to be created increases, the created void stays longer in the system and thus given the Poisson arrival process, has a higher chance of being filled. For equal chance of being filled, smaller voids may be created as the horizon index increases.

Still on Fig. 4.4 we can see that for the second iteration the optimal gap thresholds are higher, and thus stricter, than for the corresponding single creation point graphs. As void creation is also allowed for other positions of the horizon, a stricter policy can indeed be applied. In general these second iteration thresholds also decrease with increasing horizon index (for the same reason as above). For a horizon index $n = 8$ and a load of 60 % however an inversion is spotted in the second iteration threshold. This inversion possibly relates to the inversion found in the LP reduction for a single creation point and a load of 60 % (discussed next).

Fig. 4.5 shows that for a single creation point the lower load allows for a bigger relative reduction in LP. For a load of 80 % the optimal LP monotonically decreases with an increasing horizon index. Opposed to this, for a load of 60 % less improvement in LP is possible when fillable voids are created when $n = 8$ than when $n = 7$. A possible explanation is that when the load is lower (60 %), the probability of finding a horizon index $n = 8$ is smaller than for higher load (80 %), thus reducing the number of instances fillable voids can be created and the LP is reduced. The load for which this inversion is observed may be linked to the critical load of 69.3 % for which the infinite system without void creation becomes unstable ($\rho = ln(2)$, see[131]), i.e., for load values higher than this critical load the number of packets in the system with an infinite number of FDLs grows unboundedly.

As the "second iteration" curves in Fig. 4.5 illustrate, a lower load also allows for a bigger reduction in LP when voids are allowed to be created on all horizon indexes. For a load of 60 % there is reduction in LP of more than 50 %. For a load of 80 % a reduction of about 32 % is achieved. This reduction is practically the same for all values of n as for all these simulation points void creation is allowed for all positions of the horizon, each with their distinct gap threshold.

4.2.2.2 Optimized for average packet delay

Similarly to Fig. 4.4 the curves marked as "single creation point" in Fig. 4.6 show the optimal gap thresholds for the different horizon indexes, when void creation is only allowed for that index and the average packet delay is minimized. Fig. 4.7 shows the corresponding average packet delays relative to the average packet delay when no void creation is allowed, i.e., 2.98 and 6.14 for a load of 60 % and 80 % respectively.

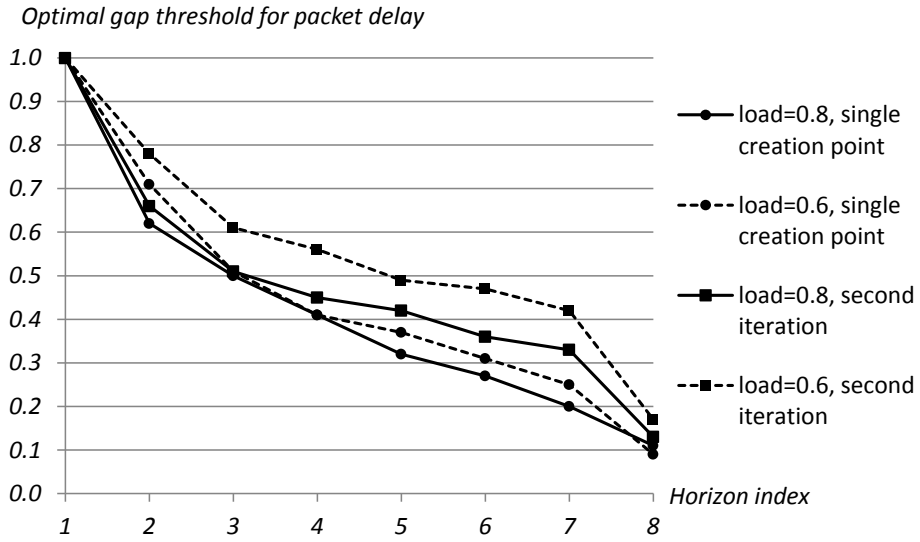


Figure 4.6: Gap threshold optimized for average packet delay for fixed packet size on a single wavelength.

Comparing Fig. 4.4 and Fig. 4.6 it is clear that the optimal thresholds are impacted by the optimization criterion (LP or delay), for both single creation point and second iteration. Despite this, similarities can be seen: the optimal threshold decreases with both increasing horizon index and increasing load. Also for a horizon index $n = 1$, no improvement in average packet delay is possible by creating a void. This again is indicated by a single creation point threshold of 1 for $n = 1$. The inversion spotted for the second iteration thresholds for a load of 60 % when $n = 8$ no longer occurs when optimizing for average packet delay, as all optimal thresholds decrease with increasing horizon.

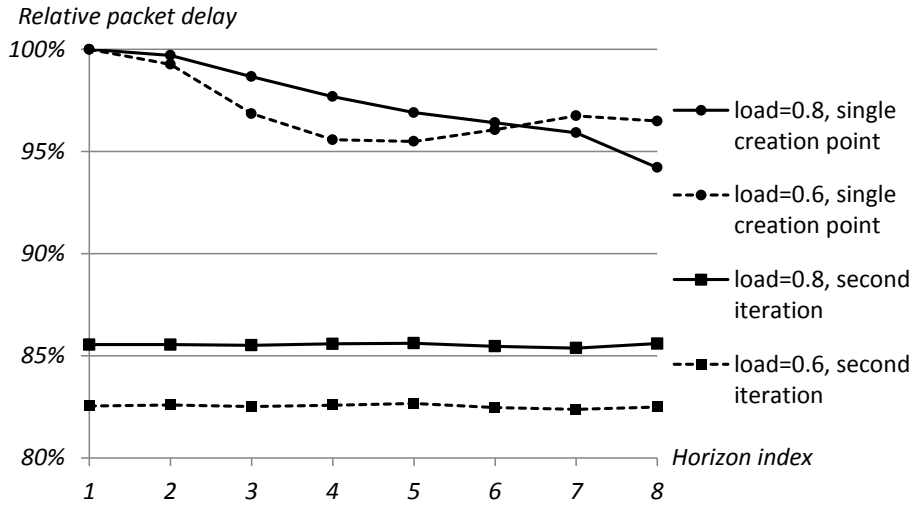


Figure 4.7: The optimized relative average packet delay for fixed packet size on a single wavelength.

Looking at Fig. 4.7 we see that the achievable improvements in average packet delay are smaller than the corresponding achievable improvements in LP in Fig. 4.5. Similar to Fig. 4.5, for a load of 80 % the optimal delay monotonically decreases with an increasing horizon index. The inversion spotted for a load of 60 % is spotted for a lower horizon index than in Fig. 4.5, i.e., $n = 6$ in Fig. 4.7 and $n = 8$ in Fig. 4.5. For $n = 8$ in Fig. 4.7 the inversion stops, and the achievable delay for $n = 8$ is smaller again than the achievable delay for $n = 7$. Despite this inversion stop, for both $n = 7$ and $n = 8$ the optimal delay is higher for a load of 60 % than for a load of 80 %. Again this inversion might be linked to the critical load of 69.3 % of the infinite system. When void creation is allowed for all horizon indexes a lower load still allows for a bigger reduction in average packet delay. For a load of 60 % there is an achievable procentual reduction in delay of about 18 %. For a load of 80 % the achievable reduction is limited to 14 %.

4.3 General packet size on a single wavelength

In Section 4.2 we proposed a new type of algorithm, the void-creating algorithm, that selectively creates larger voids only when they will likely be filled in the future. This algorithm does not only fill the available voids, but also, based on the system conditions, controls the creation of the voids, enabling better switch performance. As interesting as these results are, they

are limited to a single degenerate case in which the packet size is fixed. In this section we therefore extend the use of the void-creating algorithm to a setting with general packet size. We consider three additional distributions. A first additional distribution is an exponentially distributed packet size with the average packet size $E[B]$ equal to the granularity, $E[B] = D$. The second and third are uniformly distributed and have the same average packet size ($E[B] = D$) but different range. One is defined on the interval $[0,\ 2 \cdot D]$ and the other on $[0.5 \cdot D,\ 1.5 \cdot D]$.

In addition to the wider applicability in terms of packet size distributions the method and results in this section are backed by a numerical procedure that assigns a theoretical value to each void based on how likely the void will be filled in the future. This framework deepens the insight in the mechanism of the void-creating algorithm and makes it possible to apply it in other settings without (a lot of) additional simulations.

4.3.1 Approach and concept: void values

To create a void-creating algorithm for different packet size distributions the same single wavelength with $E[B] = D$ is considered. In this setting no fillable voids are created when the packet size is fixed ($B = E[B]$) and each packet is scheduled on the first FDL to the right of the horizon for all void-filling and non-void-filling algorithms. This is different for other packet size distributions in which each void larger than the minimal packet size (B_{min}) has a chance to be filled in the future. As a consequence the performance of non-void-filling and void-filling algorithms is no longer the same in this setting. More specifically, as void-filling algorithms enable voids to be tracked and filled, they will typically outperform their non-void-filling counterparts. Among the different void-filling algorithms however, little performance difference exists. This is as the more advanced void-filling algorithms tend to only thrive when there are many voids to choose from to schedule a packet, i.e., in a multiple wavelengths setting. As little performance difference exists among the void-filling algorithms, we therefore choose to stay consistent with the fixed size packet setting and use the D-G-VF algorithm as our benchmark. In this setting, it is thus the D-G-VF algorithm to which our improved void-creating algorithm will be compared in terms of performance.

Similar to the D-G-VF algorithm the void-creating algorithm first tries to fill a reachable void with the packet that has to be scheduled. The void closest to the zero delay reference line that overlaps an FDL in such a way that it can accommodate the packet is filled and split in two new voids, one leading and

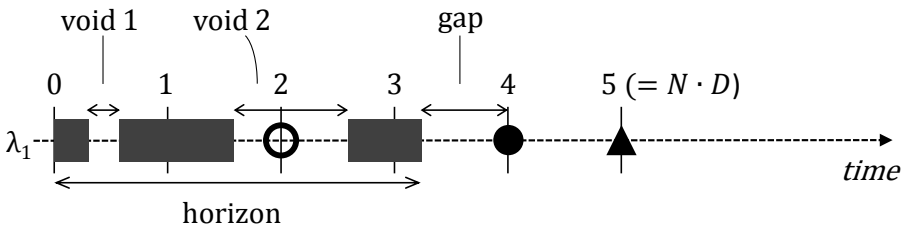


Figure 4.8: An example of a provisional schedule for a single wavelength when the packet size is variable ($B \neq E[B] = D = 1$).

one trailing the newly scheduled packet (void 2 (the circle) for packets smaller than 0.58 in the example of Fig. 4.8). In order to improve the performance the void-creating algorithm schedules in a different way if no reachable void is present in the system. If no void can be filled it sometimes schedules on the second FDL (triangle) instead of the first FDL (circle) to the right of the horizon. This is opposed to the D-G-VF algorithm, which always schedules on the first FDL to the right of the horizon. By doing so the void-creating algorithm creates voids that are larger and thus will be reachable more often than voids created by scheduling on the circle. It thus speculates that the filling of this larger void and its descendants (the successive leading- and trailing-voids) will result in an overall denser stacking of the packets.

To decide upon scheduling on the triangle or the circle a more thought-through consideration has to be made than in the case of fixed packet size. In the fixed packet size case the chance of filling a void smaller than the packet size $B = D$ is zero. The value of a void smaller than D is thus zero, irrespective of its length. The value of a void larger than D on the other hand rises monotonously with its length: the longer a fillable void is, the more reachable it will be in the future and thus the more value it has. If the difference in value between scheduling on the triangle and the circle is sufficient, the algorithm schedules on the triangle. Note that this theoretical value is not calculated in the void-creating algorithm of Section 4.2. Instead the gap threshold for which the difference in value is large enough is simply figured out by simulation. For other packet sizes however, the theoretical value of the void created by scheduling on the circle does not remain zero as the size of the gap changes. Instead, the value of both the circle and triangle voids increase as the gap and thus their length increases. The difference in their value as a function of the size of the gap depends on the exact distribution of the arriving packets.

As the method used in Section 4.2 provides little basis for extension towards applications with general packet size distribution we therefore propose a method to calculate a theoretical value of a void. The theoretical value assigned to a void is directly linked to the chance of the void being filled in the future and thus its ability to improve performance. In a nutshell, our approach is to confront this theoretical value of a void created by scheduling on the circle and that of a void created by scheduling on the triangle, basing our scheduling decision on the difference between the two. To assign a value to a void we introduced the concepts *lifetime* and *life cycle* of a void. A void's lifetime is defined as the period between the void's creation and the last moment during which it can be filled, under the hypothesis that no packet arrives. A void's life cycle $L_{max}(t)$ is a function of time that spans the void's lifetime, indicating for each time instant the maximum packet size allowable for a hypothetically arriving packet in order to still fit in the void. In Fig. 4.9 the life cycles of both the void by scheduling a packet on the circle and the triangle in Fig. 4.8 are shown. For both voids the life cycle is, just like for any other void, a sawtooth-shaped function. For the void created by scheduling on the circle, this sawtooth-shaped curve by definition never rises above the gap size, and, in the given example, periodically drops to zero, going through periods of complete unreachability for arriving packets. Complementary to this, the curve in case of scheduling on the triangle remains above zero throughout the void's entire lifetime.

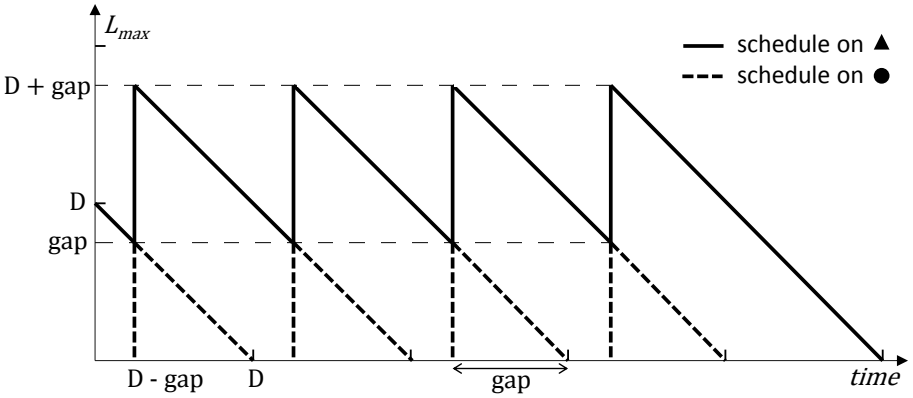


Figure 4.9: Life cycles of the voids created by scheduling on the ● and ▲ in Fig. 4.8.

Although different packet sizes are allowed in the considered setting, all packets arrive according to a Poisson process, i.e., with exponentially distributed inter-arrival times. The average *arrival rate* $\lambda = 1/E[T]$ can be derived from

the chosen load $\rho = E[B]/E[T]$ resulting in $\lambda = \rho/D$ as for all assumed packet size distributions $E[B] = D$. Ignoring the presence of other voids (current as well as future voids), the chance a void, not yet filled, is filled by a packet in the next infinitesimally small time interval dt is given by $\lambda \cdot Prob[B \leq L_{max}(t)] \cdot dt$. From the point of view of a void, arrivals of packets that fit into the void occur according to an inhomogeneous Poisson process with arrival rate $\lambda(t) = \lambda \cdot Prob[B \leq L_{max}(t)]$. Note that this Poisson process does not describe the packets that actually fill the void but the packets that would fit in the void under the hypothesis that no prior arrival already filled it. The overall expected number of packets to arrive in the life cycle of the void that are able to fill the void (Λ) is calculated by integrating $\lambda(t)$ over the lifetime of a void.

$$\Lambda = \int_{lifetime} \lambda(t)dt = \frac{\rho}{D} \cdot \int_0^\infty Prob[B \leq L_{max}(t)]\, dt. \tag{4.1}$$

Due to the nature of the Poisson proces the chance no packet arrives during its lifetime is given by $e^{-\Lambda}$. The overall normalized expected packet size to arrive in the system that is able to fill the void ($\bar{\Lambda}$) is calculated by a similar integral:

$$\bar{\Lambda} = \frac{\rho}{D} \cdot \int_0^\infty Prob[B \leq L_{max}(t)] \cdot \frac{E[B|B \leq L_{max}(t)]}{D}\, dt. \tag{4.2}$$

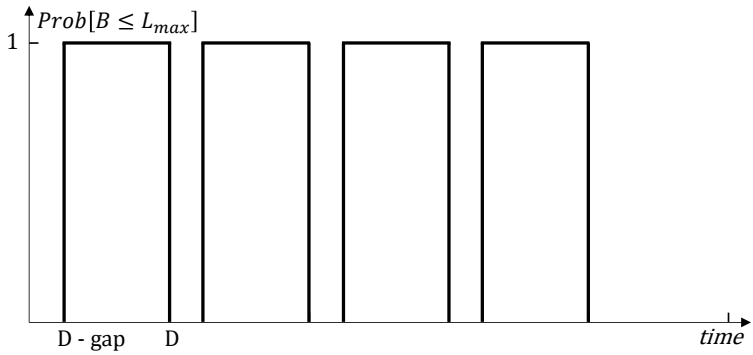


Figure 4.10: Integrand of Λ by scheduling on the ● and ▲ in Fig. 4.8 for a fixed packet size.

As an illustration, i.e., to highlight their differences, the integrands of Λ and $\bar{\Lambda}$ are shown for the considered packet size distributions in Fig. 4.10 - 4.17. Fig. 4.10 - 4.13 show the integrands of Λ. The curves in these figures are calculated and plotted based on the dashed and unbroken lines of Fig. 4.9 and

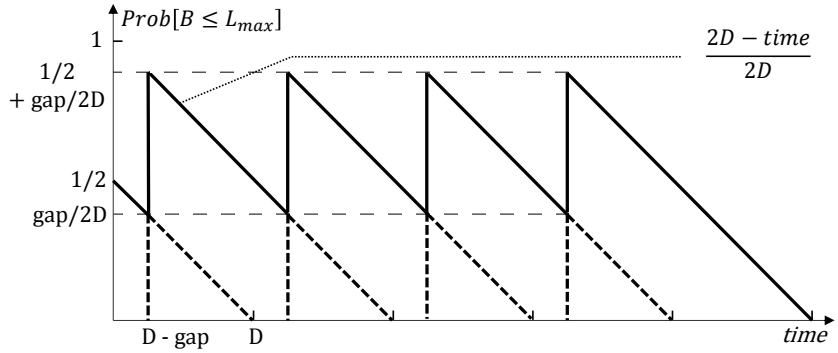


Figure 4.11: Integrand of Λ by scheduling on the ● and ▲ in Fig. 4.8 for a uniform distributed packet size on $[0,\ 2 \cdot D]$.

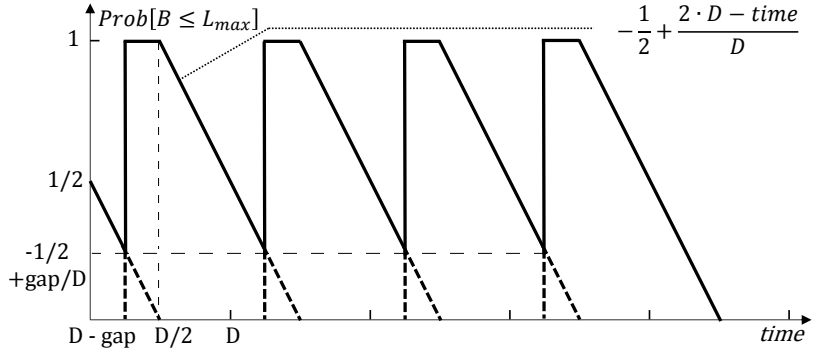


Figure 4.12: Integrand of Λ by scheduling on the ● and ▲ in Fig. 4.8 for a uniform distributed packet size on $[0.5 \cdot D,\ 1.5 \cdot D]$.

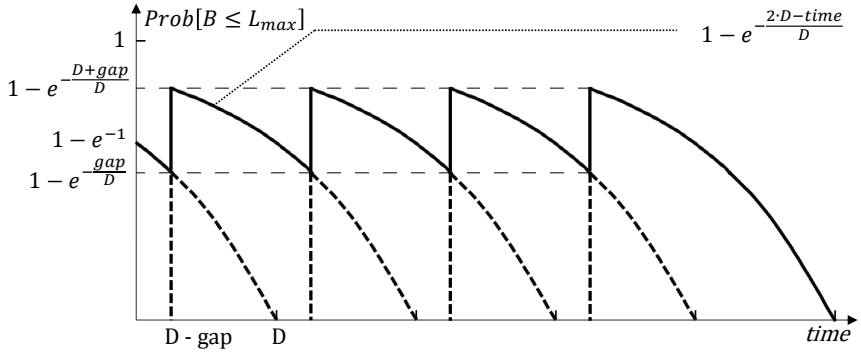


Figure 4.13: Integrand of Λ by scheduling on the ● and ▲ in Fig. 4.8 for an exponentially distributed packet size.

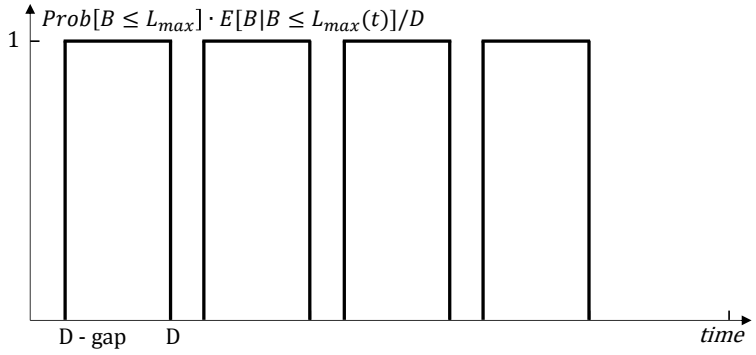


Figure 4.14: Integrand of $\bar{\Lambda}$ by scheduling on the ● and ▲ in Fig. 4.8 for a fixed packet size.

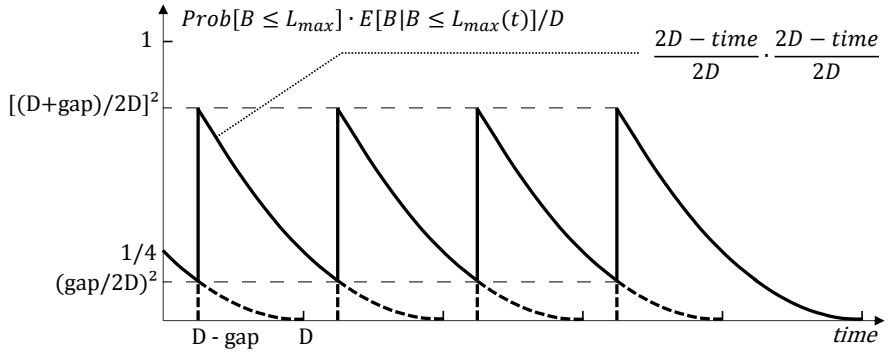


Figure 4.15: Integrand of $\bar{\Lambda}$ by scheduling on the ● and ▲ in Fig. 4.8 for a uniform distributed packet size on $[0,\ 2 \cdot D]$.

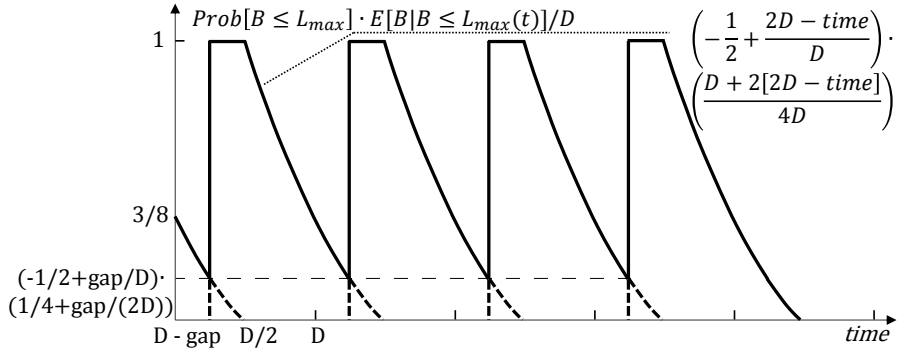


Figure 4.16: Integrand of $\bar{\Lambda}$ by scheduling on the ● and ▲ in Fig. 4.8 for a uniform distributed packet size on $[0.5 \cdot D,\ 1.5 \cdot D]$.

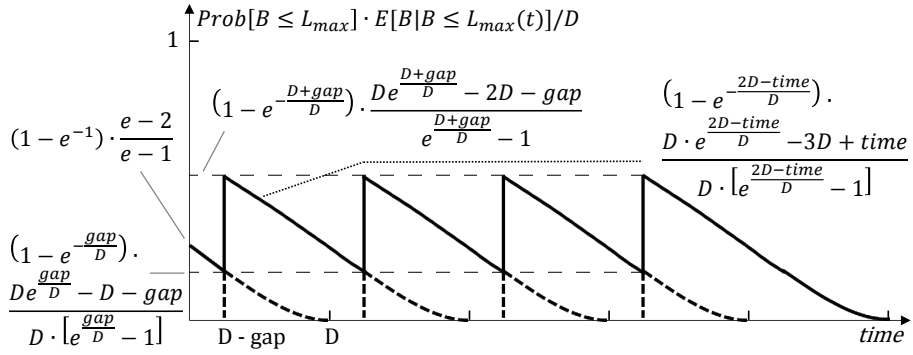


Figure 4.17: Integrand of $\bar{\Lambda}$ by scheduling on the ● and ▲ in Fig. 4.8 for an exponentially distributed packet size.

Eq. 4.1. As the cumulative distributions of our example packet distributions are known, the probability that a packet is smaller than the dashed and unbroken lines in Fig. 4.9 is calculated in a straightforward manner. For each packet distribution, this calculation transforms Fig. 4.9 to the respective Fig. 4.10 - 4.13. For the curves of Fig. 4.14 - 4.17 an additional multiplication is added in the integrand (see Eq. 4.2), resulting in different, yet often still identifiable curves for the integrand of each packet distribution on both the circle and triangle position. Moreover, the additional term in the integrand of Eq. 4.2 is once again easily calculated from the cumulative distributions.

Based on the simulation results, it is $\bar{\Lambda}$ that will used as the basis for the void value function. This is done by confronting scheduling on the circle with scheduling on the triangle in terms of a theoretical (added) value function: $\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet)$. To calculate these values, the integral of Eq. 4.2 has to be evaluated for each considered packet size distribution and both the circle and triangle positions (i.e., curves). With n denoting the horizon index, it is clear from Fig. 4.14 - 4.17 that these integrands show an n-times repeating pattern and thus result in simplified calculations which can be done by hand or desktop software. In the examples of Fig. 4.14 - 4.17, and thus Fig. 4.8, $n = 4$. To aid those readers who wish to recalculate the integrals, a function describing the first completely visible part of this pattern is shown on the Fig. 4.11 - 4.13 and Fig. 4.15 - 4.17.

With the given definition of $\bar{\Lambda}$, we can evaluate $Prob[B \leq L_{max}(t)]$ and $E[B|B \leq L_{max}(t)]$ for a variety of packet size distributions and a general value of n. A fixed packet scheduled on the circle size yields:

$$\bar{\Lambda}(\bullet) = 0.$$

When scheduling on the second FDL after the horizon (triangle) $\bar{\Lambda}$ becomes:

$$\bar{\Lambda}(\blacktriangle) = \rho \cdot n \cdot g.$$

Where g denotes gap/D and n denotes the horizon index. A uniform distribution of the packet size on the interval $[0,\ 2 \cdot D]$ results in the following $\bar{\Lambda}$:

$$\bar{\Lambda}(\bullet) = \frac{\rho}{12} \cdot n \cdot g^3.$$

$$\bar{\Lambda}(\blacktriangle) = \frac{\rho}{12} \cdot \left[n \cdot \left(1 + 3g + 3g^2\right) + 1\right].$$

For a uniform distribution of the packet size on the interval $[0.5 \cdot D,\ 1.5 \cdot D]$ there are two different cases. If $g \leq 0.5$, $\bar{\Lambda}$ becomes:

$$\bar{\Lambda}(\bullet) = 0.$$

$$\bar{\Lambda}(\blacktriangle) = \frac{\rho}{24} \cdot \left[n \cdot \left(2 + 9g + 12g^2 + 4g^3\right) + 2\right].$$

If $0.5 < g \leq 1$ $\bar{\Lambda}$ becomes:

$$\bar{\Lambda}(\bullet) = \frac{\rho}{24} \cdot n \cdot \left(1 - 3g + 4g^3\right).$$

$$\bar{\Lambda}(\blacktriangle) = \frac{\rho}{24} \cdot \left[n \cdot \left(-3 + 27g - 4g^3\right) + 2\right].$$

For an exponentially distributed packet size this $\bar{\Lambda}$ is obtained:

$$\bar{\Lambda}(\bullet) = \rho \cdot n \cdot \left[g - 2 + (2 + g) \cdot e^{-g}\right].$$

$$\bar{\Lambda}(\blacktriangle) = \rho \cdot \left[n \cdot \left(1 + e^{-g} \cdot \left(\frac{3+g}{e} - g - 2\right)\right) + \frac{3}{e} - 1\right].$$

Based on these results a void value threshold algorithm can be created for which void creation is employed if the difference in theoretical void value between scheduling on the circle and the triangle ($\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet)$) is larger than a certain threshold.

4.3.2 Performance results

4.3.2.1 Void-creating algorithm improvements

We first simulate the system without void creation for all packet size distributions, i.e., all packets are scheduled with the D-G-VF algorithm, our reference algorithm to which the performance of our void creating algorithm are compared. We assume having 10 FDLs ($N+1=10$), $D=E[B]$ and the number of simulated packets is $10\cdot 10^7$ as this yields confidence intervals narrow enough to draw reliable conclusions. The performance measures are summarized in Table 4.1 for a load of 60 % and a load of 80 %. From this table it can be seen that performance in most performance measures is better for variable packet length distributions. For the case of a fixed packet size equal to the granularity it is indeed so that no voids larger than the packet size are created and thus none can be filled. Because of this D-G-VF acts as non-void-filling algorithm in the fixed size packet case and, in general, results in a lower performance. As the void-filling mechanism favors small packets, LPsize is higher than LP for the variable sized packet distributions. This is as opposed to the fixed size packet case, for which LP and LPsize have the same values.

Table 4.1: Performance measures of D-G-VF for different packet size distributions.

D-G-VF	*Fixed B= E[B]=D*		*Exponential E[B]=D*		*Uniform on [0, 2D]*		*Uniform on [0.5D, 1.5D]*	
	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8
LP	2.1 %	14.5 %	2.6 %	9.1 %	1.9 %	9.8 %	2.2 %	12.9 %
LPsize	2.1 %	14.5 %	3.4 %	11.9 %	2.3 %	11.7 %	2.3 %	13.4 %
Packet delay	3.0	6.1	2.5	4.1	2.7	4.8	3.0	5.7
Packet gap	0.4	0.4	0.3	0.3	0.3	0.3	0.4	0.4

The improvement in LP (compared to D-G-VF) obtained by the void creating algorithm for a load of 60 % and a fixed packet size is shown in Table 4.2. In this table the columns represent the gap range and the rows the horizon index for which void creation is allowed. Improvements (i.e., a lower LP) are shown in black font, whereas combinations of horizon index and gap

ranges for which no improvement are obtained (i.e., a higher LP) are shown in white font. For each (gap range, horizon index) combination, the brighter the boxes, the more performance is improved. For example when void creation is allowed if and only if the horizon index equals six and the gap is between $0.6 \cdot D$ and $0.7 \cdot D$, an improvement of 3.4 % in LP is obtained.

Table 4.2: Performance improvement in LP obtained by the void-creating algorithm for a load of 60 % and a fixed packet size.

LP gain (%); load=0.6										
	Gap range									
Horizon index	*]0-0.1]*	*]0.1-0.2]*	*]0.2-0.3]*	*]0.3-0.4]*	*]0.4-0.5]*	*]0.5-0.6]*	*]0.6-0.7]*	*]0.7-0.8]*	*]0.8-0.9]*	*]0.9-1]*
1	-6.8	-5.4	-4.5	-3.3	-2.4	-2.0	-1.6	-1.2	-0.9	-0.8
2	-8.1	-5.6	-3.6	-2.3	-1.2	-0.1	0.2	0.4	0.4	1.0
3	-8.8	-5.3	-2.8	-0.9	0.3	1.2	1.3	1.5	1.4	1.5
4	-9.3	-4.7	-1.6	0.1	1.2	2.2	2.3	2.2	2.3	1.9
5	-9.7	-4.4	-1.1	1.2	2.5	2.7	3.0	3.1	2.9	2.6
6	-10.3	-3.9	-0.2	1.8	3.0	3.0	3.4	3.3	3.2	2.6
7	-10.6	-4.1	-0.2	1.9	2.9	3.5	3.5	3.2	3.1	2.9
8	-11.1	-4.7	-1.2	0.9	2.0	2.4	2.7	2.8	3.1	2.7

As some combinations of gap range and horizon index occur more often than others, we incorporated this consideration in the performance evaluation. Correspondingly, the percentages in Table 4.2 represent the improvement all void creations collectively produce rather than the improvement a single void creation produces on that position. Additionally, we gather information in a second table, Table 4.3, in which all improvements are divided by the number of packets that created that improvement, i.e., as a percentage of all packets that arrived. Table 4.3 is obtained for the same packet size distribution, performance measure and load as in Table 4.2. The result is a table in which the numbers no longer represent a true improvement (expressible in percentages) but do show the relative improvement obtained by a single void creation for that combination of horizon index and gap range. Similar tables as Table 4.2 and Table 4.3 can be made for different packet size distributions, performance measures and loads but are not shown. The most important observations and outcomes are discussed below.

Table 4.3: Performance improvement in LP per packet obtained by the void-creating algorithm for a load of 60 % and a fixed packet size.

LP gain per packet; load=0.6										
	Gap range									
	]0-0.1]	*]0.1-0.2]*	*]0.2-0.3]*	*]0.3-0.4]*	*]0.4-0.5]*	*]0.5-0.6]*	*]0.6-0.7]*	*]0.7-0.8]*	*]0.8-0.9]*	*]0.9-1]*
Horizon index 1	-2.5	-2.1	-1.8	-1.4	-1.1	-1.0	-0.8	-0.6	-0.6	-0.5
2	-3.5	-2.6	-1.8	-1.2	-0.7	-0.1	0.1	0.3	0.3	0.7
3	-4.7	-3.0	-1.7	-0.6	0.2	0.8	1.0	1.2	1.2	1.4
4	-6.0	-3.3	-1.2	0.1	1.0	1.9	2.1	2.2	2.5	2.2
5	-7.7	-3.7	-0.9	1.1	2.5	2.9	3.4	3.7	3.7	3.6
6	-9.9	-3.9	-0.2	2.1	3.7	3.9	4.7	4.8	5.0	4.2
7	-12.3	-5.1	-0.2	2.7	4.3	5.5	5.9	5.8	5.8	5.9
8	-15.7	-7.1	-2.0	1.5	3.6	4.6	5.4	6.2	7.1	6.7

Table 4.4 shows the largest actual improvements the void creating algorithm can achieve for every performance measure, packet size distribution and load when void creation is allowed on only a single combination of horizon index and gap range. As an unreasonably long simulation time is needed, the achievable improvements when void creation is allowed on the optimal set of horizon indexes and gap ranges are not determined. Similar to the results in Section 4.2 however, these performance improvements are expected to be a multitude of the largest obtainable improvement for a single combination. For example, whereas for a single combination and a fixed packet size an improvement of only 3.5 % for a load of 60 % and 2.7 % for a load of 80 % in LP are obtainable, the void creating algorithm in Section 4.2 achieves an improvement of 50 % for a load of 60 % and 32 % for a load of 80 % when void creation is allowed on the optimal set of combinations. Similar enhancements are expected for other packet size distributions and performance measures. Moreover we assume the highest values achievable by a single combination are a good indication of the improvement achievable when void creation is allowed on the optimal set of combinations. From Table 4.4 we can see that not for all packet size distributions a performance improvement can be achieved. More specifically for the exponentially distributed packet size there are no improvements possible for either LPsize or packet gap. Roughly speaking, the larger the variance of the arriving packet distribution, the less improvement

is achievable in any performance measure. As the variance of the arriving packet distribution is larger the arrival process is less deterministic and the outcome of a void creation is thus less predictable.

Table 4.4: Largest achievable improvement of the void-creating algorithm for a single combination of horizon index and gap range.

maximum gain	*Fixed B= E[B]=D*		*Exponential E[B]=D*		*Uniform on [0, 2D]*		*Uniform on [0.5D, 1.5D]*	
	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8
LP	3.5 %	2.7 %	0.9 %	1.1 %	1.1 %	1.1 %	2.4 %	1.9 %
LPsize	3.5 %	2.7 %	-0.5 %	-0.1 %	0.7 %	0.4 %	2.2 %	1.4 %
Packet delay	1.1 %	1.3 %	0.2 %	0.4 %	0.3 %	0.6 %	0.6 %	0.7 %
Packet gap	0.8 %	1.7 %	-0.1 %	-0.1 %	0.1 %	0.3 %	0.4 %	1.0 %

4.3.2.2 Predictive power of the theoretic void values

In general, optimizing for delay, i.e., allowing void creation for the combinations of horizon index and gap threshold that greatly reduce the average packet delay, severely limits the improvements achieved in LP and LPsize. This is because increasing the chance a packet is lost when the horizon index is high, results in less packets with a high delay being accepted in the system and thus reduces the overall average packet delay. Because the FDL buffer is finite, the void-creating algorithm optimized for delay will thus attempt to increase the chance packets are lost when the horizon index is high. Similarly, optimizing for LP and LPsize will undermine the average packet delay, accommodating an extra packet at all cost. Optimizing for average packet gap harmonizes these conflicting performance measures. As opposed to scheduling optimized for delay, there is no situation in which there is an incentive to try to lose the next arrival. Intuitively it is also clear that minimizing the gaps between packets makes the overall provisional schedule dense and compact, resulting in both a low average packet delay and a reduced packet loss. Therefore, in this section we choose the average packet gap, defined as the length of the unscheduled period preceding the packet, as our performance measure.

In Fig. 4.18 the actual gap improvements per packet (i.e., values obtained from simulation) for a fixed packet size are plotted as a function of the difference in theoretical void values between scheduling on the triangle and the circle. Hereby, theoretical void value $\bar{\Lambda}$ from Section 4.3.1 is used, as this gives a far better correlation than Λ for all packet size distributions and loads $(\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet))$. All combinations of the horizon index and gap range are shown in Fig. 4.18. The gap ranges have a length of $0.1 \cdot D$ and the value of $g = gap/D$ used in the formulas of $\bar{\Lambda}$ is set equal to the middle of each gap range. From Fig. 4.18 it is clear that the actual gap improvements per packet can be estimated based on our theoretical void values, i.e., a curve with high coefficient of determination (R^2) could be drawn through the data points. For other packet size distributions similarly shaped curves are obtained. The exact deflection of the curve still depends on the packet size distribution but is fairly indifferent to the value of the load once this packet size distribution is known. The surplus in theoretical void value $(\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet))$ for which void creation is beneficial rises as the variance of the packet size distribution increases.

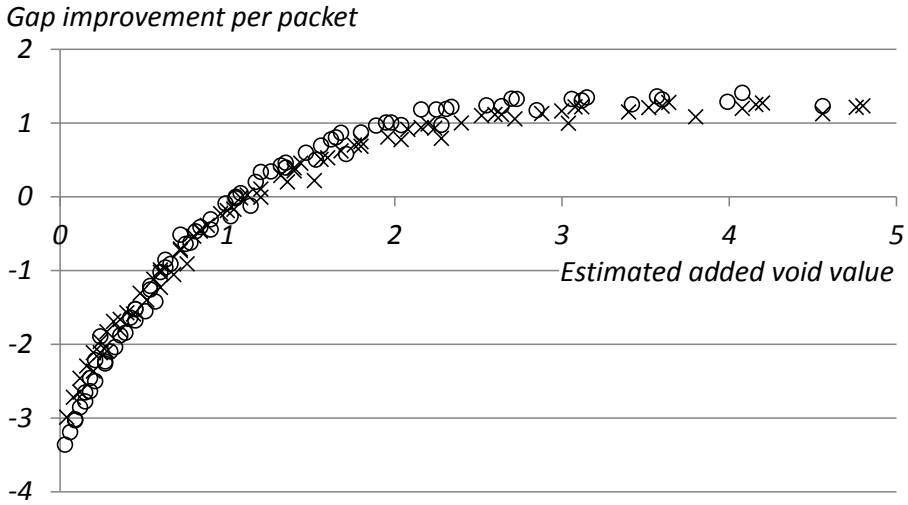


Figure 4.18: Simulated gap improvement per packet as a function of the estimated added void value of void creation for a fixed packet size distribution. ○: load=0.6. ×: load=0.8.

Based on these results a void value threshold algorithm can be created for which void creation is employed if the difference in theoretical void value between scheduling on the circle and the triangle $(\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet))$ is larger than a certain threshold. Table 4.5 shows the optimal thresholds and per-

formance improvement (both obtained by simulation) for each combination of performance measure, load and packet size distribution. The obtainable improvements for the fixed packet size from Table 4.5 are (up to 5 %) larger than the improvements obtained in Section 4.2. The void value threshold algorithm, as opposed to the algorithm from Section 4.2, can thus be used for different packet size distributions and moreover outperforms the algorithm from Section 4.2 under the same circumstances. From Table 4.5 it is clear however that obtainable performance improvement strongly depends on the packet size distribution and decreases as the variance of the packet size distribution increases. As the optimal threshold depends on (the variance of) the packet size distribution this results in a basic scheduling algorithm that is as simple to implement as a classic scheduling algorithm, the only difference being that it is tuned beforehand in order to better handle traffic of packets with a certain packet size distribution and a certain load. Hereby, note that an algorithm optimized for one packet size distribution may also yield significant performance improvements for a second or third packet size distribution, but in rare cases, also the opposite may hold true.

4.4 Fixed packet size on multiple wavelengths

4.4.1 Approach and concept: void values

To extend the analysis to multiple wavelengths a setting with fixed size packets $B = D$ is considered. Although the D-G algorithm was the benchmark algorithm in the single wavelength and fixed packet size setting this has to be reassessed for the multiple wavelength setting. More specifically, as multiple wavelengths are present, scheduling according to an algorithm that focuses on the gaps in between packets can significantly improve performance. However, as in the fixed size packets setting no fillable voids are created, the algorithm will only have to choose which wavelength to join at the back. With this reduced set of options, all gap focused algorithms will schedule an arriving packet at the back of the wavelength resulting in the smallest starting void. We therefore choose the G-D (gap-delay non-void-filling) algorithm as the benchmark algorithm to which performance of our void-creating algorithm will be compared. This G-D algorithm was also discussed in Chapter 2 and is equivalent to most variations of the min-SV and LAUC algorithms used in other literature [132].

As our reference algorithm for this setting is the G-D algorithm, our void-creating algorithm will be based on this algorithm as well. The choices and

optimal threshold	*Fixed B= E[B]=D*		*Exponential E[B]=D*		*Uniform on [0, 2D]*		*Uniform on [0.5D, 1.5D]*	
	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8
LP	1.0	1.2	1.1	1.4	1.4	1.6	1.2	1.5
LPsize	1.0	1.2	2.3	3.0	1.9	2.0	1.3	1.6
Packet delay	1.0	1.2	0.9	0.9	1.9	2.0	1.3	1.6
Packet gap	1.2	1.3	3.0	2.3	1.8	2.0	1.5	1.6

(a) Optimal thresholds

maximum gain	*Fixed B= E[B]=D*		*Exponential E[B]=D*		*Uniform on [0, 2D]*		*Uniform on [0.5D, 1.5D]*	
	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8	ρ=0.6	ρ=0.8
LP	54.1 %	36.1 %	4.5 %	6.4 %	5.7 %	6.9 %	28.7 %	19.4 %
LPsize	54.1 %	36.1 %	0.2 %	0.1 %	0.6 %	1.9 %	25.0 %	16.5 %
Packet delay	16.3 %	16.2 %	0.8 %	3.4 %	1.7 %	3.9 %	8.2 %	8.8 %
Packet gap	11.5 %	22.8 %	0.0 %	0.0 %	0.3 %	1.3 %	4.5 %	10.3 %

(b) Corresponding performance improvements

Table 4.5: Optimal thresholds and corresponding performance improvements of the void value threshold algorithm for a single wavelength setting.

order in which this void-creating algorithm operates are done in the most logical and simple way. Although different choices can be made, they will, at least in this setting, make little difference in the overall obtainable performance improvement.

A first difference for the void-creating algorithm compared to the G-D algorithm is that fillable voids, created by the void creating mechanism upon previous arrivals, can be present. When a single or multiple fillable voids are present, the void-creating algorithm chooses to fill the void that results in the lowest delay for the packet. This procedure is sustained even if a lower gap can be obtained by filling a different void or by joining a wavelength at

the back. The reasoning behind this is that an artificially created void is only advantageous when it is actually filled afterwards. Moreover, as these voids only have a limited lifetime, the void set to expire the quickest (corresponding to the lowest delay) should get filled first.

When no fillable voids are present, and the algorithm thus has to decide which wavelength to join at the back, the void creating algorithm, just like the G-D algorithm, chooses to schedule on the wavelength resulting in the smallest gap. Once the wavelength has been chosen based on this criterion, the void creating algorithm now decides upon void creation based on the void value function. Just like in the setting of a single wavelength this value function is based on the overall normalized expected packet size to arrive in the system that is able to fill the void ($\bar{\Lambda}$). Hereby, the theoretical value of scheduling on the circle is confronted with that of scheduling on the triangle in terms of a theoretical (added) value function: $\bar{\Lambda}(\blacktriangle) - \bar{\Lambda}(\bullet)$. If this difference in theoretical void value between scheduling on the circle and triangle is larger than a certain threshold, the algorithm decides to create a larger void by scheduling on the triangle position. In the example of Fig. 4.19 void creation thus corresponds to scheduling on position c instead of position b when this is triggered by the value function. Note that in this case the void-creating position c is also preferred over both positions d and e with the same delay.

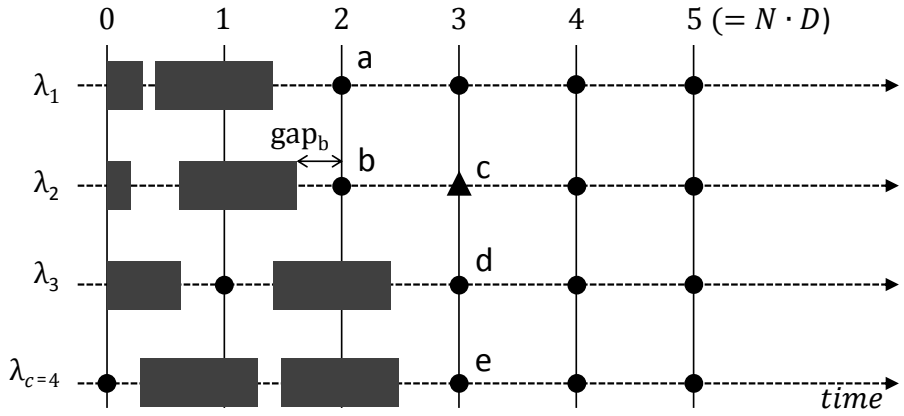


Figure 4.19: An example of a provisional schedule for fixed packet size ($B = D$) and multiple wavelengths ($c = 4$) resulting in a possible void creation on the second wavelength.

By using the same value function as in the single wavelength setting, the void creating algorithm does not take into account any other voids already

present in the system, also, and especially, not those on other wavelengths. Extending the algorithm to take into account the value of other voids could be useful but should be done wisely. Indeed, with the threshold value the algorithm still has a parameter that can tune the algorithm to be more or less strict in creating voids, possibly resulting in a shift of the optimal threshold parameter when values of other wavelengths are taken into account in the value function.

4.4.2 Performance results

Similar to the single wavelength case we first simulate the system without void creation, i.e., all packets are scheduled with the G-D algorithm as defined above, our reference algorithm to which the performance of our void creating algorithm will be compared. We assume 10 FDLs ($N + 1 = 10$), $D = E[B] = B$ and the number of simulated packets is $10 \cdot 10^7$ as this yields confidence intervals narrow enough to draw reliable conclusions. Both the number of wavelengths (c =1, 2, 4, 6 and 8) and the load are varied (ρ =60 %, 80 % and 90 %) in simulations. As even lower values of the load and a higher number of wavelengths would require an excessive number of simulated packets and accompanying long simulation time, the simulated values are limited to combinations of these sets. For all combinations of these values the performance measures (LP, LPsize, delay, gap) are determined. Table 4.6 shows the results for all performance measures and combinations of parameters (load and number of wavelengths) for the G-D algorithm. In Table 4.6 an "x" indicates whether too few packets (or even none at all) are lost, resulting in unreliable and thus omitted results for this parameter combination.

Similar as Table 4.5 does for a single wavelength setting, Table 4.7 and 4.8 shows the optimal thresholds and performance improvements (both obtained by simulation) for each performance measure, load and number of wavelengths. Note that rather than actual performance measure values, Table 4.8 thus shows performance improvements relative to the performance measure values from Table 4.6. Whereas in Table 4.6 it is natural that performance measures decrease with an increasing number of wavelengths and decreasing load, this, although observed, is not necessarily true for the values in Table 4.8 as they are mere relative performance improvements. Similarly as in Table 4.6 an "x" indicates when too few packets (or even none at all) are lost, resulting in unreliable and thus omitted results for the LP and LPsize values. An ∞ symbol in Table 4.7 (corresponding to a 0 % obtainable performance improvement) on the other hand marks the cases when no improvement can

G-D	c=1	c=2	c=4	c=6	c=8
LP (%)	2.1	0.0	x	x	x
LPsize (%)	2.1	0.0	x	x	x
Packet delay	3.0	1.0	0.4	0.2	0.1
Packet gap	0.36	0.13	0.03	0.01	0.01

(a) load=0.6

G-D	c=1	c=2	c=4	c=6	c=8
LP (%)	14.5	4.5	0.1	0.0	x
LPsize (%)	14.5	4.5	0.1	0.0	x
Packet delay	6.1	4.9	1.9	1.1	0.8
Packet gap	0.42	0.25	0.08	0.04	0.02

(b) load=0.8

G-D	c=1	c=2	c=4	c=6	c=8
LP (%)	22.5	13.8	5.7	1.7	0.3
LPsize (%)	22.5	13.8	5.7	1.7	0.3
Packet delay	7.1	7.1	6.2	4.5	2.8
Packet gap	0.42	0.28	0.16	0.09	0.05

(c) load=0.9

Table 4.6: Performance measures of G-D for a fixed packet size distribution and a varying number of wavelengths ($c = 1, 2, 4, 6$ and 8).

be achieved by the void creating algorithm. This results in never applying void creation and thus an infinitely high value threshold in the void-creating value threshold algorithm.

From Table 4.7 it is clear that the optimal threshold decreases as the number of wavelengths (c) increases. This continues up until a certain number of

optimal threshold	c=1	c=2	c=4	c=6	c=8
LP	0.9	0.7	x	x	x
LPsize	0.9	0.7	x	x	x
Packet delay	0.9	∞	∞	∞	∞
Packet gap	1.2	0.8	∞	∞	∞

(a) load=0.6

optimal threshold	c=1	c=2	c=4	c=6	c=8
LP	1.2	1.0	0.8	2.3	x
LPsize	1.2	1.0	0.8	2.3	x
Packet delay	1.2	1.0	∞	∞	∞
Packet gap	1.3	0.9	0.7	∞	∞

(b) load=0.8

optimal threshold	c=1	c=2	c=4	c=6	c=8
LP	1.3	0.9	0.7	0.5	1.4
LPsize	1.3	0.9	0.7	0.5	1.4
Packet delay	1.2	0.9	0.7	1.0	2.4
Packet gap	1.3	0.9	0.7	0.5	0.5

(c) load=0.9

Table 4.7: Optimal thresholds for the void value threshold algorithm for a fixed packet size distribution and a varying number of wavelengths ($c = 1, 2, 4, 6$ and 8).

wavelengths, at which point no performance improvement is achieved by the void-creating algorithm and the optimal threshold thus jumps to infinity. This is also clear in the graphs of Fig. 4.20, 4.21 and 4.22 which show the maximum gain (i.e., reduction giving rise to performance gain) in performance improvement in relation to the number of wavelengths (c) and the

maximum gain (%)	*c=1*	*c=2*	*c=4*	*c=6*	*c=8*
LP	53.5	42.8	x	x	x
LPsize	53.5	42.8	x	x	x
Packet delay	16.1	0	0	0	0
Packet gap	11.4	3.4	0	0	0

(a) load=0.6

maximum gain (%)	*c=1*	*c=2*	*c=4*	*c=6*	*c=8*
LP	35.9	40.6	19.6	27.5	x
LPsize	35.9	40.6	19.6	27.5	x
Packet delay	16.1	13.6	0	0	0
Packet gap	22.7	19.8	4.9	0	0

(b) load=0.8

maximum gain (%)	*c=1*	*c=2*	*c=4*	*c=6*	*c=8*
LP	26.0	25.3	21.6	12.6	0
LPsize	26.0	25.3	21.6	12.6	0
Packet delay	12.9	11.1	8.1	3.2	0.1
Packet gap	24.8	20.8	14.1	7.8	2.0

(c) load=0.9

Table 4.8: Optimal performance improvements of the void value threshold algorithm for a fixed packet size distribution and a varying number of wavelengths ($c = 1, 2, 4, 6$ and 8).

load. Although not a direct consequence from the changing parameters, the obtainable performance improvements in general do tend to decrease with an increasing number of wavelengths and lower load. This decrease however is not always monotonically as for some performance measures and values of the load an intermediate maximum in obtainable performance improvement can be observed.

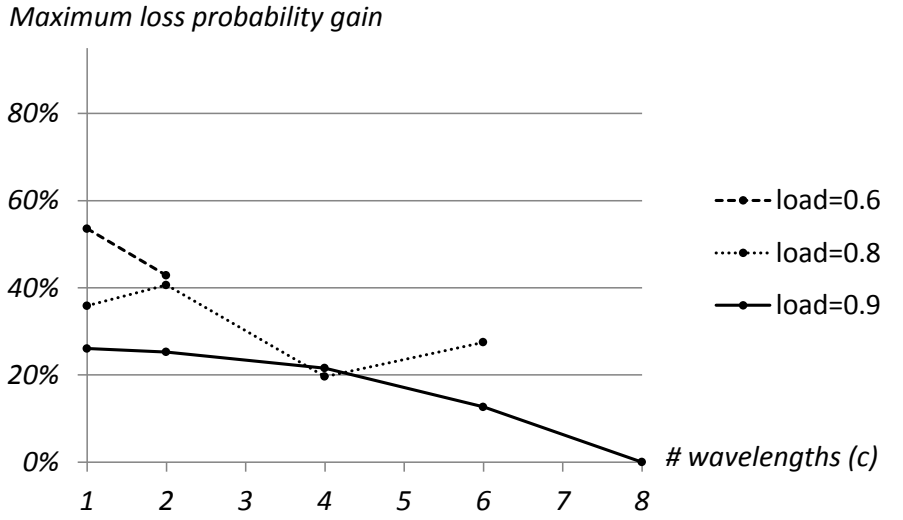


Figure 4.20: Maximum gain for loss probability as a function of the number of wavelengths (c).

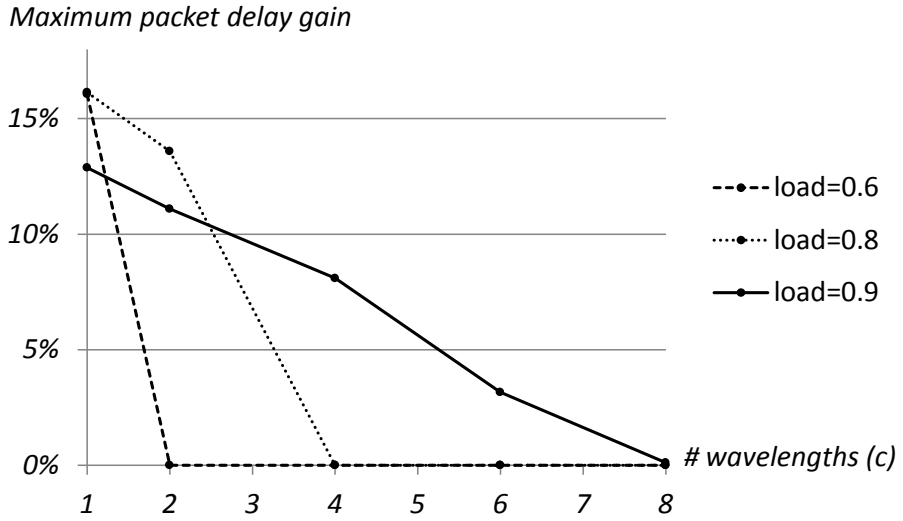


Figure 4.21: Maximum gain for average packet delay as a function of the number of wavelengths (c).

From these results it is clear that the void-creating algorithm, using the same threshold based method, can indeed be applied on multiple wavelength settings for a fixed packet size. The resulting performance improvements, which are more than significant, highlight that a basic extension of the algorithm already leads to significant performance gain. Although gap focused algo-

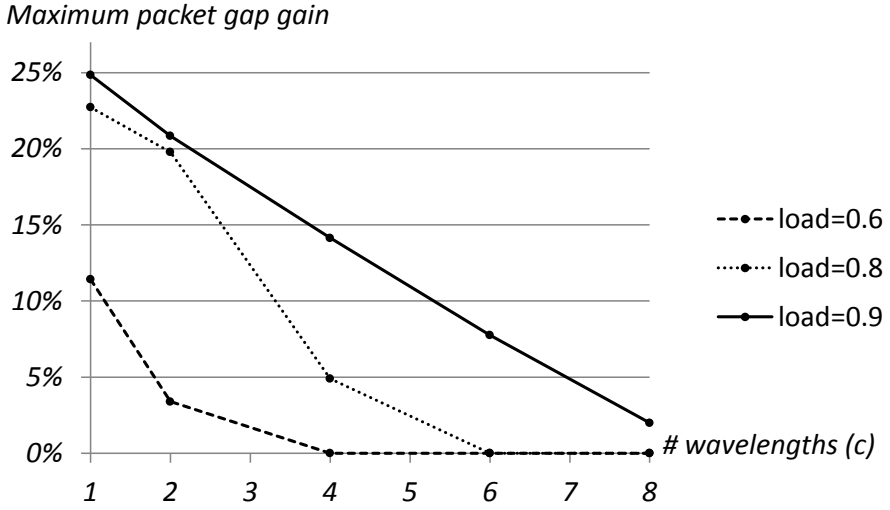


Figure 4.22: Maximum gain for packet gap as a function of the number of wavelengths (c).

rithms as G-D operate fundamentally different than delay-based algorithms in this setting, our void creating algorithm still manages to improve performance in the same order of magnitude. This clearly shows that void creation is not a competing mechanism for more advanced void filling and can thus be used mostly in a complementary manner, to improve the performance of a variety of algorithms. In this setting the performance of this algorithm however does tend to decrease with an increasing number of wavelengths and decreasing load. In the settings that were studied, performance improvement seemed to decrease along with the (absolute) value of the performance measure considered.

4.5 Methodology

Similar to the cost-based algorithms of Chapters 2 and 3, the performance of the void-creating algorithms is evaluated by means of Monte Carlo simulation and discrete event simulation (DES). Although there are some differences concerning the exact implementation, these are in general minor in nature. We therefore choose to not include pseudo-codes for the void-creating algorithms but rather focus on the most important aspects and those aspects distinctive to void-creating algorithms below.

Similar as for other algorithms, the events list consists of the arrivals of packets. In each iteration the arrival of a packet is simulated, i.e., the inter-arrival time (T_{packet}) (the time between the current packet and the next), the packet length of the current packet (B_{packet}) and the wavelength w ($w = 1 \ldots c$) on which it arrives, are generated using random number generators. In the case of a single wavelength, as in Sections 4.2 and 4.3, or a fixed packet size, as in Sections 4.2 and 4.4, the wavelength and/or packet size are fixed and no random choice has to be made.

Although different algorithm parameters than those in Chapter 2 and 3 are optimized, the used methodology, in which algorithm parameters that need to be optimized are varied in different simulation runs, is similar. This is particularly the case for the algorithms in Section 4.3 and 4.4, in which the optimal void-value thresholds for the different packet size distributions are determined by varying them in different simulation runs. In Section 4.2 a slightly different iterative approach is used to optimize the gap thresholds. In a first and auxiliary set of simulations we determine the optimal gap threshold when voids are only allowed to be created on a single horizon index. This is done separately for each value of n ($n = 1 \ldots N-1$) and gives us two sets (one for LP and one for average packet delay) of both $N-1$ optimal gap threshold values for which either LP or average packet delay is minimized. Starting from these so-called single creation point thresholds, the optimal gap thresholds when void creation is allowed on all horizon indexes are determined. To do this, all gap thresholds but one are fixed to the optimal single creation point thresholds. The one gap threshold that is not fixed is varied and optimized ($0 < y_n < 1$ in steps of 0.01).

Consecutively this is done once for each horizon index ($n = 1 \ldots N-1$) in a different simulation trace. In this way the gap threshold of each horizon index is optimized, with the other thresholds fixed. This gives us a set of $N-1$ new optimal gap thresholds which are called the first iteration thresholds. In a second iteration we fix all gap thresholds but one to these first iteration thresholds and again optimize the one that is not fixed. Again only the gap threshold that is optimized is varied ($n = 1 \ldots N-1$). This results in a set of $N-1$ second iteration thresholds. These iterations can be repeated as many times as desired until an acceptable convergence is noticed in the iteration thresholds and their corresponding performance, marking a good approximation of the actual void-creating algorithm. In general only a few (2-3) iterations were necessary to obtain a complete convergence of the thresholds in the examples of this chapter.

As either full wavelength conversion is assumed or the analysis can be confined to a single wavelength (because no WCs are present), the system state only consists of a single part. For the void-creating algorithm of Section 4.2 the system state includes the value of the horizon and the beginning and ending of the single fillable void (if present) that is allowed in the system. Indeed, as the packet size is fixed, only voids larger than the fixed packet size $B = D$ have to be tracked. Although limited in size, the system state is however not as confined as that of classic non-void-filling algorithms, that only have to keep track of the horizon. In the multiple wavelength setting of Section 4.4, more than one fillable void, even on a single wavelength, is allowed resulting in a larger system state. The void-creating algorithm of Section 4.3, designed for a single wavelength and general packet size, keeps track of all voids larger than the smallest packet size of the given packet size distribution. As a consequence the system state consists of a two-dimensional void matrix similar to that of classic void-filling algorithms. Void-filling algorithms (or void-creating algorithms with an equally complex system state representation) are computationally more complex and more difficult to implement in a switch than the non-void-filling algorithms. Within both classes of system state complexity, however, no significant difference exists in complexity of implementation. This is important as we can thus say that any improvement achieved within the same class of system state complexity is achieved without complicating implementation.

Table 4.9: Comparison of simulation times of different algorithms under equal conditions.

Type of algorithm	Average simulation time for 10^4 arrivals (s)
Non-void-filling	0.1921
Void-creating	0.2974
Void-filling	3.8644

To compare the implementation complexity of void-creating algorithms the arrival of $10 \cdot 10^4$ packets was simulated in [90] for the fixed size packet and single wavelength setting of Section 4.2. This was done using stripped-down versions of the corresponding void-creating algorithm, i.e., without keeping track of any performance measures as LP or average packet delay. The algorithms thus only decide on which FDL each packet is scheduled, just like a switch-level implementation would. All simulations were carried out on the same PC (Intel Core i7, CPU @ 2.40 GHz) under the same circumstances

and with all algorithm parameters equal (i.e., the parameters used for Fig. 4.4-4.5, load=0.8, second iteration). Table 4.9 clearly shows that keeping track of all the voids in the void-filling algorithm is, apart from useless in the fixed size packet setting (since no void can be filled), also very costly in terms of computation. The void-creating algorithm on the other hand is nearly as fast as the non-void-filling algorithm as it only keeps track of one extra void besides the horizon.

4.6 Conclusions

In this chapter we proposed a new type of algorithm called the void-creating algorithm which, based on the current system state, decides to create larger voids than strictly necessary. Different implementations of the void-creating mechanism were proposed for fixed packet size on a single wavelength, general packet size on a single wavelength and fixed packet size on multiple wavelengths.

For the setting of a fixed packet size on a single wavelength the algorithm decides upon void creation based on the gap size. By means of an iterative simulation method the optimal gap thresholds for different horizon values are determined. Depending on the position of the horizon, the condition on the gap size can thus either be more or less strict. Improvements in loss probability of up to 30 % for a high load (80 %) and 50 % for a lower load (60 %) are achievable. The achievable improvements for average packet delay are smaller with an achievable reduction of 14 % for a load of 80 % and 18 % for a load of 60 %. The optimal gap threshold values for which these improvements in loss probability and average packet delay are achieved, though not identical, are in close proximity. The optimal gap threshold are lower for higher loads and for higher horizon values. This is because for lower loads and for lower horizon values the number of expected packets to arrive, before the void disappears, is lower. This lack of high arrival density thus has to be compensated by a larger size of the created void, which corresponds with higher gap thresholds.

For the setting of a general packet size on single wavelength the algorithm decides upon void creation based on the added void value. The theoretical value assigned to a void is directly linked to the chance of the void being filled in the future and thus its ability to improve performance. It was shown that our choice of theoretical void values makes it possible to decide upon void creation once the packet size distribution, performance measure and load are

known. The result is a scheduling algorithm that is as simple to implement as a classic scheduling algorithm, deciding upon void creation using a closed-form expression derived from the packet size distribution. The void value threshold algorithm outperforms existing void-filling and, above all, can be used for different packet size distributions. The achievable improvement can reach tens of percentages but does depend on the packet size distribution and, in general, decreases as the variance of the packet size distribution increases.

For the setting of a fixed packet size on multiple wavelengths, the method based on the added void value is extended with only small modifications. Particularly, the adapted algorithm performs very well for a modest number of wavelengths like 2 or 4. Although gap focused algorithms, such as the reference algorithm for this setting G-D, operate fundamentally differently than delay-based algorithms, which were the reference algorithms in the previous settings, our void creating algorithm still manages to improve performance in the same order of magnitude. This clearly shows that void creation is not a competing mechanism for more advanced void filling and can thus be used mostly in a complementary manner, to improve the performance of a variety of algorithms.

Chapter 5

Forward looking scheduling algorithms for fiber loop buffers

Space travel is bunk.

Sir Harold Spencer Jones, Astronomer Royal of the UK, 1957
(two weeks later Sputnik orbited the Earth)

Throughout Chapters 2, 3 and 4 we focused on developing algorithms for Fiber Delay Line (FDL) buffers. As mentioned in Chapter 1 fiber loop buffers are considered a viable alternative as they have a smaller footprint and bigger flexibility. In this chapter we therefore focus on new algorithms for fiber loop buffers. Although constrained to a completely different setting, these algorithms will use similar tactics as the proposed scheduling algorithms for FDL buffers. More specifically, by selectively delaying packets longer than strictly necessary, our proposed algorithms try to take into account future packet arrivals and as such outperform existing scheduling algorithms.

5.1 System Model

5.1.1 Fiber loop buffer recapitulated

One of the most compact implementations of optical buffering today is a fiber loop buffer. As opposed to feed-forward FDL buffers where every line is traversed only once, fiber loop buffers allow contending packets to recirculate multiple times within the same coiled fiber loop. Although alternative

designs as dual-loop optical buffers exist [133–136], most use a set of single fiber loops in parallel which can all accommodate a single packet at once (shown in Fig. 5.1). Because packets can only exit a loop after a discrete number of recirculations, fiber loops can only provide a discrete set of delays. As opposed to electronic memory (RAM), packets can thus not be retrieved at will, resulting in small time gaps or voids in between packets on the outgoing line. Moreover, as packets recirculate in the same loop, fiber loops can only accommodate packet sizes smaller than or equal to their loop length and packet length directly limits the resolution of possible delays. Since the footprint is preferably kept small, with a low number of fiber loops, and also the number of recirculations a packet can make in a loop is kept low to prevent signal degeneration, scheduling algorithms in which the resources are used as efficiently as possible are needed to achieve low packet loss and average packet delay.

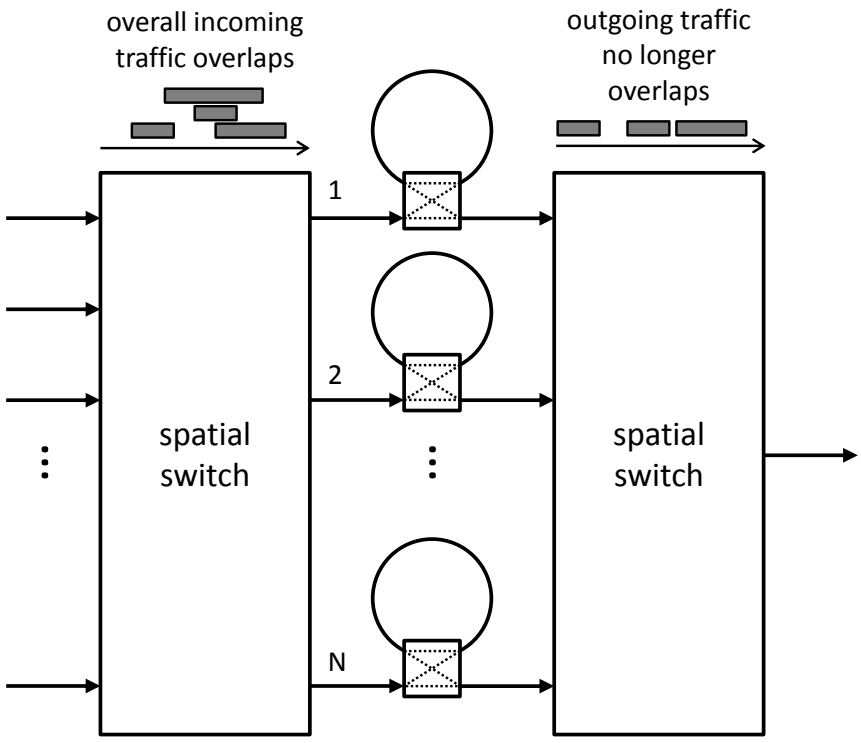


Figure 5.1: Parallel optical fiber loop buffer to resolve contention at the input.

In [137] an analytical model is used to evaluate performance of the void-avoiding schedule (VAS) for fixed size packets equal to the loop length. The void-avoiding schedule is a post-reservation scheme [115], allowing the packets to enter the buffer freely, only deciding later when a packet has to exit its loop. In [137] it is shown that performance of the VAS is significantly better than that of algorithms with a pre-reservation scheme, e.g., FCFS, in which the number of recirculations is decided upon arrival of a packet.

In this chapter we evaluate the performance of the void-avoiding schedule (VAS) for uniformly distributed packet size in both unlimited and constricted fiber loop buffer settings. We also propose two new algorithms, WAS and XAS, which, to the best of our knowledge, are the first known algorithms to outperform VAS. Particularly, both yield significant improvement for variable-length packet size, as discussed further below.

5.1.2 Assumptions

Throughout this chapter the same continuous-time setting as in the previous chapters is assumed. The fiber loop buffer is assumed to be located at and dedicated to a single outgoing port of an optical switch. Wavelength converters, if present within the switch, are assumed to perform conversion to a single outgoing wavelength associated with this single outgoing port. The analysis can thus be limited to a single wavelength. We assume the joint packet arrival at the output port on this single wavelength is a Poisson process, i.e., the inter-arrival times T are exponentially distributed with an average of $E[T]$. The length of arriving packets, B, is assumed to be uniformly distributed on the interval $[0,\ S]$ with an average of $E[B] = S/2$. Related, the overall incoming traffic load at the output port is given by $\rho = E[B]/E[T] = S/(2 \cdot E[T])$.

Because of the nature of the arrival process it is possible that different arrivals overlap upon their arrival, at which instant one of the contending packets has to be temporarily buffered in one of the fiber loops. We assume a set of parallel fiber loops of length S, that indepent of the packet's size, each can accommodate a single packet. The delays assignable to a packet are thus multiples of S. The number of fiber loops and the maximum number of times a packet can recirculate both are varied in simulations. Combinations of both finite (4, 8, and 16) and infinite values for these buffer parameters are evaluated.

5.2 WAS and XAS scheduling algorithms

5.2.1 Approach and concept

As in a buffer loop setting, the well-known FCFS algorithm is outperformed by the void avoiding schedule (VAS), we choose the latter as our benchmark algorithm. In VAS, packets that arrive are transmitted immediately if the outgoing line is available and else are stored in a fiber loop. After each loop recirculation, the availability of the outgoing line is checked. If the outgoing line is available upon such a check, the packet exits its fiber loop and is sent. If the outgoing line is not available, the packet is recirculated again and the procedure is repeated. VAS does not preserve the arrival order and is a post-reservation algorithm.

In terms of resource usage, VAS is a greedy algorithm, sending a packet whenever a transmission opportunity, i.e., an available outgoing line and either an arrival or a finished recirculation, arises. The newly proposed WAS algorithm, where "W" may refer to the aim of minimizing the **W**ait for the outgoing line to turn available after departure of the two packets, is more considerate and well aware of the other packets present in the system. When a transmission opportunity arises, WAS will first calculate the time needed to send each combination of two packets (packets i and j, $i \neq j$) present in the system. When $N+1$ packets are present in the system (i.e., packets which are either in the buffer or a new arrival), this gives an $(N+1) \times (N+1)$ two dimensional matrix with the time needed to send each combination. In this matrix rows are assumed to be the first packet sent and columns the second (rows before columns). Only when the packet is the first packet (i.e., the row, not the column) of the lowest combination in this matrix, the WAS algorithm will send this packet. Otherwise, depending on whether the transmission opportunity is a new arrival or a finished recirculation, this packet is buffered in a new loop or given another round in its loop. Note that in this situation the lowest combination will not necessarily be the next two packets to be sent as the matrix is re-evaluated at every transmission opportunity. Similarly, when the packet that triggered the transmission opportunity is actually sent, the packet that was also part of the lowest combination is not guaranteed to be transmitted next.

Similar to the WAS algorithm, the XAS algorithm, where "X" may refer to the aim of e**X**tending the period during which the outgoing line is effectively used by the two packets, also calculates a combination matrix to decide upon transmission when a transmission opportunity arises. In this matrix the

efficiency of the outgoing line is calculated for each combination by dividing the sum of both packet lengths by the total time needed to send each combination. Only when the packet that triggered the transmission opportunity is the first packet of the combination with the highest efficiency, the XAS algorithm will actually send this packet.

In Fig. 5.2 and Table 5.1 the difference between VAS, WAS and XAS is shown for an example with 5 fiber loops. Fig. 5.2 shows the occupation of the fiber loops upon a transmission opportunity (i.e., an arrival of a new packet and an available outgoing line in this case). The different fiber loops are shown horizontally and the packets they contain by means of an arrow on top of each line. In this visual representation the fiber loops are represented horizontally, i.e., as if they were laid flat after being disconnected in the point where one can decide upon recirculation of a packet. As time passes by, the arrows move to the right, possibly restarting their motion on the left side of the fiber when the arrowhead reaches the right end of the fiber loop (i.e., when the packet is recirculated). In the example of Fig. 5.2, fiber loops 1-4 are occupied while fiber loop 5 is still available. The newly arrived packet is shown in parallel to the already scheduled packets and aligns with the right side of the fiber loops. As a transmission opportunity is created by the new packet and the available outgoing line, one thus has to decide upon transmission or buffering of the packet.

The upper half of Table 5.1 shows the time to exit each combination of two packets (in terms of S) and the lower half shows the efficiency of the outgoing line during each of those combination. The fastest way to exit two packets, given the current buffer occupation, is to first send the packet in loop 1 and then the packet in loop 3. This takes $0.80 \cdot S$. The fastest way to exit two packets, given that we first send the new packet is to send the packet in loop 1 after the new packet. This takes $1.25 \cdot S$. As this is clearly a higher value than the optimal combination of $0.80 \cdot S$, WAS will decide to not send the new packet and buffer it in loop 5. Similarly the highest overall efficiency, i.e., 79.3 %, is achieved by first sending the packet in loop 4 and then the packet in loop 2. The highest achievable efficiency when the new packet is sent first, is 73.3 %. This is achieved by sending the packet in loop 2 after the new packet. Similar to WAS, XAS thus decides to not send the new packet but rather buffer it in loop 5.

When either the number of fiber loops or the maximum number of recirculations is constricted, the WAS and XAS algorithms will transmit a packet upon a transmission opportunity if doing otherwise would result in an immediate

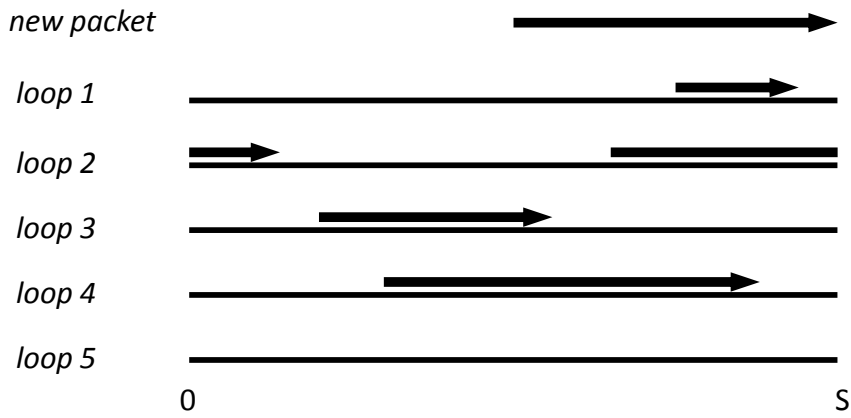


Figure 5.2: Example of a fiber loop occupancy with 5 fiber loops upon arrival of a new packet and an available outgoing line (i.e., a transmission opportunity).

Table 5.1: Decision matrices for WAS and XAS for the example of Fig. 5.2.

WAS (time to send (·S))		*Second packet*				
		new	*1*	*2*	*3*	*4*
first packet	*new*	/	1.25	1.35	1.80	1.70
	1	1.50	/	1.35	**0.80**	1.70
	2	2.50	2.25	/	1.80	2.70
	3	1.50	1.25	1.35	/	1.70
	4	1.50	1.25	1.35	1.80	/
XAS (efficiency)		*Second packet*				
		new	*1*	*2*	*3*	*4*
first packet	*new*	/	55.2 %	73.3 %	47.8 %	63.6 %
	1	46.0 %	/	50.4 %	68.8 %	45.3 %
	2	39.6 %	30.2 %	/	47.2 %	39.6 %
	3	57.3 %	44.0 %	63.0 %	/	55.3 %
	4	72.0 %	61.6 %	**79.3 %**	52.2 %	/

and unnecessary loss. Suppose for example that upon arrival of a new packet the output line is available but all of the fiber loops (finite set) are occupied by other packets. In such a case both WAS and XAS will transmit the new arrival, even though a more favorable combination may be present in the fiber loops.

By doing so the new arrival need not be dropped and unnecessary loss is prevented. Likewise, when the maximum number of recirculations is reached upon the end of a loop recirculation, WAS and XAS will always transmit the packet if the output line is available. In the terminology of [112] we could say that both WAS and XAS are tuned to avoid the use of preventive drop. In the example of Fig. 5.2 and Table 5.1 both WAS and XAS will send the new packet first if the setting would contain 4 instead of 5 loops. This is because all 4 fiber loops would be occupied and thus not available to schedule a new packet.

Note that in the case of fixed size packets both WAS and XAS schedule in exactly the same way as VAS. Indeed, as all packets have the same size, the combination that minimizes the time to transmit a pair of packets will always consist of the packet causing the transmission opportunity. Only when packet sizes are not equal to a fixed size, WAS and XAS schedule differently, and thus possibly better, than VAS.

5.2.2 Performance results

To obtain a complete and representative image of the performance of the various algorithms we look at different performance measures for different settings. For the case with an unlimited number of fiber loops and no restriction on the number of recirculations, no packets are lost and we compare the algorithms on the average packet delay. In case either, or both, the number of fiber loops or the maximum number of recirculations is limited, the loss probability (LP), or equivalently, packet loss, is our main performance measure. In addition, we study a related performance measure which we refer to as LPsize, accounting for the total size of packets lost relative to the total size of packets, or, equivalently, the fraction of data lost (in terms of packet size, not count).

As extending the analytical method from [137] to different algorithms, settings and packet size distributions proved to be too challenging, the performance of the algorithms is evaluated by means of Monte Carlo Simulations. Specifically, all algorithms are programmed in Matlab using a discrete event simulation (DES).

Fiber loops are assumed to have a length of one time unit and packet lengths to be uniformly distributed on the interval $[0,\ 1]$. Load is varied from 0.6 to 0.95 in steps of 0.05 by changing the average inter-arrival time of the

Poisson arrival process. The arrival of 10^6 packets is simulated 10 times for each algorithm and parameter combination. In this way adequate average performance measures and accompanying confidence intervals are obtained.

Fig. 5.3 shows the average packet delay for the setting with an unlimited number of fiber loops and no restriction on the maximum number of recirculations. As in this setting, for the load values investigated, the FCFS algorithm results in an unstable regime, it is not included in the graph. From Fig. 5.3 it is clear that in the unrestricted case and with a uniform packet size distribution only XAS can outperform VAS. Table 5.2 shows the relative performance improvement (in percentage) XAS can obtain in terms of waiting time compared to VAS. As the load increases, the obtainable improvement also goes up, reaching an improvement of almost 20 % for a load of 0.95.

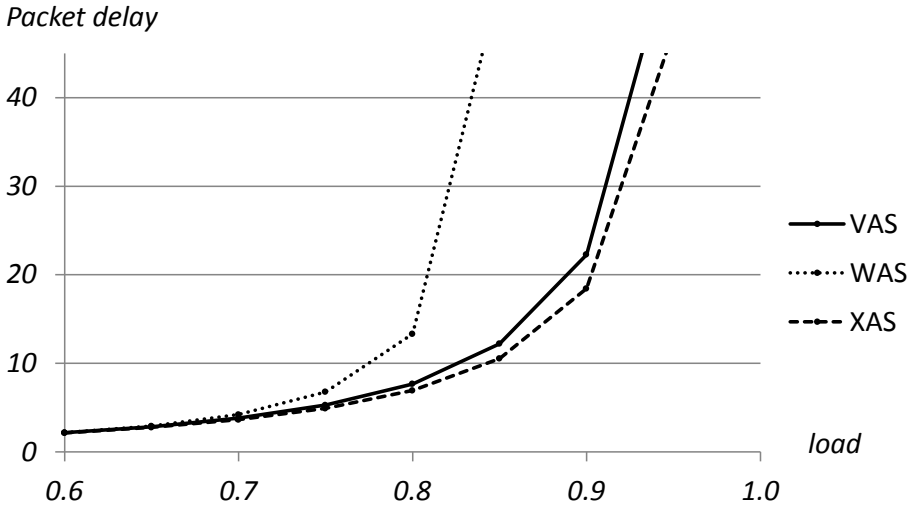


Figure 5.3: Average packet delay for the VAS, WAS and XAS algorithms in an unrestricted buffer setting.

Table 5.2: Percentage-wise performance improvement in average packet delay of XAS relative to VAS for different load values in an unrestricted buffer setting.

Packet delay reduction	*load*							
	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
XAS vs. VAS	1.1 %	2.4 %	4.2 %	7.3 %	9.6 %	13.7 %	17.2 %	18.1 %

In Table 5.3 (WAS) and Table 5.4 (XAS) the performance improvements (in percentage) compared to VAS of LP and LPsize are shown. This is done for load values of 0.6, 0.7, 0.8 and 0.9, and all combinations of the number of loops and the maximum number of recirculations (both take on values of 4, 8, 16 and infinity). From these tables it is clear that not for all combinations of parameters a performance improvement is possible. In general, but not always, performance improvements increase for lower load values, a lower number of maximum recirculations, and a higher number of loops. Comparing WAS and XAS, the WAS algorithm seems to be better suited to improve LP with improvements of up to 28 % in a setting with a low load (0.6) and a large buffer size (16), but this only for small maximum number of recirculations (4). The XAS algorithm on the other hand seems to better at improving the LPsize, with improvements of up to 41 % for certain parameter combinations, i.e., for a load value of 0.6, a maximum number of recirculations of 16 and a number of loops equal to 16 or infinite. For the case of an infinite number of recirculations and loops no packets are lost. In these cases NaN, meaning Not a Number, is shown in Table 5.3 and Table 5.4.

Note that the trends along certain parameter variations of Tables 5.3 and 5.4 are not necessarily visible in the actual performance differences of VAS, WAS and XAS. This is clear from Fig. 5.4 - 5.11 showing the difference in LP of VAS and WAS (Fig. 5.4 - 5.7) and the difference in LPsize of VAS and XAS (Fig. 5.8 - 5.11) for all parameter combinations. For example although the relative improvement tends to decrease with an increasing load, the actual differences are more steady or even increase. This is as both LP and LPsize increase with an increasing load and a smaller relative improvement can thus correspond with a larger actual performance difference. Similarly, as both LP and LPsize decrease with an increasing number of loops, the relative higher performance improvements for a higher number of loops do not necessarily translate in higher performance differences. This as opposed to the maximum number of recirculations, for which both the relative improvement and the actual performance difference decrease with an increasing number.

5.3 WAS and XAS: threshold extension

In order to strive for a further improvement of the performance of the WAS and XAS algorithms we introduce an additional threshold variable. Similar to the gap and value thresholds proposed in Chapter 4, the threshold variables in this section will be varied to achieve optimal performance.

Table 5.3: Percentage-wise performance improvement in LP and LPsize of WAS relative to VAS for different load values in various restricted buffer settings.

WAS vs. VAS		***load = 0.60***				***load = 0.70***			
		# loops				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		LP reduction (%)							
max recirculations	*4*	17.3	27.3	27.7	27.4	11.3	21.4	22.0	21.9
	8	7.0	23.5	23.4	23.7	4.3	16.6	18.2	18.1
	16	4.1	13.7	-0.7	-1.3	1.7	6.4	-1.6	-3.6
	∞	3.8	21.2	53.6	NaN	1.8	4.8	3.4	NaN
		LPsize reduction (%)							
max recirculations	*4*	8.8	8.8	8.9	8.4	3.4	1.3	1.2	0.8
	8	3.4	-3.1	-8.5	-8.1	0.8	-10.1	-16.7	-16.6
	16	3.9	-5.0	-59.8	-61.3	1.2	-10.1	-59.7	-63.4
	∞	3.6	21.5	53.0	NaN	1.8	4.9	3.3	NaN
		load = 0.80				***load = 0.90***			
		# loops				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		LP reduction (%)							
max recirculations	*4*	7.5	17.7	18.7	18.6	5.2	15.0	16.6	16.5
	8	2.5	12.6	15.8	15.9	1.3	9.8	15.6	15.7
	16	0.5	1.6	2.3	-0.3	-0.6	0.3	6.7	5.5
	∞	0.2	-4.4	-39.2	NaN	-0.6	-6.9	-32.7	NaN
		LPsize reduction (%)							
max recirculations	*4*	0.5	-3.2	-3.7	-3.8	-1.2	-5.6	-6.8	-6.8
	8	-0.8	-11.7	-20.7	-20.6	-1.9	-11.9	-21.5	-21.5
	16	0.0	-12.2	-51.3	-58.6	-1.2	-11.3	-40.4	-49.5
	∞	0.3	-4.4	-39.0	NaN	-0.6	-6.8	-32.6	NaN

Table 5.4: Percentage-wise performance improvement in LP and LPsize of XAS relative to VAS for different load values in various restricted buffer settings.

XAS vs. VAS		***load = 0.60***				***load = 0.70***			
		# loops				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		LP reduction (%)							
max recirculations	*4*	9.9	15.9	16.2	16.3	4.6	8.3	8.7	8.7
	8	-0.9	10.1	12.4	12.1	-1.0	3.4	5.1	5.5
	16	-3.3	-3.9	0.7	0.7	-2.0	-0.7	1.3	0.7
	∞	-3.8	-4.3	-9.3	NaN	-1.8	2.0	7.6	NaN
		LPsize reduction (%)							
max recirculations	*4*	17.5	30.1	30.6	30.6	11.4	22.3	23.2	23.0
	8	2.9	31.6	37.9	37.6	2.5	22.0	29.7	30.1
	16	-3.1	8.3	41.0	40.9	-1.8	8.5	35.5	36.7
	∞	-3.8	-4.3	-13.9	NaN	-1.9	2.0	8.0	NaN
		load = 0.80				***load = 0.90***			
		# loops				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		LP reduction (%)							
max recirculations	*4*	1.2	3.2	3.6	3.6	-0.6	-0.1	0.1	0.2
	8	-1.2	-0.6	-0.2	0.2	-1.6	-2.8	-3.6	-3.4
	16	-1.1	0.2	-2.0	-2.5	-0.8	-0.1	-4.3	-6.2
	∞	-1.1	3.3	13.4	NaN	-0.5	3.2	10.5	NaN
		LPsize reduction (%)							
max recirculations	*4*	7.1	16.2	17.3	17.4	4.6	11.8	13.0	13.2
	8	2.1	15.0	22.5	23.0	1.3	10.1	16.9	17.4
	16	-0.7	7.8	26.3	29.4	-0.4	5.9	18.2	21.8
	∞	-1.1	3.2	13.4	NaN	-0.5	3.3	10.5	NaN

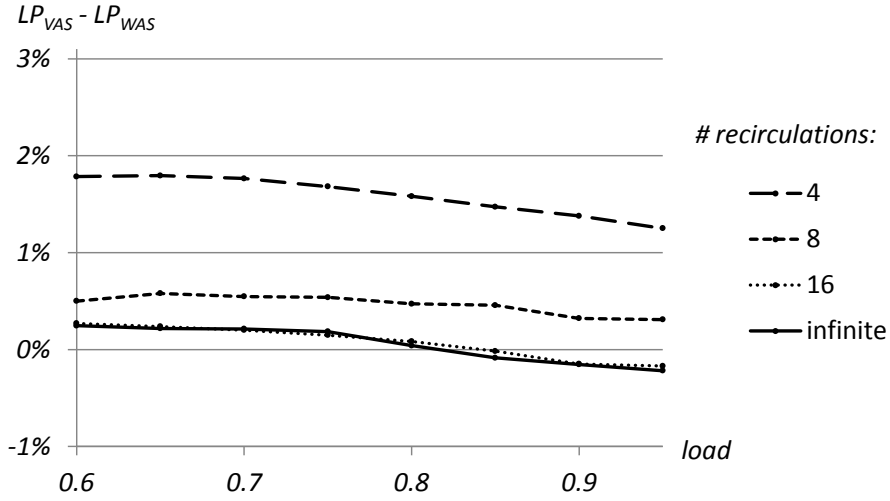


Figure 5.4: LP difference between VAS and WAS for 4 loops and different maximum number of recirculations.

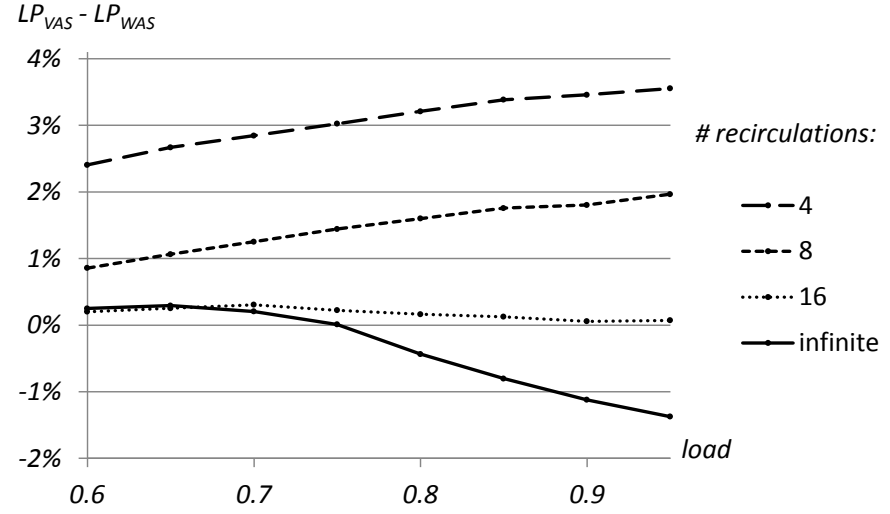


Figure 5.5: LP difference between VAS and WAS for 8 loops and different maximum number of recirculations.

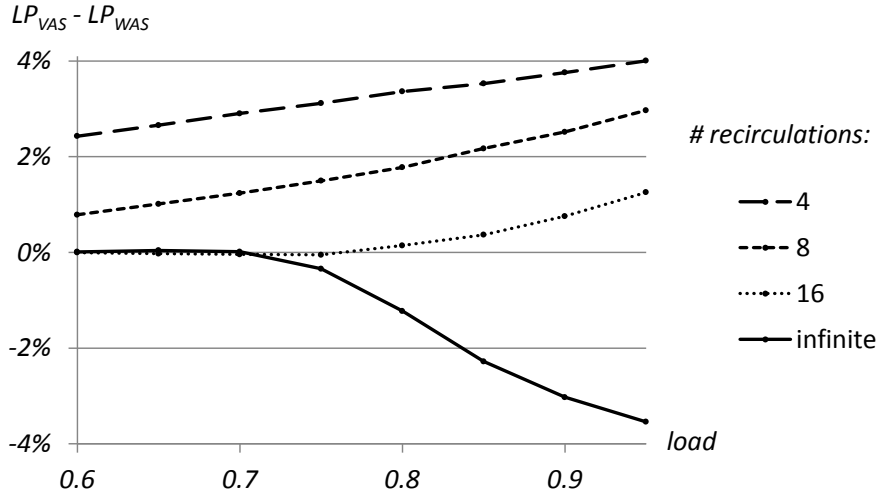


Figure 5.6: LP difference between VAS and WAS for 16 loops and different maximum number of recirculations.

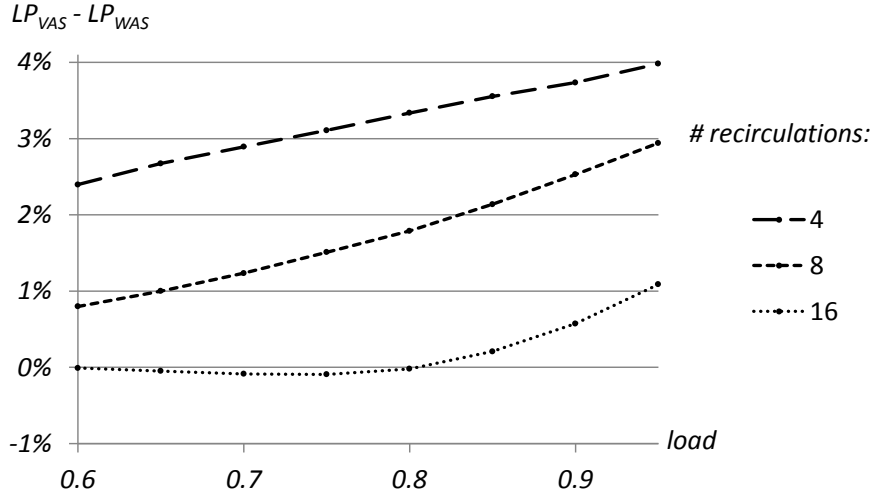


Figure 5.7: LP difference between VAS and WAS for an infinite number of loops and different maximum number of recirculations.

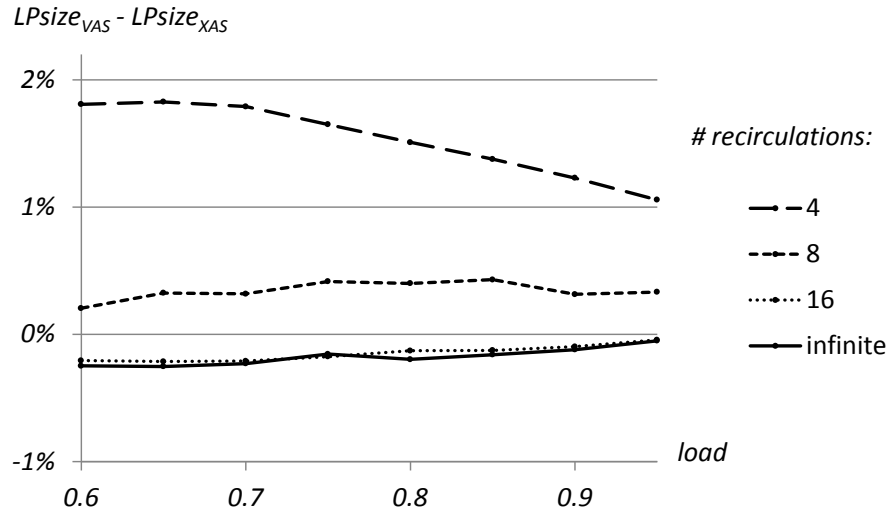


Figure 5.8: LPsize difference between VAS and XAS for 4 loops and different maximum number of recirculations.

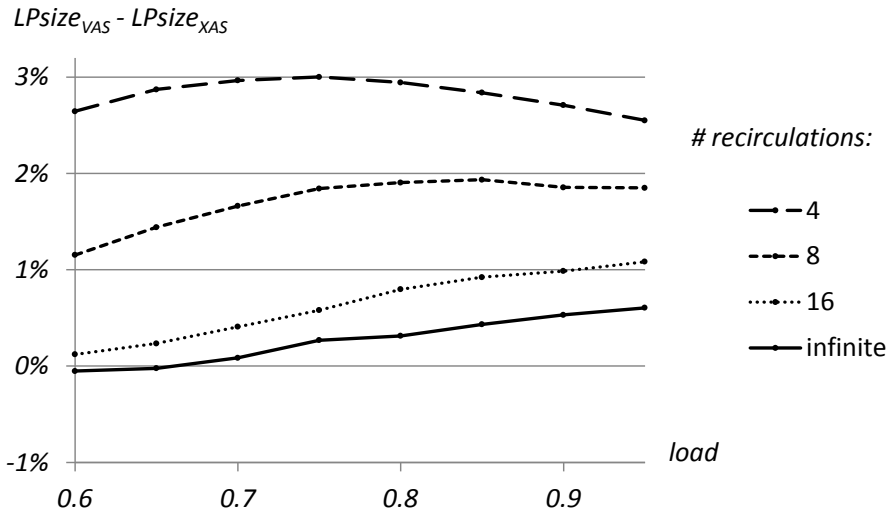


Figure 5.9: LPsize difference between VAS and XAS for 8 loops and different maximum number of recirculations.

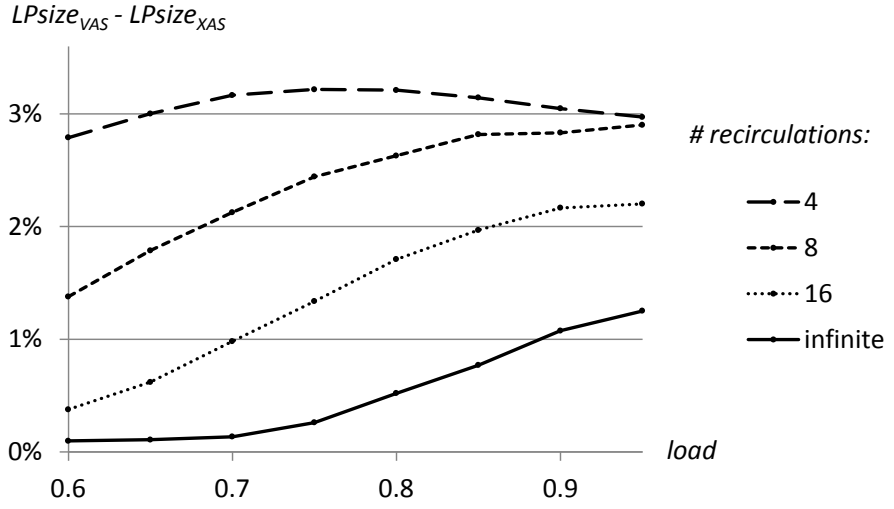


Figure 5.10: LPsize difference between VAS and XAS for 16 loops and different maximum number of recirculations.

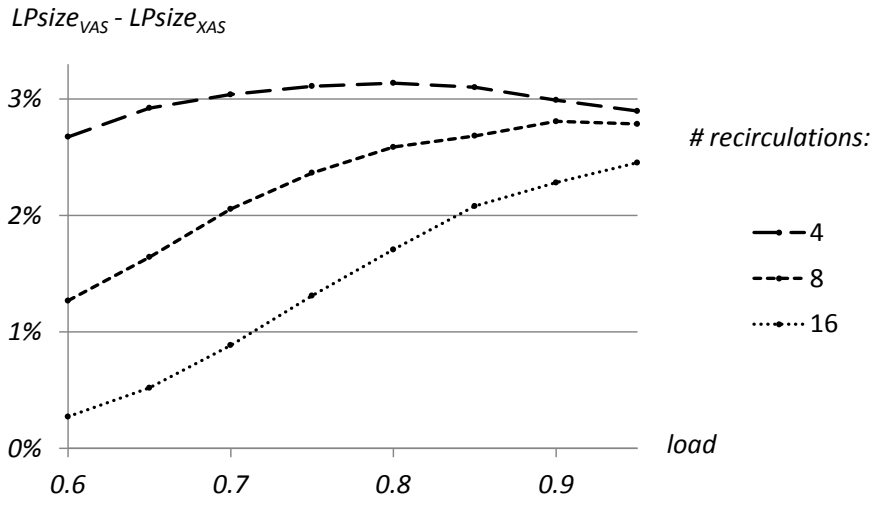


Figure 5.11: LPsize difference between VAS and XAS for an infinite number of loops and different maximum number of recirculations.

5.3.1 Approach and concept

The assumptions and the way in which the threshold extended algorithms schedule packets are similar to those discussed in Section 5.2 for the WAS and XAS algorithms (without threshold extension). We will therefore only focus on those aspects that are specifically different for the T-WAS and T-XAS algorithms.

Similar to the WAS algorithm, the T-WAS algorithm (Threshold extension of WAS), when a transmission opportunity arises, first calculates the time needed to send each combination of two packets (packets i and j, $i \neq j$) present in the system. When $N + 1$ packets are present in the system (i.e., packets in the buffer or a new arrival), this gives an $(N + 1) \times (N + 1)$ two dimensional matrix with the time needed to send each combination. In this matrix, rows are assumed to be the first packet sent and columns the second (rows before columns). As opposed to WAS, T-WAS however does not always discard the transmission opportunity when the packet causing the transmission opportunity is not the first packet (i.e., the row, not the column) of the lowest combination in this matrix. Instead, in an additional calculation, the T-WAS algorithm determines the ratio of the overall lowest combination of the matrix and the lowest combination containing the current packet as the first packet. When this ratio is smaller than the threshold (varied and optimized in different simulation runs), the T-WAS algorithm, similarly to WAS algorithm, does not send the packet but rather, depending on whether the transmission opportunity is a new arrival or a finished recirculation, buffers it in a new loop or gives it another round in its loop. When, on the other hand, the ratio is larger than the chosen threshold, the difference between the optimal combination and the combination containing the packet that causes the transmission opportunity is considered small enough and the T-WAS algorithm will nevertheless send the packet causing the transmission opportunity. As the calculated ratio is the proportion of the time it takes to send the lowest combination to that of a combination that takes longer it is always between 0 and 1. This is thus the range in which the threshold parameter is varied and optimized in simulation. Note that, similar to the WAS algorithm, when the packet that triggered the transmission opportunity is actually sent, the packet that was also part of the lowest local or overall combination is not guaranteed to be transmitted next. Similarly the overall lowest combination will not necessarily be the next two packets to be sent as the matrix is re-evaluated at every transmission opportunity.

The T-XAS algorithm (Threshold extension of XAS) calculates the same combination matrix as the XAS algorithm to decide upon transmission when a transmission opportunity arises. In this matrix the efficiency of the outgoing line is calculated for each combination by dividing the sum of both packet lengths by the total time needed to send each combination. In a second calculation the T-XAS algorithm calculates the ratio of the highest combination containing the packet causing the transmission opportunity as the first packet and the overall highest combination of the matrix. When this ratio is lower than the threshold (varied and optimized in different simulation runs), the T-XAS algorithm, similarly to the XAS algorithm, does not send the packet but rather, depending on whether the transmission opportunity is a new arrival or a finished recirculation, buffers it in a new loop or gives it another round in its loop. However, when the ratio is larger than the chosen threshold, the difference in efficiency between the optimal combination and the combination containing the packet that causes the transmission opportunity is considered small enough and the T-XAS algorithm will nevertheless send the packet causing the transmission opportunity. Although the ratio in the T-XAS algorithm is the inverse of the ratio of the T-WAS algorithm, it is, because of the way in which the elements of the matrix are calculated, also always between 0 and 1. Similarly to the T-WAS parameter, the T-XAS parameter is varied and optimized in this range.

In the example of Fig. 5.2 and Table 5.1 (upper half) the T-WAS algorithm will decide to send the newly arrived packet if the ratio of the overall optimal combination ($0.80 \cdot S$) and the optimal combination given that we first send the new packet ($1.25 \cdot S$) is larger than the chosen threshold (0.80/1.25 > threshold). Similarly the T-XAS algorithm will, based on the lower half of Table 5.1, decide to send the newly arrived packet if the ratio of the optimal combination given that we first send the new packet (73.3 %) and the overall optimal combination (79.3 %) is larger than the chosen threshold (73.3/79.3 > threshold). In those cases where the ratio is larger than the chosen threshold, the T-WAS and/or T-XAS algorithm would assess that the added value of not sending the newly arrived packet is too low.

Similar to WAS and XAS, the T-WAS and T-XAS algorithms will prevent unnecessary loss. When either the number of fiber loops or the maximum number of recirculations is restricted, they will thus transmit a packet upon a transmission opportunity if doing otherwise would result in an immediate and unnecessary loss. This is the case when for example upon arrival of a new packet the output line is available but all of the fiber loops (finite set) are occupied by other packets. In such a case both the T-WAS and T-XAS

algorithms will transmit the new arrival even though the difference with the most favorable combination in the matrix is large enough to argue against immediate transmission.

5.3.2 Performance results

To evaluate the performance of the T-WAS and T-XAS algorithms we take the same assumption as in Section 5.2.2. More specifically fiber loops are assumed to have a length of one time unit and packets to be uniformly distributed on the interval $[0,\ 1]$. Load is varied from 0.6 to 0.95 in steps of 0.05 by changing the average inter-arrival time of the Poisson arrival process.

To obtain a complete and representative image of the performance of the various algorithms we again look at different performance measures for different settings. For the case with an unlimited number of fiber loops and no restriction on the number of recirculations, no packets are lost and we compare the T-XAS algorithm on the average packet delay. The T-WAS algorithm is not evaluated for the unrestricted setting as it was shown in Section 5.2.2 that the WAS algorithm is unable to outperform VAS. We therefore choose to solely focus on the T-XAS algorithm in the unrestricted setting.

In case either, or both, the number of fiber loops or the maximum number of recirculations is limited, it was shown in Section 5.2.2 that in general WAS outperforms XAS when it comes to improving LP compared to the reference algorithm VAS. The XAS algorithm on the other hand seemed to better at improving the LPsize. Taking into account these observations we therefore choose to focus on improving LP for the T-WAS algorithm and LPsize for the T-XAS algorithm.

Table 5.5 extends the results of Table 5.2 showing the performance improvement in waiting time (in percentage) the T-XAS algorithm can obtain when compared to the VAS algorithm (row 2 in Table 5.5). This performance improvement is calculated using the same reference base, i.e., the performance of the VAS algorithm. The corresponding optimal thresholds for which these performance improvements are obtained are shown in the line below (row 3 in Table 5.5). In the bottom row (row 4 in Table 5.5) the added value of using the threshold is shown. This is the part (in percentage) of the performance improvement of the T-XAS algorithm that can be attributed to the incorporation of the threshold mechanism. The total improvement the T-XAS algorithm can obtain is indeed the sum of the improvement by using the XAS

algorithm (row 1 in Table 5.5) and the improvement by additionally using a threshold mechanism (row 2 - row 1 in Table 5.5). The ratio of this value to the total improvement ((row 2- row 1)/row 2) is thus a measure of the added value of the threshold mechanism.

It was already clear from Table 5.2 that as the load increases, the obtainable improvement of XAS vs. VAS also goes up. Table 5.5 shows that as the obtainable improvement of XAS vs. VAS increases, the added value of using an additional threshold extension decreases. In the case of an unrestricted buffer setting, extending the XAS algorithm with a threshold is thus only beneficial in those cases where the XAS algorithm (without a threshold) can only offer a limited performance improvement in average packet delay. As the added value of using an threshold mechanism decreases with an increasing load, the optimal threshold value increases. Indeed, as the threshold increases, the T-XAS algorithm increasingly approximates the XAS algorithm's scheduling behaviour and as such its performance.

Table 5.5: Performance improvement in average packet delay, optimal threshold and added value of the threshold mechanism of the T-XAS algorithm for different load values in an unrestricted buffer setting.

Packet delay reduction	*Load*							
	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
XAS vs. VAS (%)	1.1	2.4	4.2	7.3	9.6	13.7	17.2	18.1
T-XAS vs. VAS (%)	6.3	8.1	10.1	12.5	14.1	17.1	18.6	18.1
optimal threshold	0.85	0.90	0.90	0.90	0.90	0.95	0.95	1
added value threshold (%)	83	70	58	42	32	20	8	0

Similar to Table 5.5, Tables 5.6 and 5.7 extend the results of Table 5.3 and Table 5.4. In this we focus on improving LP for the T-WAS algorithm and LPsize for the T-XAS algorithm. In Table 5.6 the improvement in LP of the WAS and T-WAS algorithms is compared. It is clear that the T-WAS algorithm can always outperform the WAS algorithm. This comes naturally as the T-

WAS algorithm performs equally to the WAS algorithm when the threshold parameter is set to 1. Optimizing this threshold thus automatically results in a performance at least as good as the WAS algorithm. The same observation can be made in Table 5.7 showing the improvement in LPsize of the XAS and T-XAS algorithms.

From Table 5.3 and Table 5.4 it was already clear that in general, but not always, performance improvement increases for lower load values, a lower number of maximum recirculations, and a higher number of loops. Similar to the trend in Table 5.5, Tables 5.6 and 5.7 show that in general as the obtainable improvement of WAS vs. VAS or XAS vs. VAS increases, the added value of using an additional threshold extension decreases.

Tables 5.8 and 5.9 show the corresponding optimal thresholds for which the optimal performance improvements of the threshold algorithms in Tables 5.6 and 5.7 are obtained. The added value of using the threshold, i.e., the part (in percentage) of the performance improvement that can be attributed to the incorporation of the threshold mechanism, is also shown in these tables. As WAS and XAS are unable to outperform VAS for some parameter values, the added value of using the threshold is greater than 100 % in these cases.

For the T-XAS algorithm the optimal threshold value stays fairly constant, i.e., in the range between 0.85 and 1. This is as opposed to the T-WAS algorithm for which case the optimal thresholds can be a lot lower reaching values of 0.25. However, these low values are clearly outliers and are linked to those parameter combinations in which WAS greatly falls behind on VAS in terms of performance. In general XAS underperforms VAS less often and mostly in a less severe way than WAS.

Despite some high numbers in added value, the overall improvement of the threshold algorithm compared to VAS can still be limited in some cases. This is for example the case for the T-WAS algorithm for a load of 0.90, 16 loops and an infinite number of recirculations. In this case the added value reaches 11033 % but the overall performance improvement of the T-WAS algorithm is merely 0.3 % as the WAS algorithm falls behind the performance of the VAS algorithm by 32.7 %. The corresponding optimal threshold is an outlier of 0.25, highlighting that for this parameter combinations the optimal algorithm greatly approximates the behaviour of the VAS algorithm. In fact, given the minimal overall performance improvement it is advisable in such cases to use the VAS algorithm and not the more complicated T-WAS algorithm.

Table 5.6: Percentage-wise performance improvement in LP of the WAS and T-WAS algorithms relative to VAS for different load values in various restricted buffer settings.

WAS vs. VAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	17.3	27.3	27.7	27.4	11.3	21.4	22.0	21.9
	8	7.0	23.5	23.4	23.7	4.3	16.6	18.2	18.1
	16	4.1	13.7	-0.7	-1.3	1.7	6.4	-1.6	-3.6
	∞	3.8	21.2	53.6	NaN	1.8	4.8	3.4	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	7.5	17.7	18.7	18.6	5.2	15.0	16.6	16.5
	8	2.5	12.6	15.8	15.9	1.3	9.8	15.6	15.7
	16	0.5	1.6	2.3	-0.3	-0.6	0.3	6.7	5.5
	∞	0.2	-4.4	-39.2	NaN	-0.6	-6.9	-32.7	NaN
T-WAS vs. VAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	17.3	27.3	27.7	27.4	11.3	21.4	22.0	21.9
	8	8.7	23.5	23.5	23.7	5.9	16.9	18.7	18.8
	16	6.5	18.9	8.5	7.4	4.2	10.4	8.3	6.3
	∞	6.1	25.3	64.0	NaN	4.2	10.6	27.2	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	7.8	17.7	18.7	18.6	5.5	15.0	16.6	16.5
	8	3.9	13.4	16.2	16.3	2.5	10.6	16.0	15.9
	16	2.5	5.0	9.7	8.4	1.4	2.9	11.0	11.3
	∞	2.4	2.0	4.9	NaN	1.3	0.2	0.3	NaN

Table 5.7: Percentage-wise performance improvement in LPsize of the XAS and T-XAS algorithms relative to VAS for different load values in various restricted buffer settings.

XAS vs. VAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	17.5	30.1	30.6	30.6	11.4	22.3	23.2	23.0
	8	2.9	31.6	37.9	37.6	2.5	22.0	29.7	30.1
	16	-3.1	8.3	41.0	40.9	-1.8	8.5	35.5	36.7
	∞	-3.8	-4.3	-13.9	NaN	-1.9	2.0	8.0	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	7.1	16.2	17.3	17.4	4.6	11.8	13.0	13.2
	8	2.1	15.0	22.5	23.0	1.3	10.1	16.9	17.4
	16	-0.7	7.8	26.3	29.4	-0.4	5.9	18.2	21.8
	∞	-1.1	3.2	13.4	NaN	-0.5	3.3	10.5	NaN
T-XAS vs. VAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	18.7	30.1	30.6	30.6	12.8	22.3	23.2	23.0
	8	7.4	32.6	37.9	37.6	6.7	23.1	29.9	30.1
	16	3.5	18.2	42.3	42.8	3.5	15.6	36.5	37.1
	∞	2.7	12.5	30.4	NaN	3.4	12.5	29.9	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	8.9	16.4	17.3	17.4	6.1	11.9	13.0	13.2
	8	5.4	16.7	22.8	23.0	3.9	11.5	16.9	17.4
	16	3.4	12.4	27.9	29.5	2.8	9.2	18.7	21.8
	∞	3.2	10.3	23.7	NaN	2.7	7.3	15.7	NaN

To avoid a possible misinterpretation of Tables 5.8 and 5.9 we therefore underlined those combinations in the added value parts for which the performance improvement of the threshold algorithm compared to VAS is at least 5 % and the added value of the threshold is at least 20 %. This means that for those parameter combinations the performance improvement of the threshold algorithm is increased by at least 1 % because of the threshold mechanism. Looking at those underlined values we can see that, in general, the best settings to apply a threshold mechanism are those with a high number of recirculations. In other settings either WAS or XAS (depending on the performance measure), or VAS are, while not necessarily better, preferred for their lower complexity.

5.4 Methodology

Similar to the scheduling algorithms in Chapters 2 - 4, the performance of the scheduling algorithms is evaluated by means of Monte Carlo Simulations. Specifically, all algorithms are programmed in Matlab using a discrete event simulation (DES). Despite these resemblances, the different nature of the optical buffering stage implies there are some important differences concerning the exact implementation. We therefore go into more detail on the exact implementation of all variables, system components and their interactions.

As the assumptions on the WCs are as such that the analysis can be confined to a single wavelength, the system state, similar to the one in Chapter 4 only consists of a single part, describing the occupancy and state of the buffer loops. For each fiber loop the state includes the size and arrival number of the packet in it. If the loop is not occupied, both can be set to 0. Besides this, the state also includes a so-called phase value for each loop. This value gives the time that is left for the packet's head to complete its current recirculation in the loop and is used to quickly calculate the matrix that aids WAS, XAS and corresponding threshold variants to decide upon transmission of a packet. Last, but not least, the state also includes a value denoting the availability of the outgoing line, i.e., whether there is currently a packet being sent or not.

Similar to the algorithms in Chapters 2 - 4 the events list includes the arrivals of packets. Each time a packet arrival occurs in the event list, the arrival of the next packet is simulated, i.e., the inter-arrival time (T_{packet}) (the time between the current packet and the next) and the packet length (B_{packet}) is generated using a random number generator. After this each of the algorithms checks whether the outgoing line is available or not. If the outgoing

Table 5.8: Optimal threshold and added value of the threshold mechanism of the T-WAS algorithm from Table 5.6 for different load values in various restricted buffer settings.

optimal threshold T-WAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	1	1	1	1	0.95	1	1	1
	8	0.90	1	0.95	1	0.90	0.95	0.95	0.95
	16	0.90	0.90	0.75	0.75	0.85	0.90	0.55	0.60
	∞	0.90	0.90	0.90	NaN	0.90	0.90	0.80	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	0.95	1	1	1	0.90	1	1	1
	8	0.90	0.95	0.95	0.95	0.85	0.95	0.95	0.95
	16	0.90	0.90	0.55	0.55	0.90	0.90	0.75	0.60
	∞	0.90	0.75	0.45	NaN	0.90	0.30	0.25	NaN
added value threshold (%)		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	0	0	0	0	1	0	0	0
	8	21	0	1	0	27	2	3	4
	16	37	28	109	119	60	38	119	157
	∞	39	16	16	NaN	57	56	88	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	4	0	0	0	5	0	0	0
	8	33	7	2	2	48	7	3	1
	16	84	68	76	105	143	86	39	51
	∞	92	320	898	NaN	146	3500	11033	NaN

Table 5.9: Optimal threshold and added value of the threshold mechanism of the T-XAS algorithm from Table 5.7 for different load values in various restricted buffer settings.

optimal threshold T-XAS		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	0.95	1	1	1	0.90	1	1	1
	8	0.85	0.95	1	1	0.85	0.95	0.95	1
	16	0.85	0.90	0.95	0.95	0.85	0.90	0.95	0.95
	∞	0.8	0.85	0.85	NaN	0.85	0.85	0.90	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	0.90	0.95	1	1	0.90	0.95	1	1
	8	0.90	0.95	0.95	1	0.85	0.95	1	1
	16	0.85	0.90	0.95	0.95	0.85	0.90	0.95	1
	∞	0.85	0.90	0.90	NaN	0.85	0.90	0.95	NaN
added value threshold (%)		*# loops*				*# loops*			
		4	*8*	*16*	*∞*	*4*	*8*	*16*	*∞*
		load = 0.60				***load = 0.70***			
max recirculations	*4*	6	0	0	0	11	0	0	0
	8	61	3	0	0	63	5	1	0
	16	189	54	3	4	151	46	3	1
	∞	241	134	146	NaN	156	84	73	NaN
		load = 0.80				***load = 0.90***			
max recirculations	*4*	20	1	0	0	25	1	0	0
	8	61	10	1	0	67	12	0	0
	16	121	37	6	0	114	36	3	0
	∞	134	69	43	NaN	119	55	33	NaN

line is indeed available, the VAS algorithm immediately schedules the new arrival on the outgoing line, hereby changing the value denoting the availability of the outgoing line that is part of the system state. WAS, XAS and the corresponding threshold variants on the other hand, use the system state to calculate the time needed to send each combination of two packets (packets i and j, $i \neq j$) present in the system. When $N+1$ packets are present in the system (i.e., packets in the buffer or a new arrival), this gives an $N \times N$ two dimensional matrix with the time needed to send each combination. In this matrix rows are assumed to be the first packet sent and columns the second (rows before columns). The XAS algorithm and threshold variant also, based on this matrix, calculate an additional matrix containing the efficiency of the outgoing line for each combination by dividing the sum of both packet lengths by the total time needed to send each combination. If, based on the policy of the specific algorithm the new arrival is to depart immediately (calculated based on the correct matrix, see Section 5.2.1 and 5.3.1) or all available fiber loops are occupied, the value denoting the availability of the outgoing line is changed and this packet is listed to get a zero delay. When the outgoing line is unavailable or if the algorithm decides that the packet needs to be buffered (and it is possible as one of the loops is unoccupied), the system state values of that loop are changed to that of the corresponding packet (its packet length, reference number and phase value=S). When the outgoing line is unavailable and all loops are occupied the packet is lost. In this case and when an unlimited number of fiber loops are assumed, a new loop is initialized with the values of the corresponding packet.

At the same time a packet is scheduled in a loop, an entry is added to the event list. This entry corresponds with the next (future) scheduling opportunity of this packet after it is has completed a single recirculation and contains the time, packet size and reference and the loop it is in. At this point the algorithm will run through the same logic as that for a packet arrival to decide upon transmission or buffering the packet for an additional recirculation. In case the number of recirculations is limited and the maximum is reached, the packet will always be scheduled if the outgoing line is available in order to avoid a policy of "preventive drop". If the outgoing line is unavailable the loop is cleared and the packet is lost. If the packet is recirculated the current scheduling opportunity is crossed out and replaced by the next scheduling opportunity containing the same packet details but with different time stamp equal to the current time stamp + S. Note that different to the event list of the algorithms with FDL buffers, the event list here thus includes more than merely the arrivals of the packets. Although alternative implementations are possible, this implementation is suited best for the type of algorithm

discussed in this chapter, i.e., with a post-reservation scheme, allowing the packets to enter the buffer freely, only deciding later when a packet has to exit its loop.

This method of processing the first event from the event list and reordering the event list in chronological order is repeated iteratively until a predetermined large number of arrivals have arrived and have been scheduled. Algorithm parameters that need to be optimized are varied in different simulation runs and performance measures are tracked along the way.

5.5 Conclusions

In this chapter we proposed the WAS and XAS scheduling algorithms for optical fiber loop buffers in a variable sized packets setting. Performance was evaluated by means of Monte Carlo simulation and showed that these outperform the current state-of-the-art void-avoiding schedule (VAS) in both unlimited and restricted buffer settings. Both algorithms succeed in doing so by "looking ahead", i.e., by taking into account the schedule of other packets present in the system to selectively delay packets longer than strictly necessary. In this way XAS is capable of improving average packet delay with almost 20 % for high loads in the unlimited buffer setting. In the restricted buffer setting WAS algorithm is better at improving loss probability (LP) while XAS is better at improving LPsize (related to LP, taking into account packet size). Both algorithms succeed to improve their corresponding performance measure with tens of percentages.

To improve the performance of both WAS and XAS even further and for a wider parameter range, we extended the scheduling mechanism of WAS and XAS with a threshold parameter. By optimizing this threshold, the process of selectively delaying packets longer than strictly necessary can be made more or less strict and as such be fitted to each setting. In the case of an unrestricted buffer setting extending the XAS algorithm with a threshold is only beneficial in those cases where the XAS algorithm (without a threshold) can only offer a limited performance improvement in average packet delay. This corresponds to low values of the load and results in performance improvements that are up to 5 times as large as those without a threshold parameter. As the added value of using a threshold mechanism decreases with an increasing load, the optimal T-XAS algorithm increasingly approximates the XAS algorithm's scheduling behaviour and as such its performance. For the restricted buffer setting a similar trend can be seen, i.e., the threshold

extension adds most value for those instances where the algorithms without threshold can offer no or only limited performance improvement. While the algorithms without threshold underperform VAS strongly for some parameter combinations, the overall improvement of the threshold algorithms compared to VAS, although always positive, can still be severely limited. The settings for which the overall performance improvement of the threshold algorithms is at least 5 % and the added value of the threshold is at least 20 %, in general, correspond with those parameter combinations in which the maximum number of recirculations is high.

Chapter 6

Conclusions, future work and outlook

But what... is it good for?

Engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip

In this chapter we provide an overview of the most important insights and summarise our work presented in this dissertation. As opposed to the conclusions after each chapter, we choose to focus more on the qualitative and generalizable aspects rather than the exact obtainable numeric performance improvements. Besides we also look at possible future work and contemplate about the future of optical packet and burst switched networks.

6.1 Overview of the main results

- Within the same class of computational complexity performance of algorithms can greatly vary and be improved by a variety of scheduling strategies. This is important: any improvement achieved within a class of algorithms is achieved without complicating implementation resulting in algorithms that are attractive for switch design. **Sections 2.2.3, 3.5, 4.5, 5.4**
- Although devoted to a completely different setting, the void creating algorithm in Chapter 4 for FDL buffers and the WAS, XAS and threshold algorithms in Chapter 5 for fiber loops, use similar tactics. More specifically, by selectively delaying packets longer than strictly necessary, both types of algorithms try to take into account future packet

arrivals and as such outperform existing scheduling algorithms. This clearly highlights the generality strived for in the underlying scheduling strategies when designing algorithms. In a similar way threshold parameters and cost functions are used throughout this dissertation to optimize performance and take into account multiple facets simultaneously when scheduling. **Sections 4.2-4.4, 5.2, 5.3**

- Although results obtained by simulation are occasionally criticized for lacking generality and insight, results in this dissertation show that insights obtained and quantified by simulations are often applicable to a wide range of settings. Although algorithm parameters and threshold values are optimized for the specific settings used in our simulations, the approach is often generally applicable. Moreover, the examined insights were not only quantified by simulation but often also fundamentally improved through feedback of intermediate simulation results. As the investigated settings and algorithms are inherently complex, it remains to be seen whether any analytical method could attain this level of fine detail and, hence, insight. **Sections 2.2.1, 3.3.1, 3.4.1, 4.2.1, 4.3.1, 4.4.1, 5.2.1, 5.3.1**

- By using a cost function that takes into account several scheduling criteria, performance in terms of LP and energy consumption of the WCs can be improved significantly. However, the exact value of this improvement does depend a lot on the assumptions and parameter values of the system and setting. **Sections 2.2.2, 3.3.2, 3.4.2**

- When the number of WCs is strictly limited both existing and cost based algorithms perform (slightly) worse when more wavelengths are present for an equal overall load. This is as opposed to the setting of unlimited wavelength conversion, in which more wavelengths result in a higher multiplexing gain and thus highly (multiple orders of magnitude) improved performance. For a limited number of WCs the rescheduling is bounded by the number of WCs which have to be shared among a higher number of requests per time unit to change the wavelength. As more WCs are added, the WCs become less and less a limiting factor and the number of wavelengths determines how good contention is resolved. There is thus an intermediate value of WCs for which a higher number of wavelengths becomes beneficial. The introduction of a wavelength conversion cost term shifts this value to a lower number of WCs. **Section 2.2.2**

- For the cost function that takes into account both gap and delay (C and C-VF), it was shown that for the setting of unlimited wavelength

conversion performance improvements in LP of tens of percentages are reachable for certain values of the system parameters. This was especially true for the void-filling variant C-VF. When the number of wavelength converters is limited, the obtainable improvements in LP of C and C-VF decrease as less wavelength converters are present. **Section 2.2.2**

- The operation of the C and C-VF algorithms critically depends on the cost function and thus on the choice of the weighing factors (algorithm parameter α). The optimal value of α is fairly constant for both the setting of unlimited wavelength conversion and a limited number of WCs, and this for a broad range of other parameter combinations. The only minor exception to this rule is C-VF in the setting of a limited number of WCs, for which the optimal α decreases with a decreasing number of WCs. **Section 2.2.2**

- When using a cost function that takes into account the gap, delay and a term that relates to the use of the WCs, reduction in energy consumption can be traded off for performance decrease by means of a simple parameter β in the setting of unlimited wavelength conversion (CW and CW-VF). Results suggest that the region of 0 - 20 % energy consumption reduction yields improved energy efficiency (i.e., more energy consumption reduction than performance decrease) with the given method. When in this setting of unlimited wavelength conversion α and β are chosen in such a way that performance of existing benchmark algorithms is matched, the energy consumption can be reduced by tens of percentages for both CW and CW-VF. **Section 3.3.2**

- In the setting of a limited number of WCs, introducing a term in the cost function that penalises the use of the WCs can improve both the LP and the energy consumption simultaneously, i.e., without the trade-off. LP can be optimized as for a limited set of WCs, introducing a term that penalises their use results in a more effective use of a scarce resource. Over-restricting the use of the WCs, however, results in an under-utilisation of the WCs. When optimizing for LP, an optimal β thus emerges for intermediate values. **Section 3.4.2**

- The optimal value of β for which LP is optimized in the setting of a limited number of WCs, decreases with an increasing number of WCs. As more WCs are present, the use of a WC can, and should be, less restricted. The optimal value of β also increases for a higher number of wavelengths. This as the usage of the WCs has to be even more restricted as they are shared among more wavelengths. **Section 3.4.2**

- In general a wavelength conversion cost that rises as less WCs are vacant gives the best LP improvement in the setting of a limited number of WCs. As little performance difference exists between different rising costs, a linear rising cost may be preferred because of simplicity. **Section 3.4.2**

- Different implementations of the void-creating mechanism are able to improve performance in a variety of performance measures (LP, delay and gap) for fixed packet size on a single wavelength, general packet size on a single wavelength and fixed packet size on multiple wavelengths. **Sections 4.2-4.4**

- Although a void-creating approach based on gap thresholds and iterative simulations are able to improve performance for the fixed packet size and single wavelength setting, the method based on void values performs better for this setting. Moreover as the void value method can also be used for a setting of general packet size on a single wavelength and fixed packet size on multiple wavelengths, it should be the overall preferred method. **Sections 4.2-4.4**

- The achievable improvement of the void-creating algorithm based on void values can reach tens of percentages but depends on the packet size distribution and, in general, decreases as the variance of the packet size distribution increases. The result is nevertheless a scheduling algorithm that is as simple to implement as a classic scheduling algorithm, deciding upon void creation using a closed-form expression derived from the packet size distribution. **Section 4.3**

- For the setting of a fixed packet size on multiple wavelengths, the method based on the added value is extended with only little modifications. As results point out, the adapted algorithm performs very well for a modest number of wavelengths like 2 or 4. **Section 4.4**

- Although the reference algorithm for the multiple wavelength setting is the gap focused G-D algorithm whose operation fundamentally differs from that of the delay-based reference algorithms of the single wavelength setting, our void creating algorithm still manages to improve performance in the same order of magnitude. This clearly shows that void creation is not a competing mechanism for more advanced void filling and can thus be used mostly in a complementary manner, to improve the performance of a variety of algorithms. **Section 4.4**

- The WAS and XAS algorithms for optical fiber loop buffers in a variable sized packets setting outperform the current state-of-the-art void-

avoiding (VAS) schedule in both unlimited and restricted buffer settings. In the unrestricted buffer setting only XAS can outperform VAS in terms of packet delay. In the restricted buffer setting WAS is better at improving LP while XAS is better at improving LPsize. **Section 5.2**

- The T-WAS and T-XAS algorithms greatly outperform their counterparts without threshold for those parameter combinations in which WAS and XAS can only offer a limited or no performance improvement compared to VAS. For the unrestricted buffer setting this corresponds to low values of the load. For the restricted buffer this, in general, corresponds to those parameter combinations for which the maximum number of recirculations is high. **Section 5.3**

6.2 Future work

Here, we discuss possible directions for future work. We order the proposed paths firstly from general to more specific and secondly according to the order of their corresponding chapter in this dissertation.

- As the complexity of the settings and algorithms in this dissertation prevents evaluating their performance in an analytic way, we resort to Monte Carlo and discrete event simulation techniques. Although this method has proven to not only result in the performance evaluation of a wide range of settings and algorithms but also in an increased insight in the underlying scheduling strategies, obtaining closed form expressions by means of an analytical method would greatly improve this insight and undoubtedly result in even better performing and fine-tuned scheduling algorithms. Future effort may therefore be directed at analyzing the proposed algorithms by means of analytical methods. To do this it seems advisable to extend existing analytical derived results for simplified settings and algorithms as in [126, 131, 137] rather than from the complex and detailed settings and algorithms proposed in this dissertation. **Section 1.9.3**

- As mentioned in the introductory chapter, the order in which the optical buffer and wavelength converters are probed can be changed. Despite this inherent flexibility, we assumed a wavelength conversion before buffering setting throughout this dissertation. This as the focus is on improving the effective use of optical buffers and the second stage, whether it is the optical buffer or the wavelength converters,

can always be used more effectively. However, as in some settings the number of wavelength converters is strictly limited, they can be an even scarcer resource than the optical buffer capacity. It would therefore be interesting to analyse the performance of some settings with wavelength conversion after optical buffering. Besides this future work could also look into cases wherefore the order in which wavelength converters and optical buffers are probed can change dynamically, i.e., on a packet to packet basis. **Section 1.8.1**

- When using a cost function that takes into account the use of the WCs, results suggest that the region of 0 - 20 % energy consumption reduction yields improved energy efficiency (i.e., more energy consumption reduction than performance decrease) for the setting of unlimited wavelength conversion. For larger reductions, it could thus be beneficial to develop a different approach. Cost functions that incorporate the length of the packets that are scheduled on a different wavelength could for example provide alternative or additional means to take into account the energy consumption of the WCs. **Section 3.6**

- The method of void values opens up the opportunity to create an algorithm, void-creating or more classic, in which a simple trade-off between performance and computational complexity is possible by, based on the void values, keeping track of more or less voids. **Section 4.6**

- As the obtainable performance improvement of the void-creating algorithm for fixed packet size on multiple wavelengths tends to decrease with an increasing number of wavelengths of 6, 8 or more, future work may focus on extending the void-creating algorithm for multiple wavelengths to take into account the values of other voids in the threshold value function. An extension to general packet size on multiple wavelengths may then also be part of the scope. **Section 4.6**

- WAS, XAS and their threshold variants currently take into account the total transmission time and transmission efficiency of combinations of two packets to decide upon delaying a packet more than necessary. An interesting path for future work could be to extend this analysis by looking at combinations of more than two packets. This would increase computational complexity but would probably also lead to further performance improvement. **Section 5.5**

6.3 Outlook

If the quotes at the beginning of each chapter in this dissertation demonstrate anything, it is that the future is hard to predict, even for an expert. Whether and when optical packet or burst switching will be a commercial and operational viable solution to address switching in the core Internet nodes is hard to tell. There are a few underlying trends that increasingly push the balance in favour of optical switching solutions as time passes by: the ever increasing demand for Internet traffic, the fundamental limitations of electronic switching and the technological advances in optical switching. On the other hand, a number of operational, technological and economical bottlenecks currently still prevent optical switching to achieve the breakthrough that once was envisioned to happen a lot sooner. Moreover, given the genuine drivers, electronic switching may well be revolutionized by a dark horse as well. Nevertheless the author is convinced that optical packet and burst switching remain viable alternatives for future network architectures, not in the least as they come up to the needs of a more simplified and scalable network architecture. If and when optical packet or burst switching becomes the go-to technology for core node switching, optical buffering and the complementary scheduling algorithms will be at the heart of switch design. The author therefore hopes that this dissertation will have its place in the deployment of optical packet and burst switching technology in future networks.

Appendices

Appendix A

Pseudocode for the non-void-filling C algorithm

(see section 2.2.3)

input $(D,\ c,\ \rho,\ \#$ arrivals, $E[B],\ N,\ \alpha)$

initialize *horizons* as zero column vector $(1 \times c)$
initialize *# lost packets* $= 0$
initialize *payload converted packets* $= 0$

while $i <$ # arrivals

    generate B_{packet} = packet_ length$(E[B])$
    generate T_{packet} = packet_ interarrival_ time$(E[B],\ c,\ \rho)$
    generate w = packet_ wavelength(c)

    if lowest value in *horizons* $> N \times D$
        # lost packets = # lost packets $+1$
    else
        initialize *cost* as zero column vector $(1 \times c)$
        initialize *scheduling wavelength*
        initialize *minimal cost* $= \infty$

        for $j = 1 : c$
            slack $=$ *horizons*$(1,\ j)$ modulo D
            if slack $= 0$
                gap $= 0$
                delay $=$ *horizons*$(1,\ j)$
            else
                gap $= D$ – *slack*$(1,\ j)$

delay $=$ *horizons*$(1,\ j)+$ *gap*$(1,\ j)$
end

cost$(1,\ j) = \alpha \times$ *gap* $+\ (1-\alpha) \times$ *delay*
if *horizons*$(1,\ j) > N \times D$
cost$(1,\ j) = \infty$
end

if *cost*$(1,\ j) <$ *minimal cost* $\|$
$(j =$ *w* & & *cost*$(1,\ j) =$ *minimal cost*$)$
minimal cost $=$ *cost*$(1,\ j)$
scheduling wavelength $= j$
end

end

horizons$(1,$ *scheduling wavelength*$)$ $=$ delay$(1,$ *scheduling wavelength*$)$
$+B_{packet}$

if *scheduling wavelength* $\neq w$
payload converted packets $=$ *payload converted packets* $+B_{packet}$
end

end

horizons $=$ maximum(*horizons* – $T_{packet},\ 0)$
$i = i + 1$

end

Appendix B

Pseudocode for the CWC-VF algorithm

(see section 2.2.3)

input $(D,\ c,\ \rho,\ r,\ \#$ arrivals, $E[B],\ N,\ \alpha,\ \beta)$

initialize *V* as zero vector $(c \times 1 \times 2)$
$V(:,\ 1,\ 2)$
initialize *WCs* as zero column vector $(1 \times r)$
initialize *# lost packets* $= 0$

while $i <$ # arrivals

generate $B_{packet} =$ packet_ length$(E[B])$
generate $T_{packet} =$ packet_ interarrival_ time$(E[B],\ c,\ \rho)$
generate $w =$ packet_ wavelength(c)

initialize *gap* as ∞ matrix $(c \times N + 1)$
initialize *cost* as ∞ matrix $(c \times N + 1)$
initialize *scheduling wavelength* $= \infty$
initialize *scheduling FDL* $= \infty$
initialize *scheduling void* $= \infty$
initialize *minimal cost* $= \infty$
initialize *available WCs* $= 0$
initialize *available WC* $= 0$

for $k = 1 : r$
if *WCs*$(1,\ k) = 0$
available WCs $=$ *available WCs* $+1$
available WC $= k$

```
    end
end

for j = 1 : c
    for k = 1 : size(V, 2)
        delay = ceiling(V(j, k, 1)/D)
        if V(j, k, 1) ≤ (N − 1) × D & &
        ( delay×D + B_packet) <V(j, k, 2)
            gap(j, delay+1) =delay×D−V(j, k, 1)
            cost(j, delay+1) = α×gap(j, delay+1)
            +(1 − α) × (k − 1) × D
            if j ≠ w
                if available WCs = 0
                    cost(j, delay+1) = ∞
                else
                    cost(j, delay+1) = cost(j, delay+1)
                    +β × D × (r−available WCs+1)/r
                end
            end
        end

        if cost(j, delay+1) < minimal cost ||
        (j = w & & cost(j, delay+1) = minimal cost)
            minimal cost = cost(j, delay+1)
            scheduling wavelength = j
            scheduling FDL = delay+1
            scheduling void = k
        end
    end
end

if minimal cost = ∞
    # lost packets = # lost packets +1
else
    scheduling wavelength ≠ w
        WCs(1, available WC) = B_packet
    end
    if V(scheduling wavelength, size(V, 2), 1) = ∞
        scheduling_ wavelength_ with_ dummy_ void = 1
        initialize new_ V as ∞ matrix (c×size(V, 2) × 2)
    else
        scheduling_ wavelength_ with_ dummy_ void = 0
```

```
            initialize new_ V as ∞ matrix (c×size(V, 2) + 1 × 2)
        end
        new_ V(:, 1 :size(V, 2), :) =V(:, 1 :size(V, 2), :)
        new_ V(scheduling wavelength, scheduling void, 2) =
        (scheduling FDL−1) × D
        new_ V(scheduling wavelength, scheduling void+1, 1) =
        (scheduling FDL−1) × D + B_packet
        new_ V(scheduling wavelength, scheduling void+1, 2) =
        (V(scheduling wavelength, scheduling void, 2)
        if scheduling void < (size(V, 2) − 1)×
        scheduling_ wavelength_ with_ dummy_ void
            new_ V(scheduling wavelength, scheduling void+2 :
            size(new_ V, 2), :) =
            V(scheduling wavelength, scheduling void+1 : size(V, 2), :)
        end
        V = new_ V
    end

    V = maximum(V − T_packet, 0)
    WCs = maximum(WCs − T_packet, 0)

    initialize zero_ end as zero column vector (1 × c)

    for j = 1 : c
        for k = 1 : size(V, 2)
            if V(j, k, 1) = 0
                zero_ end(j) = k
            else
                break
            end
        end
    end

    initialize new_ V as ∞ matrix (c × (size(V, 2)−
    minimum(zero_ end)) × 2)
    for j = 1 : c
        new_ V(j, 1 :size(V, 2)−zero_ end(j), :) =
        V(j, zero_ end(j) + 1 :size(V, 2), :)
    end

    V = new_ V
    i = i + 1
end
```

Bibliography

[1] Gapminder, "Gapminder tools: bubble maps," 2017. [Online]. Available: http://www.gapminder.org/tools

[2] P. M. Fernandez, "Circuit switching in the internet," Ph.D. dissertation, Stanford University, 2003.

[3] Cisco press release, "The zettabyte era: trends and analysis," June 2017. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

[4] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, Dec 1998.

[5] S. Deffree, "ARPANET establishes 1st computer-to-computer link, october 29, 1969," Oct. 2017. [Online]. Available: http://www.edn.com/electronics-blogs/edn-moments

[6] ESA press release, "How many stars are there in the universe?" 2018. [Online]. Available: http://www.esa.int/Our_Activities/Space_Science/Herschel/How_many_stars_are_there_in_the_Universe

[7] N. Margalit, "Star versus ring topologies in metropolitan-network applications," Sep 1999. [Online]. Available: http://www.lightwaveonline.com/articles/print/volume-16/issue-10.html

[8] K. Finley, "Internet by satellite is a space race with no winners," June 2015. [Online]. Available: http://www.wired.com/2015/06/elon-musk-space-x-satellite-internet/

[9] S. Duggan, "International high capacity connectivity through submarine cables," 2017. [Online]. Available: http://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/Pages/default.aspx

[10] S. Bjornstad, "Optical telecom networks: The ultimate capacity accross land and sea - efficient utilization required," 2015. [Online]. Available: https://www.nito.no

[11] Nippon Telegraph and Telephone Corporation, "One petabit per second fiber transmission over a record distance of 200 km," Mar 2017. [Online]. Available: http://www.ntt.co.jp/news2017/1703e/pdf/170323a.pdf

[12] W. Rogiest, "Stochastic modeling of optical buffers," Ph.D. dissertation, Ghent University, 2008.

[13] T. Simonite, "Moore's law is dead. now what?" May 2016. [Online]. Available: http://www.technologyreview.com/s/601441/moores-law-is-dead-now-what/

[14] Intel press release, "Platform 2015: Intel processor and platform evolution for the next decade," 2015. [Online]. Available: http://web.archive.org/web/20110427072037/http://epic.hpi.uni-potsdam.de/pub/Home/TrendsAndConceptsII2010/HW_Trends_borkar_2015.pdf

[15] C. Carvalho, "The gap between processor and memory speeds," in *Proceedings of 3rd Internal Conference on Computer Architecture (ICCA)*, Jan 2002, pp. 1–8.

[16] K. Koltsov, "Have you ever wondered why your long distance VoIP call quality is bad?" 2016. [Online]. Available: http://www.linkedin.com/pulse/have-you-ever-wondered-why-your-long-distance-voip-call-koltsov/

[17] B. Mukherjee, "Architecture, control, and management of optical switching networks," in *Proceedings of Photonics in Switching*, Aug 2007, pp. 43–44.

[18] W. V. Heddeghem, B. Lannoo, D. Colle, M. Pickavet, F. Musumeci, A. Pattavina, and F. Idzikowski, "Power consumption evaluation of circuit-switched versus packet-switched optical backbone networks," in *Proceedings of IEEE Online Conference on Green Communications*, Oct 2013, pp. 56–63.

[19] G. J. Eilenberger, "Integrated electrical/optical switching for future energy efficient packet networks," in *Proceedings of 2011 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference*, Mar 2011, pp. 1–3.

[20] N. Bisht and R. Jain, "Optical networking: Recent advances, trends, and issues," *Annual Review of Communications*, vol. 55, pp. 1–8, Nov 2002.

[21] S. Perrin, "The optical switching revival: Rebuilding optical networks for packets," 2009. [Online]. Available: http://www.heavyreading.com/details.asp?sku_id=2356&skuitem_itemid=1170

[22] S. Tomic, B. Statovci-Halimi, A. Halimi, W. Muellner, and J. Fruehwirth, "ASON and GMPLS - overview and comparison)," *Photonic Network Communications*, vol. 7, no. 2, pp. 111–130, Mar 2004.

[23] H. press release, "White paper on Huawei MS-OTN low-latency network solution," 2016. [Online]. Available: http://www.huawei.com/en/press-events/news/2016/10/Low-Latency-White-Paper-PGM2-Concept

[24] P. Littlewood and E. Follis, "Optical transport networking," 2016. [Online]. Available: http://media.ciena.com/documents/Experts_Guide_to_OTN_ebook.pdf

[25] H. press release, "Huawei and SK Telecom successfully deploy 200G ASON backbone commercial network," 2017. [Online]. Available: http://www.huawei.com/en/press-events/news/2017/2/200G-ASON-Backbone-Commercial-Network

[26] K. Bergman and S. Rumley, "Optical switching performance metrics for scalable data centers," in *Proceedings of the 21st OptoElectronics and Communications Conference (OECC)*, July 2016, pp. 1–3.

[27] H. Rastegarfar, K. Khavari, S. LaRochelle, L. A. Rusch, and A. Leon-Garcia, "A high-performance network architecture for scalable optical datacenters," in *Proceedings of IEEE Photonic Society 24th Annual Meeting*, Oct 2011, pp. 439–440.

[28] D. Zhang, H. Guo, T. Yang, and J. Wu, "Optical switching based small-world data center network," *Computer Communications*, vol. 103, pp. 153 – 164, May 2017.

[29] S. Verma, H. Chaskar, and R. Ravikanth, "Optical burst switching: a viable solution for terabit IP backbone," *IEEE Network*, vol. 14, no. 6, pp. 48–53, Nov 2000.

[30] M. Yoo, C. Qiao, and S. Dixit, "QoS performance of optical burst switching in IP-over-WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2062–2071, Oct 2000.

[31] B. Nleya and A. Mutsvangwa, "QoS considerations in OBS switched backbone networks," *Global Journal of Computer Science and Technology*, vol. 14, no. 5, pp. 23–31, Oct 2014.

[32] G. Chen, H. Guo, D. Zhang, Y. Zhu, C. Wang, H. Yu, Y. Wang, J. Wang, J. Wu, X. Cao, N. Yoshikane, T. Tsuritani, I. Morita, and M. Suzuki, "First demonstration of holistically-organized metro-embedded cloud platform with all-optical interconnections for virtual datacenter provisioning," in *Proceedings of Opto-Electronics and Communications Conference (OECC)*, June 2015, pp. 1–3.

[33] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838–1851, Oct 2000.

[34] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: a new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16–23, May 2004.

[35] K. Sriram, D. Griffith, S. Lee, and N. Golmie, "Optical burst switching: Benefits and challenges," in *Proceedings of 1st International Workshop Optical Burst Switching (WOBS)*, Oct 2003, pp. 1–12.

[36] H. press release, "Huawei launches world's first 10 petabit all-optical switch prototype leading the way to ultra-large capacity green optical networks," 2012. [Online]. Available: http://pr.huawei.com/en/news/hw-124643-petabitprototypeultra-largecapacity.htm#.Wnx71a7tzDC

[37] M. Imran, M. Collier, P. Landais, and K. Katrinis, "Software-defined optical burst switching for HPC and cloud computing data centers," *Journal of Optical Communication Networks*, vol. 8, no. 8, pp. 610–620, Aug 2016.

[38] B. Rajesh and M. H. Kumar, "Designing of optical burst switching for data center networks," *International Journal of Engineering and Science Research*, vol. 5, no. 10, pp. 1306–1320, Oct 2015.

[39] S. Tariq and M. Bassiouni, "Performance evaluation of MPTCP over optical burst switching in data centers," in *International Telecommunications Symposium (ITS)*, Aug 2014, pp. 1–5.

[40] M. Imran, P. Landais, M. Collier, and K. Katrinis, "Performance analysis of optical burst switching with fast optical switches for data center networks," in *17th International Conference on Transparent Optical Networks (ICTON)*, July 2015, pp. 1–4.

[41] R. Rajaduray, "Unbuffered and limited-buffer all-optical networks," Ph.D. dissertation, University of California, 2005.

[42] D. J. Blumenthal, B. E. Olsson, G. Rossi, T. E. Dimmick, L. Rau, M. Masanovic, O. Lavrova, R. Doshi, O. Jerphagnon, J. E. Bowers, V. Kaman, L. A. Coldren, and J. Barton, "All-optical label swapping networks and technologies," *Journal of Lightwave Technology*, vol. 18, no. 12, pp. 2058–2075, Dec 2000.

[43] R. Kakarla and D. Venkitesh, "Demonstration of optical header recognition for BPSK data using novel design of logic gates," *Optics Communications*, vol. 363, pp. 117 – 122, Mar 2016.

[44] D. Blumenthal, "Photonic packet switching and optical label swapping," *Optical Networks Magazine*, vol. 2, pp. 54–65, Jan 2001.

[45] G. N. Rouskas and L. Xu, *Optical Packet Switching*. Springer US, June 2005, pp. 111–127.

[46] R. V. Caenegem, D. Colle, M. Pickavet, P. Demeester, K. Christodoulopoulos, K. Vlachos, E. Varvarigos, L. Stampoulidis, D. Roccato, and R. Vilar, "The design of an allo-optical packet switching network," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 52–61, Nov 2007.

[47] S. Yao, S. J. B. Yoo, B. Mukherjee, and S. Dixit, "All-optical packet switching for metropolitan area networks: opportunities and challenges," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 142–148, Mar 2001.

[48] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: an overview," *IEEE Communications Magazine*, vol. 38, no. 2, pp. 84–94, Feb 2000.

[49] A. Jajszczyk and H. T. Mouftah, "Photonic fast packet switching," *IEEE Communications Magazine*, vol. 31, no. 2, pp. 58–65, Feb 1993.

[50] I. SZCZESNIAK, "Overview of optical packet switching," *Theoretical and applied informatics*, vol. 21, no. 3-4, pp. 167–180, Nov 2009.

[51] R. Ramaswami, "Optical networking technologies: what worked and what didn't," *IEEE Communications Magazine*, vol. 44, no. 9, pp. 132–139, Sep 2006.

[52] R. S. Tucker, "How to build a petabit-per-second optical router," in *Proceedings of 19th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS)*, Oct 2006, pp. 486–487.

[53] ——, "Scalability and energy consumption of optical and electronic packet switching," *Journal of Lightwave Technology*, vol. 29, no. 16, pp. 2410–2421, Aug 2011.

[54] T. S. El-Bawab and J.-D. Shin, "Optical packet switching in core networks: between vision and reality," *IEEE Communications Magazine*, vol. 40, no. 9, pp. 60–65, Sep 2002.

[55] R. Stabile, "Integrated InP optical switch matrices performance for packet data networks," in *Proceedings of 21st OptoElectronics and Communications Conference (OECC)*, July 2016, pp. 1–3.

[56] M. J. O'Mahony, D. Simeonidou, D. K. Hunter, and A. Tzanakaki, "The application of optical packet switching in future communication networks," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 128–135, Mar 2001.

[57] K. I. Kitayama, A. Hiramatsu, M. Fukui, T. Tsuritani, N. Yamanaka, S. Okamoto, M. Jinno, and M. Koga, "Photonic network vision 2020; toward smart photonic cloud," *Journal of Lightwave Technology*, vol. 32, no. 16, pp. 2760–2770, Aug 2014.

[58] S. J. B. Yoo, "Energy efficiency in the future Internet: The role of optical packet switching and optical-label switching," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 17, no. 2, pp. 406–418, Mar 2011.

[59] J. Choi, G.-H. Lee, T. Kim, Y. Kim, and J. Park, "All-optical packet buffer system based on controllable-delay recirculating loop," *Microwave and Optical Technology Letters*, vol. 56, no. 9, pp. 2058–2061, June 2014.

[60] M. Moralis-Pegios, G. Mourgias-Alexandris, T. Alexoudi, M. Cherchi, M. Harjanne, T. Aalto, N. Pleros, and K. Vyrsokinos, "Optical time-slot interchanger and si-based delay lines towards integrated feed-forward buffers," in *Proceedings of IEEE Photonics Society Summer Topical Meeting Series (SUM)*, July 2017, pp. 21–22.

[61] A. Triki, A. Gravey, P. Gravey, and M. Morvan, "Long-term CAPEX evolution for slotted optical packet switching in a metropolitan network," in *Proceedings of International Conference on Optical Network Design and Modeling (ONDM)*, May 2017, pp. 1–6.

[62] Y. Ji, J. Zhang, Y. Zhao, X. Yu, J. Zhang, and X. Chen, "Prospects and research issues in multi-dimensional all optical networks," *Science China Information Sciences*, vol. 59, no. 10, pp. 1–14, Sep 2016.

[63] M. N. Dharmaweera, R. Parthiban, and Y. A. Şekercioğlu, "Toward a power-efficient backbone network: The state of research," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 198–227, Jan 2015.

[64] E. Burmeister, D. Blumenthal, and J. Bowers, "A comparison of optical buffering technologies," *Optical Switching and Networking*, vol. 5, no. 1, pp. 10 – 18, Mar 2008.

[65] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, Dec 1998.

[66] R. Jankuniene and P. Tervydis, "The contention resolution in OBS network," *Elektronika ir Elektrotechnika*, vol. 20, no. 6, June 2014.

[67] S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "A unified study of contention-resolution schemes in optical packet-switched networks," *Journal of Lightwave Technology*, vol. 21, no. 3, pp. 672–683, Mar 2003.

[68] C.-F. Hsu, T.-L. Liu, and N.-F. Huang, "Performance analysis of deflection routing in optical burst-switched networks," in *Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, June 2002, pp. 66–73.

[69] J. Bisht and A. Goel, "Pre-deflection routing with control packet signal scheme in optical burst switch networks," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 8, no. 3, pp. 520–523, Mar 2014.

[70] S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "All-optical packet-switched networks: a study of contention-resolution schemes in an irregular mesh network with variable-sized packets," in *Proceedings of SPIE OptiComm*, Sep 2000, pp. 235–246.

[71] C. M. Gauger, "Optimized combination of converter pools and FDL buffers for contention resolution in optical burst switching," *Photonic Network Communications*, vol. 8, no. 2, pp. 139–148, Sep 2004.

[72] Cisco press release, "Introduction to DWDM technology," 2001. [Online]. Available: https://www.cisco.com/c/dam/global/de_at/assets/docs/dwdm.pdf,

[73] R. S. Tucker, R. Parthiban, J. Baliga, K. Hinton, R. W. A. Ayre, and W. V. Sorin, "Evolution of WDM optical IP networks: A cost and energy perspective," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 243–252, Feb 2009.

[74] D. Richardson, J. Fini, and L. Nelson, "Space division multiplexing in optical fibres," *Nature Photonics*, vol. 7, pp. 354–362, May 2013.

[75] K. Saitoh and S. Matsuo, "Multicore fiber technology," *Journal of Lightwave Technology*, vol. 34, no. 1, pp. 55–66, Jan 2016.

[76] R. S. Luís, H. Furukawa, G. Rademacher, B. J. Puttnam, and N. Wada, "Hybrid optical packet and circuit switching in spatial division multiplexing fiber networks," in *Proceedings of International Conference on Optical Network Design and Modeling (ONDM)*, May 2017, pp. 1–4.

[77] T. Mizuno, H. Takara, K. Shibahara, A. Sano, and Y. Miyamoto, "Dense space division multiplexed transmission over multicore and multimode fiber for long-haul transport systems," *Journal of Lightwave Technology*, vol. 34, no. 6, pp. 1484–1493, Mar 2016.

[78] J. Wang, C. McArdle, and L. P. Barry, "Large-scale optical datacentre networks using hybrid fibre delay line buffers and packet retransmission," in *Proceedings of 18th International Conference on Transparent Optical Networks (ICTON)*, July 2016, pp. 1–4.

[79] A. Kushwaha, S. K. Bose, and Y. N. Singh, "Analytical modeling for performance studies of an FLBM-based all-optical packet switch," *IEEE Communications Letters*, vol. 5, no. 5, pp. 227–229, May 2001.

[80] T. Tanemura, I. M. Soganci, T. Oyama, T. Ohyama, S. Mino, K. A. Williams, N. Calabretta, H. J. S. Dorren, and Y. Nakano, "Large-capacity compact optical buffer based on inp integrated phased-array switch and coiled fiber delay lines," *Journal of Lightwave Technology*, vol. 29, no. 4, pp. 396–402, Feb 2011.

[81] M. Sumetsky, "Dispersionless low-loss miniature slow light delay lines based on optical fibers," in *Proceedings of Optical Fiber Communication Conference*, Mar 2014, pp. 1–3.

[82] R. S. Tucker, P.-C. Ku, and C. J. Chang-Hasnain, "Slow-light optical buffers: capabilities and fundamental limitations," *Journal of Lightwave Technology*, vol. 23, no. 12, pp. 4046–4066, Dec 2005.

[83] A. V. Dmitriev, N. A. Toropov, and M. Sumetsky, "Miniature optical delay lines and buffers," in *Proceedings of the 18th International Conference on Transparent Optical Networks (ICTON)*, July 2016, pp. 1–3.

[84] K. Van Hautegem, W. Rogiest, and H. Bruneel, "Improving the energy efficiency of scheduling algorithms for OPS/OBS buffers," *Photonics Network Communications (PNET)*, vol. 29, no. 2, pp. 183–197, Apr 2015.

[85] ——, "Improving performance and energy consumption of shared wavelengths converters in OPS/OBS," *Optical Switching and Networking (OSN)*, vol. 17, pp. 15–38, July 2015.

[86] ——, "Selective void creation/filling for variable size packets and multiple wavelengths," *Journal of Industrial and Management Optimization (JIMO)*, vol. 13, no. 5, pp. 1–18, Jan 2017.

[87] ——, "OPS/OBS scheduling algorithms: incorporating a wavelength conversion cost in the performance analysis," in *Proceedings of the IEEE 32nd International Performance Computing and Communications Conference (IPCCC)*, Dec 2013, pp. 1–9.

[88] ——, "Scheduling in optical switching: Deploying shared wavelength converters more effectively," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2014, pp. 3418–3424.

[89] ——, "Void-creating algorithm in OPS/OBS: mind the gap," in *Proceedings of the AIP International Conference on Numerical Analysis and Applied Mathematics (ICNAAM)*, vol. 1648, Sep 2014, pp. 1–4.

[90] ——, "Fill the void: Improved scheduling for optical switching," in *Proceedings of the 27th IEEE International Teletraffic Congress (ITC)*, Sep 2015, pp. 82–88.

[91] ——, "Optical switching for variable size packets: Improved void filling through selective void creation," in *Proceedings of the 11th International Conference on Queueing Theory and Network Applications (QTNA)*, Dec 2016, pp. 1–8.

[92] M. D. I. Research, "Optically switched networking," Nov. 2005. [Online]. Available: https://www.cl.cam.ac.uk/teaching/2005/DigiCommII/intel-guest.ppt

[93] C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an optical internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69–84, Mar 1999.

[94] M. Yoo, C. Qiao, and S. Dixit, "The effect of limited fiber delay lines on QoS performance of optical burst switched WDM networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, vol. 2, June 2000, pp. 974–979.

[95] ——, "QoS performance of optical burst switching in IP-over-WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2062–2071, Oct 2000.

[96] C. M. Gauger, "Dimensioning of FDL buffers for optical burst switching nodes," in *Proceedings of Working Conference on Optical Network Design and Modelling (ONDM)*, Feb 2002.

[97] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838–1851, Oct 2000.

[98] S. L. Danielsen, B. Mikkelsen, C. Joergensen, T. Durhuus, and K. E. Stubkjaer, "WDM packet switch architectures and analysis of the influence of tunable wavelength converters on the performance," *Journal of Lightwave Technology*, vol. 15, no. 2, pp. 219–227, Feb 1997.

[99] H. Feng, E. Patzak, and J. Saniter, "Size and cascadability limits of SOA based burst switching nodes," in *Proceedings of 28th European Conference on Optical Communication*, vol. 3, Sep 2002, pp. 1–2.

[100] H. Buchta, E. Patzak, J. Saniter, and C. Gauger, "Limits of effective throughput of optical burst switches based on semiconductor optical amplifiers," in *Proceedings of Optical Fiber Communications Conference*, vol. 1, Mar 2003, pp. 215–217.

[101] J. Gabriagues and J. Jacob, "OASIS: A high-speed photonic ATM switch: results and perspectives," in *Proceedings of 15th international switching symposium (ISS)*, Apr 1995, pp. 457–461.

[102] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: a switch with large optical buffers for packet switching," *Journal of Lightwave Technology*, vol. 16, no. 10, pp. 1725–1736, Oct 1998.

[103] D. K. Hunter, D. Cotter, R. B. Ahmad, W. D. Cornwell, T. H. Gilfedder, P. J. Legg, and I. Andonovic, "Buffered switch fabrics for traffic routing, merging, and shaping in photonic cell networks," *Journal of Lightwave Technology*, vol. 15, no. 1, pp. 86–101, Jan 1997.

[104] F. Forghieri, A. Bononi, and P. R. Prucnal, "Analysis and comparison of hot-potato and single-buffer deflection routing in very high bit rate optical mesh networks," *IEEE Transactions on Communications*, vol. 43, no. 1, pp. 88–98, Jan 1995.

[105] F. Masetti, M. Sotom, D. de Bouard, D. Chiaroni, P. Parmentier, F. Callegati, G. Corazza, C. Raffaelli, S. L. Danielsen, and K. E. Stubkjaer, "Design and performance of a broadcast-and-select photonic packet switching architecture," in *Proceedings of European Conference on Optical Communication*, vol. 3, Sep 1996, pp. 309–312.

[106] M. J. Karol, "A shared-memory optical packet (ATM) switch," in *Proceedings of the 6th IEEE Workshop on Local and Metropolitan Area Networks*, Oct 1993, pp. 205–211.

[107] Y. Nakahira, H. Inoue, and Y. Shiraishi, "Evaluation of photonic ATM switch architecture - proposal of a new switch architecture," in *Proceedings of the 15th International Switching Symposium (ISS)*, vol. 2, Apr 1995, pp. 128–132.

[108] T. Zhang, K. Lu, and J. P. Jue, "Architectures and performance of fiber delay line buffers in packet-based multifiber optical networks," in *Proceedings of Optical Fiber Communication Conference (OFCC)*, vol. 1, Mar 2005, p. 3.

[109] T. Zhang, "A framework for fiber delay-line buffers in packet-based asynchronous multifiber optical networks (PAMFONET)," *International Journal of Communication Systems*, vol. 25, no. 2, pp. 158–168, Feb 2012.

[110] L. Tancevski, L. Tamil, and F. Callegati, "Nondegenerate buffers: an approach for building large optical memories," *IEEE Photonics Technology Letters*, vol. 11, no. 8, pp. 1072–1074, Aug 1999.

[111] F. Callegati, "Optical buffers for variable length packets," *IEEE Communications Letters*, vol. 4, no. 9, pp. 292–294, Sep 2000.

[112] J. Lambert, B. Van Houdt, and C. Blondia, "Single-wavelength optical buffers: non-equidistant structures and preventive drop mechanisms," in *Proceedings of Networking and Electronic Commerce Research Conference (NAEC)*, Oct 2005, pp. 545–555.

[113] R. Langenhorst, M. Eiselt, W. Pieper, G. Grosskopf, R. Ludwig, L. Kuller, E. Dietrich, and H. G. Weber, "Fiber loop optical buffer," *Journal of Lightwave Technology*, vol. 14, no. 3, pp. 324–335, Mar 1996.

[114] M. Calzavara, P. Gambini, M. Puleo, M. Burzio, P. Cinato, E. Vezzoni, F. Delorme, and H. Nakajima, "Resolution of ATM cell contention by multiwavelength fiber loop memory," in *Proceedings of European conference on Optical Communication (ECOC)*, vol. 2, Sep 1994, pp. 567–570.

[115] A. Rostami and S. S. Chakraborty, "On performance of optical buffers with specific number of circulations," *IEEE Photonics Technology Letters*, vol. 17, no. 7, pp. 1570–1572, July 2005.

[116] A. Liu, C. Wu, Y. Gong, and P. Shum, "Dual-loop optical buffer (DLOB) based on a 3x3 collinear fiber coupler," *IEEE Photonics Technology Letters*, vol. 16, no. 9, pp. 2129–2131, Sep 2004.

[117] X. Wang, I. Kim, Q. Zhang, P. Palacharla, and T. Ikeuchi, "Efficient all-optical wavelength converter placement and wavelength assignment in optical networks," in *Proceedings of Optical Fiber Communications Conference and Exhibition (OFC)*, Mar 2016, pp. 1–3.

[118] A. Kaur, K. Singh, and B. Utreja, "Wavelength converters in optical communication systems," *Engineering Science and Technology: An international Journal*, vol. 3, no. 2, Apr 2013.

[119] T. Durhuus, B. Mikkelsen, C. Joergensen, S. L. Danielsen, and K. E. Stubkjaer, "All-optical wavelength conversion by semiconductor optical amplifiers," *Journal of Lightwave Technology*, vol. 14, no. 6, pp. 942–954, Jun 1996.

[120] V. S. Puttasubbappa and H. G. Perros, "Performance analysis of limited-range wavelength conversion in an OBS switch," *Telecommunication Systems*, vol. 31, no. 2, pp. 227–246, Mar 2006.

[121] J. Yates, J. Lacey, D. Everitt, and M. Summerfield, "Limited-range wavelength translation in all-optical networks," in *Proceedings of 15th Annual Joint Conference of the IEEE Computer Societies: Networking the Next Generation (INFOCOM)*, vol. 3, Mar 1996, pp. 954–961.

[122] H. N. Tan, T. Inoue, J. Kurumida, and S. Namiki, "Cascaded operation of wavelength converter for dual-polarization phase-modulated signal," in *Proceedings of OptoElectronics and Communication Conference and Australian Conference on Optical Fibre Technology*, July 2014, pp. 557–559.

[123] F. Callegati, W. Cerroni, and G. D. Maio, "Power consumption reduction in OPS with wavelength conversion," in *Proceedings of 13th International Conference on Transparent Optical Networks*, June 2011, pp. 1–4.

[124] A. Law, *Simulation modeling and analysis*, 4th ed. New York, USA: McGraw-Hill, 2007.

[125] J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation - international edition*, 5th ed. Essex, England: Pearson, 2014.

[126] W. Rogiest, D. Fiems, K. Laevens, and H. Bruneel, "Modeling the performance of FDL buffers with wavelength conversion," *IEEE Transactions on Communications*, vol. 57, no. 12, pp. 3703–3711, Dec 2009.

[127] F. Callegati, W. Cerroni, and G. S. Pavani, "Key parameters for contention resolution in multi-fiber optical burst/packet switching nodes," in *Proceedings of 4th International Conference on Broadband Communications, Networks and Systems (BROADNETS)*, Sep 2007, pp. 217–223.

[128] F. Callegati, A. Campi, and W. Cerroni, "Fast and versatile scheduler design for optical packet/burst switching," *Optical Switching and Networking*, vol. 8, no. 2, pp. 93–102, Apr 2011.

[129] W. Van Heddeghem, F. Idzikowski, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Power consumption modeling in optical multilayer networks," *Photonic Network Communications*, vol. 24, no. 2, pp. 86–102, Oct 2012.

[130] K. Georgakilas and A. Tzanakaki, "The impact of optical wavelength conversion on the energy efficiency of resilient WDM optical networks," in *2011 13th International Conference on Transparent Optical Networks*, June 2011, pp. 1–4.

[131] W. Rogiest, J. Lambert, D. Fiems, B. V. Houdt, H. Bruneel, and C. Blondia, "A unified model for synchronous and asynchronous FDL buffers allowing closed-form solution," *Performance Evaluation*, vol. 66, no. 7, pp. 343–355, July 2009.

[132] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient channel scheduling algorithms in optical burst switched networks," in *Proceedings of IEEE INFOCOM*, Apr 2003, pp. 2268–2278.

[133] A. Liu, C. Wu, M. Lim, Y. Gong, and P. Shum, "Optical buffer configuration based on a 3x3 collinear fibre coupler," *Electronics Letters*, vol. 40, pp. 1017 – 1019, Sep 2004.

[134] S. Fu, P. Shum, N. Q. Ngo, C. Wu, Y. Li, and C. Chan, "An enhanced SOA-based double-loop optical buffer for storage of variable-length packet," *Journal of Lightwave Technology*, vol. 26, no. 4, pp. 425–431, Feb 2008.

[135] C.-y. Tian, C.-q. Wu, G.-n. Sun, X. Li, and Z.-y. Li, "Quality improvement of the dual-wavelength signals in DLOB via power equalization," *Optoelectronics Letters*, vol. 4, no. 5, pp. 361–364, Sep 2008.

[136] Y. Wang, C. Wu, Z. Wang, and X. Xin, "A new large variable delay optical buffer based on cascaded double loop optical buffers (DLOBs)," in *Proceedings of the Conference on Optical Fiber Communication*, Mar 2009, pp. 1–3.

[137] W. Rogiest, D. Fiems, and J.-P. Dorsman, "Analysis of fibre-loop optical buffers with a void-avoiding schedule," in *Proceedings of Valuetools 2014*, Dec 2014, pp. 1–7.