A Tool to Model Traffic Aggregation in Networks of Reconfigurable Optical ADD/DROP Multiplexers

J.M. Fourneau
PRiSM, Université de Versailles-St-Quentin
45, Av. des Etats-Unis,
Versailles, France
jmf@prism.uvsq.fr

N. Izri
PRiSM, Université de Versailles-St-Quentin
45, Av. des Etats-Unis,
Versailles, France
noiz@prism.uvsq.fr

D. Verchère
Alcatel Lucent Bell-Labs France
Nozay, France
Dominique.Verchere@alcatel-lucent.com

We present a tool built to analyse the performance and optimise the design of an all optical ring network based on Reconfigurable Optical ADD and DROP Multiplexers (ROADM in the following). The analysis is based on numerical analysis of discrete time Markov chains and some heuristics to optimise the network configuration until the Quality of Services requirements are satisfied.

Numerical analysis of Markov chains, channel aggregation, performance evaluation.

1. INTRODUCTION

Wavelength Division Multiplexing (WDM in the following) is now established as a successful technique to provide high bandwidth for backbones, metropolitan and local area networks. But optical networks must be more easily configured and managed to meet the performance requirements. Indeed, future optical networks will also require that the WDM layer delivers advanced functionalities such as dynamic reconfigurability, automatic wavelength provisioning and a reconfigurable bandwidth between sources and destinations. For this reason Reconfigurable Optical Add and Drop Multiplexer (ROADM) appears as a practical solution for cost-effective advanced optical networks and multiple architectures for ROADM has emerged in the last years (see for instance Tang and Alan Shore (2006) and Shankar et al. (2007) and references therein).

Briefly a ROADM can be viewed as a device consisting in three parts: a demultiplexer, a multiplexer and optical Wavelength Selective Switch (WSS in the following) sandwiched in between (see Fig. 1). Here we denote as a channel a wavelength on a particular link. The first f channels entering the ROADM are in transit and they do not functionally

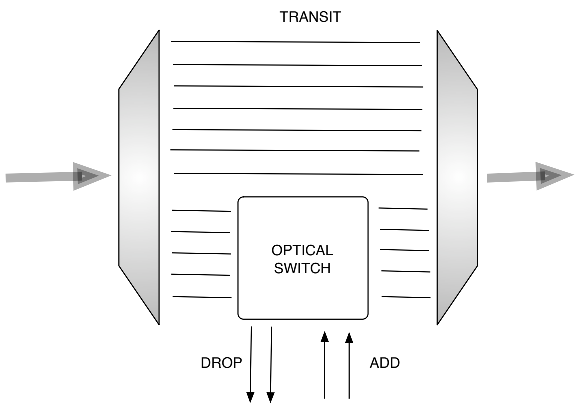

Figure 1: *Schematic Diagram of a ROADM*

enter the optical WSS (they may however physically enter the switch to be optically regenerated but this is out of the scope of our paper). The remaining channels enter the WSS where they are switched. ADD and DROP links are also connected to the WSS. Remember that ADD and DROP links are respectively used to insert or extract optical packets, or SDH (Synchronous Digital Hierarchy) frames from the network.

Besides its considerable practical interest, the design and the optimization of a ROADM based network for a given routing matrix is still an open problem and the literature is quite poor on this topic. Here we consider a possible modification of an existing OADM to increase its capacity and flexibility using a partial O/E/O conversion. The WSS receives L links and we have W wavelength per link. The SDH frames carry packets going to the same destination. In the typical architectures, the frames are associated to an origin and a destination (an OD pair in the following).

We change this assumption here due to the aggregation process we now describe. Some channels are in a new mode denoted as an aggregation mode. SDH frames carried on these channels leave the WSS on the DROP link. Then the packets (IP, IP/MPLS, ATM or Ethernet) are extracted of the SDH frame and they are sent into the buffer waiting for a new SDH frame going to their destination. Note that all the packets carried by an aggregation frame enter the buffer again.

At the same time, all the packets in a drop frame leave the network. The SDH frame which is freed by the packets can receive packets waiting in this input buffer. Clearly, we expect to improve the occupancy of the SDH frames when the traffic is not sufficient to fill all the frames for a particular OD pair. As the traffic for that pair is mixed with the traffic with the same destination we expect that the frames are much more efficiently used after the aggregation. However aggregation has two drawbacks: delays and loss.

First the extraction of the packets out of the frame and the buffering add some delays. While the extraction needs a constant time, waiting for the next available frame in the input buffer requires a random time which is possibly not upper-bounded. Thus we need some analysis of the buffer to check the tail of the delays distribution.

Second, the buffer is finite and as we add more packets entering the buffer we also increase the loss probability. We must check that this probability is below a threshold.

Both performance requirements (delays and losses) are checked by the tool during the analysis. The tool is based on numerical analysis of Markov chain. Such a technique provides accurate results and very efficient process as the chains are small. The tools is based on **XBorne** (see Fourneau et al. (2003) for a description) a software tool we previously developed to build and compare Discrete time Markov chains (DTMC in the following). **XBorne** is a set of C programs and some of them are used for the analysis of queues during the optimisation process.

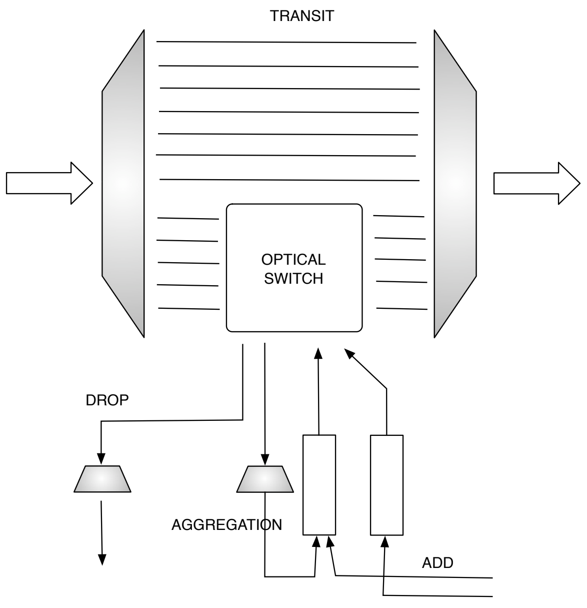

Figure 2: *The aggregation channel in a ROADM*

One must decide for a given matrix of traffic between OD pairs how many channels must be in aggregation mode and which ones are in this mode. Clearly we do not need all the channels to be in aggregation: a packet using a channel in aggregation takes less time to be included into an SDH frame but it takes longer to join its destination. Indeed every aggregation process needs a delay which is larger than the transit time between the demultiplexer and the multiplexer. Thus we have a tradeoff between the occupancy of the SDH frames and the transport time between some OD pairs. We may expect that the first channels in aggregation have a high impact on the occupancy and decrease the delay while the next channels in aggregation slightly change the occupancy and increase the end to end delay. Note also that the end to end delay in an aggregation channel is now random because the packets enter some queues where they wait for an SDH frame. This is a clear difference with the usual architecture where after they have been emitted in an SDH frame, packets have a constant transport delay.

We study aggregation channels to improve the CARRIOCAS testbed presented in Audouin et al. (2007) which is developed by Alcatel and several French organizations. CARRIOCAS is a multi-service network supporting stringent distributed applications in terms of end-to-end delay requirements. SDH frames were used until July 2008 on CARRIOCAS pilot networks but the interfaces of the Photonic Cross-Connects (PXC) were upgraded to enable Ethernet packet frames to be directly mapped over Optical Transport Units (OTU). It has the advantages to reduce the wrapping overhead at the aggregation operations.

Then it is proposed to change SDH frames into Optical Transport Units (OTU). This change does not modify the assumptions and the dimensioning results of the ROADM model. The optical switches has to be configured in order to optimize the fulfilling ratio of the OTUs of each wavelength connection versus the end-to-end transport delay of each OTU between its OD pair (i.e. the ingress PXC and egress PXC configured for the connection).

We have previously obtained in (Fourneau et al. (2009)) some theoretical results. First we have proved using the stochastic comparison approach developed in (Stoyan (1983)) that the population in the buffer is stochastically smaller when we aggregate more channels (even when we add the traffic associated to these channels) taking into account that we also increase the capacity of the servers and that the buffer capacity is infinite. This explain that the insertion into an aggregation channel requires less time when the capacity are summed up and the traffics are aggregated. It is also a theoretical result to explain why multiplexing is efficient. We have also established an analytical formula to obtain the expected number of packets waiting in the buffer when the buffer capacity is infinite. Here we consider a more realistic assumption: the buffer capacity is finite and we try to minimises the buffer size and the number of channel. We present a numerical analysis technique and a set of heuristics to design a network which satisfies the QoS constraint with a small (and sometimes minimal) number of wavelengths. However we do not have proof of optimality.

The rest of the papers is as follows. First in section 2, we give a description of the switch and the network we model and we illustrate the optimisation problem showing the pro and cons of aggregations. Section 3 is devoted to the description of the Markov chains. We also show and illustrate the process we used to obtain the numerical results. In section 5, we present the tool and how we use it. The tool name is **MAROCO** for MARkovian Optical Channel Optimisation. In section 6, we present a very simple case study to emphasis some points of the method.

2. A MODEL FOR A RING NETWORK OF ROADM

Let us now more precisely describe the network, the mechanism and the delays involved in the transport and aggregation processes before we describe the techniques and the tool.

2.1. Topology and Capacity

The network is an unidirectional ring. The other direction is used for protection and is not considered in the model and the tool. The ring is used to interconnect N ROADMs. Thus we assume that a ROADM has an input fibre and an output fibre. Each fibre can use W wavelengths. f of them are in transit mode while the remaining are in ADD/DROP mode (let say d or in aggregation mode (say a)). Thus we have $f + a + d = W$.

2.2. Delays

Let us now consider again the aggregation of packets depicted in Fig. 2 to define the various delays involved in the process. The transit time of a SDH frame on the f transit channels is T_1. When a SDH frame is carried by an aggregation channel, the packets are extracted from the frame. The delay required for this extraction is T_2 unit of times. We assume that $T_2 \geq T_1$ due to some physical constraints: i.e. the transit time is faster than the extraction time.

Electronic packets are gathered in the buffer and they enter the SDH frames in the switch. Insertion of the frame will also requires a delay equal to T_4. T_1, T_2 and T_4 are constant values depending of the switching system. An electronic packet entering the buffer requires a random response time before joining a frame. Let T_3 be this random variable describing the time needed to enter a frame. Note that the frame will be created just before it can be inserted on the ring. We clearly have $T_3 \geq 0$. In the following we show how to compute T_3, its distribution and its expectation.

For the sake of simplicity we assume that T_2, T_3 and T_4 are multiple of T_1. Thus we have a discrete-time systems and the slot time is T_1.

2.3. Effect of aggregation

We assume that for each destination the traffic in aggregation is mixed with the fresh traffic, in the same shared buffer. The buffer is associated with a channel and therefore with a destination and possibly a class of service. The buffer fills $a + d$ frames carried by the same number of channels.

We assume that the packets have the same size and that the number of packets is a discrete random variable between 0 and K. After extraction the electronic packets join the ADD buffer in a batch. This buffer also receives packets from the outside. We assume that the arrivals from the outside also follows an i.i.d. batch process of size Q. The buffer capacity for channel c at ROADM r is B^r_c.

Let us now show how we can compute the end to end delay, the utilisation of the frames, the tail of the delay and many performance measures. First note that we do not take into account the propagation delay which is only dependent of the OD pair and which is not impacted by the aggregation process.

For the sake of completeness we first show how aggregation modifies the distribution of the delays, the sizes of the queues, the utilisation of the frames and the loss rates. For the sake of readability we assume that the traffic is homogenous and we do not need to distinguish between the stations. Thus all the queues are supposed to have the same load and we do not have to index them. Of course such an assumption is not made for the numerical computation we describe in the following sections. Consider an electronic packet in his OD transport. Assume that the path length for this flow $f1$ is L hops.

First assume that the electronic packet does not use an aggregated channel until its destination. The delay at in the input queue is T_3 for entering an SDH frame and T_4 for joining the optical ring. We do not take into account the propagation delay but the frame uses $L-1$ ROADM nodes in transit. Finally when it arrives at its destination, we must add a delay equal to T_2 to extract the electronic packets and send them to the upper layers. Thus we have a total time equal to;

$$T_a = T_3 + T_4 + (L-1) * T_1 + T2$$

Now assume that we add a channel in aggregation to help the traffic from that OD pair to enter the network. We now have a queue with two batches of arrivals and two channels. We have increased the multiplexing. We have proved in (Fourneau et al. (2009)) with this configuration with infinite buffers the new value of T_3 for the packets of $f1$ is smaller in the stochastic sense to the former value of T_3 before aggregation.

Finally consider the channels which have been aggregated. Let $f2$ be the channel which is aggregated with $f1$. We want to understand how the performance of the packets it carried are impacted by the aggregation. Clearly before aggregation the packets of $f2$ need T_1 to transit the nodes where aggregation takes place. Now they need $T_2 + T_3 + T_4$ to be extracted and enter the network again. Clearly the end to end delay for the packets of $f2$ increases. Similarly, the packets of $f2$ enter now two finite queues: the first one which is the same in both experiments and the shared queue after aggregation in the second experiment. Thus the number of rejected packets of flow $f2$ increases due to aggregation queue. Note that the only way to not increase the loss rate is to suppose that the buffer size is infinite. As it is clearly not possible, it is worthy to remark that an aggregation will always increase the loss rates for flow $f2$. Thus we need to check if the new loss rate is below the threshold required by the QoS agreement.

Now let us turn to the occupation rate of the frame. When an empty frame arrives at the ROADM, the packets are emitted. As the capacity of the frame is K packets, the occupation of the frame is the minimum between K and the size of the buffer at the time arrivals of the frame. Thus, changing the arriving process of packets due to aggregation may have an impact on the occupation rate of the frame even if we also increases the channel capacity. We have already observed such a phenomenon during the analysis the ECOFRAME ring (Cadéré et al. (2010)). The occupation of the slots or frames is dependent of the variance of the arrival and departure processes.

The transport delay is the sum of the insertion time, the transit time and the output or extraction time. The end-to-end transit time delay increase with the number of extraction/aggregation operations along the route of the wavelength connection. The insertion time is decreasing with aggregation of channel (see proof in Fourneau et al. (2009)), and the output time is constant. Thus finding a good number of aggregation channel and the precise channels to aggregate makes sense. Note that we speak about "good solution" instead of "optimal solution". Indeed some configurations of aggregation and transit channels may lead to solutions which are not comparable, since the OD pairs involved in the aggregation are not the same. Thus we prefer to define admissible solutions. A solution is the description of all channels for all switches in the network and the list of channel in aggregation at each switch. Such a solution is feasible if the end to end delay for any OD pair in the traffic matrix is smaller than a threshold, and the loss rates are below another threshold. The method we have developed tries to find a feasible solution. The actions we can make consist in aggregating channels and adding memory. Note that the optimisation problem is related to the many end to end delay experienced by the traffics in the network. Clearly if we want to optimize the utilization of the SDH frames, it is sufficient to operate all the channels in aggregation mode as a channel in aggregation is much more efficiently used than a transit channel.

3. BATCH QUEUES WITH CONSTANT SERVICE TIME: A MARKOV CHAIN DESCRIPTION

We model the ADD part of the ROADM as a queue with constant services and a finite or infinite buffer capacity. We begin with some notation. Let C be the channel capacity and B be the buffer size.

The arrivals from the aggregation links follows an iid batch distribution and we also assume that the arrivals of packets from the outside at the ADD link also follows an i.i.d. batch distribution. The description of this distribution is one on the input of the tool. Let A_n be the batch of packet arriving due to traffic aggregation and let F_n be the fresh traffic of electronic packets coming from the outside at time n. For the sake of readability we present a generic model where the index of the queues and channels have been omitted. All the models have the same structure but they have distinct parameters and they are all solved numerically. Remember that the support of A_n is $[0, K]$ and the support of F_n is $[0, Q]$.

The service is discrete and up to K packet enter one frame at each time when the channel uses only one wavelength. Clearly we model a discrete-time system and we consider a discrete-time $Batch/D/C/B$ queue. We further assume that the services take place before the arrivals. Let X_n be the number of packets in the queue. Assuming iid distributions for the batches, we have a simple Markov chain to model the queue:

Property 1 *The population in the ADD buffer is governed by the following equation when the buffer capacity is infinite:*

$$X_n = (X_{n-1} - C + F_n + A_n)^+, \qquad (1)$$

or

$$X_n = min(B, ((X_{n-1} + F_n + A_n) - C)^+), \qquad (2)$$

when B is finite. Remember that if the channel is not operated in aggregation mode then $F_n = 0$. C is a multiple of K. If there is no aggregation, then $C = K$, otherwise we have $C = K * a$ where a is the number of channels still allocated after the optimisation process. Furthermore the number of packets in the SDH frames associated to the channel and departing the queue is:

$$Out_n = min(C, X_n + A_n + F_n).$$

Assume that we model channel i, the distribution of Out is used to obtain the distribution of F_i. Thus, if the channel is aggregated in a next step of the optimisation process, we have already computed the distribution of the batch entering the queue. Note that this is not a fixed point equation because of the routing assumptions. The distributions of the F_i are used to compute the distributions of F_j but there is no directed cycles in the computations.

We assume that the buffer capacity is finite and we analyze the Discrete Time Markov Chain (DTMC) given by Eq. 2 when the batches of arriving customers are i.i.d. Assume that the buffer size is finite and equal to B. We obtain a DTMC with $B + 1$ states which is very simple to analyze with state of the art numerical algorithms (see for instance Stewart's book (Stewart (1995))). As our solver is based on a previous set of programs developed to study Markov chains, we begin with a brief description of this first tool.

4. A BRIEF OVERVIEW OF XBORNE

We use **XBorne** to build the chain and solve the steady-state distribution. **XBorne** is a set of C programs to build, store, manipulate, bound and solve Markov chains (see Fourneau et al. (2003)). More precisely, we have programs for:

- Markov chain generation:

  We proceed by a Breadth-First Search from a chosen initial state (see Haverkort et al. (1999) for such a method for states generated by a Petri net). Here the transitions are described by an evolution equation based on states and events. The states are described by a multidimensional vector, therefore they are included into a Cartesian product. Each transition is associated to an event. An event is described by a probability which may be state-dependent and by the transitions it triggers for all the states. So, the user has to modify 4 functions written in C to specify the Cartesian product including the states, the initial state to perform the visit, the probability of a transition, and the evolution equation which describes all the transitions. Once these functions have been provided, the program is compiled to obtain a new stochastic matrix generation tool which is stored in the model directory.

- Censored Markov chain generation:

  Using the same type of interface (i.e. some C functions used to describe the model) we can obtain a Censored Markov chain. The main difference is that we give an upper bound of the maximal number of states and we impose that some states are not visited (they are taboo). Censored Markov chain are used to control the truncation of very large state space. Only some blocks of the chain will be generated and several strategies have been implemented to obtain or not these blocks.

- Exchange of models:

  Matrices can be exported to Scilab and HBM and Matrix Market format or imported from these representations. We can also generate the DTMC from a continuous time Markov chain or modift the DTMC to improve the accuracy of a stochastic bound as proved in Dayar et al. (2003).

- Manipulation of matrices:

  Matrices may be reordered and transposed. One of the major assumptions of most bounding algorithms is the ordering of the states. First, the rewards must be non decreasing functions of the state-index for strong stochastic comparison. Furthermore, some algorithms based on a partition of states assume that states which belongs to the same set have contiguous indices. Finally, it must be clear that the states ordering is a crucial issue for the accuracy of strong stochastic bounds as shown in (Pekergin et al. (2005)). The accuracy of the bound is directly impacted by the ordering due to the filling of the bounding matrix. Thus we provide a method to reorder the states of the matrix according to the reward. A C function is given by the user. This function describes the reward. It is not necessary for the rewards to be distinct. The ordering is based on the natural ordering on these rewards. The Transpose operation is performed on disk to handle very large matrices. It is necessary to allow column based algorithms to be applied on models.

- Computation of stationary distribution:

  Some simple algorithms like GTH, Gauss Siedel, SOR are available (see Steward's book Stewart (1995) for a description of these algorithms). When needed, state of the art solvers available on the web can be used after transforming the model into a recognised representation like Matrix Market.

- Computation of stationary rewards:

  Once the steady-sate distribution has been obtained it is possible to perform a numerical integration of an arbitrary reward function. Some classical rewards are already available and are computed efficiently.

- Computation of Strong Stochastic Bounds for matrices:

  Many algorithms are provided. They are based on Vincent's algorithm described in (Abu-Amsha and Vincent (1998)), on the lumpability of the bound as considered in (Truffet (2000); Fourneau et al. (2004)) or on building bounds for censored Markov chains as proposed in (Truffet (1997); Fourneau et al. (2007)).

- Computation of element-wise bounds of steady-state probabilities:

  New algorithms for the computation of component-wise bounds recently published (Busic and Fourneau (2010)) have been added.

The algorithms which compute stochastic bounds are not really needed in **MAROCO** because they aim to reduce the size of the chain to obtain a guarantee. Such a solution is not required here as the buffers have small sizes. Typically B is less than 10^4. However we must be very careful about the the accuracy of the numerical computations. We advocate that using **XBorne** for a more application oriented tool is very simple. Similarly it may be easy to use some of the algorithms for bounding distributions we have developed for stochastic model checking, see for instance (Ben Mamoun et al. (2006); Fourneau et al. (2005); Pekergin and Younès (2005)).

5. MAROCO , A TOOL TO DIMENSION ROADM BASED NETWORKS

In this section we present our tool **MAROCO** to improve the number of channels used in the network subject to some QoS parameters. The user only has to provide some inputs: the buffer size, the description of the distribution of arrivals, the traffic matrix, the policy used to optimise its network. Then the transition matrices of the Markov chains are generated and stored on disk. They are finally solved using a classical solver and the statistics and the rewards are computed and returned to the user for the optimisation. The accuracy of the numerical computations must be checked carefully. Thus we use Gauss Siedel algorithm to solve the steady-state distribution because it is proven in (Heyman (1987)) to be more accurate than other iterative algorithms. We first describe the input given by the user before we describe the optimisation processes performed by the tool.

5.1. Description of the Batch of Fresh Customers

Assuming that we have made some measurements on the network, we typically are able to get the average batch size (say m in the following and the maximal size Q for electronic packets and the frame size K. We obtain a system of equations consisting of only two equations (equations 3, 4 while we have $Q + 1$ unknown probabilities. To overcome this problem, we assume three cases of distribution which are extreme values or rather common. Note that it is possible to change these distributions and use more precise measurements. Indeed the complete vector of probability is an input of the model generation in **XBorne** . Note that even if we are able to compute the variance (or the second moment v) of the distribution, this is not sufficient to find all the probabilities and we must make some assumptions about the distributions of the batch. Let p_i the probability that the batch has size i.

$$\forall i, 0 \leq p_i \leq 1$$

$$\sum_{i=0}^{Q} p_i = 1 \quad (3)$$

$$\sum_{i=0}^{Q} i \times p_i = m \quad (4)$$

$$\sum_{i=0}^{Q} i^2 \times p_i = v \quad (5)$$

5.1.1. Highest variability
The distribution corresponds to the most variable distribution according to the icx ordering (an ordering among distributions based on variability). See (Shaked and Shantikumar (1994)) and (Busic et al. (2006)) for the theory and an application to buffer modelling. This distribution is defined by :

$p_0 \neq 0$; $p_Q \neq 0$ and $p_i = 0$ elsewhere.

For this distribution, we get the two probabilities p_0 and p_Q:

$$p_0 = \frac{(Q-m)}{Q}$$
$$p_Q = \frac{m}{Q}$$

5.1.2. Lowest variability
The distribution with lowest variation is also given by (Shaked and Shantikumar (1994)). The distribution is concentrated on the average using one or two non zero value depending on the fact that m is an integer or not. If m is an integer, then the batch has size m with probability 1. If m is not an integer, then the support of the distribution is $\lfloor m \rfloor$ and $\lceil m \rceil$ with adequate probability to obtain an expectation equal to m.

5.1.3. Truncated geometric distribution
Remember that the distribution is given by $\frac{p_{i+1}}{p_i} = a$. Thus

$$p_i = p_0 \times a^i; \; \forall i \quad (6)$$
$$p_0 = \frac{(1-a)}{(1-a^{Q+1})}, \quad (7)$$

and a is computed such that the expectation is equal to m.

Truncated Poisson distribution is also possible.

5.2. Condition for a channel selection

To select the channels to be aggregated or those who can help during transmission, we must verify two conditions:

- We use channels that go to the same destination, passing through the same path between the source and the destination of the overloaded channel.
- The load of these channels must be small. This is defined by a threshold given by the user.

5.3. Selection policy for candidate channel

We can consider three methods for selecting a channel. We recall that a channel is a candidate if and only if it verifies the two conditions already presented in 5.2. These selected channels will be stored as and when the list *ChannelList*.

5.3.1. Lower load
choose the channel on the smallest load among channels Candidate.

5.3.2. Random choice
randomly select a candidate from among those candidates. **if** (Y = number of channels candidates ChannelList = size) **then** $candidatOK = ChannelList[random(1, Y)]$.

5.3.3. First channel
As its name implies, this method selects the first channel satisfying the conditions 5.2.

5.4. Aggregation process for an overloaded channel

The numerical aggregation process proceeds as follows:

1. first make an assignment of the channels to the OD pairs,
2. compute the average end to end delay based on the numerical analysis of the input queue to obtain $E[T_3]$ plus the number of hops (L) which gives $(l-1)T_1 + T_2$.
3. check if the solution is feasible (i.e. for each OD pair, $E[T3] + (l-1)T_1 + T_2$ is smaller than the threshold).
4. If the answer is no, find a candidate channel (say O_1D_1) and try to aggregate several flows using this channel.
5. Compute again the end to end delay for the channel modified in the previous step and

check if the solution is feasible. Compute also the modified loss rate and verify if it is lower than the threshold. If the loss rate is too high, increase the buffer size. Note that increasing the buffer sire results in increasing the end to end delay. Therefore one must iterate until both end to end delays and loss rates are good enough.

6. Continue until the solution is feasible or it is not possible to find a candidate channel for the aggregation.

5.5. Optimisation process for a lightly loaded channel

The candidate channel is a channel with a small load and with a large number of hops. Due to the load, one can aggregate new traffics and due to the length one can cross several ROADMs where aggregation can be performed.

When wavelength λ has been optimized, this means that we managed to find a channel ω through this path with a low charge can also withstand the load of the length of wave λ. In this case, the new load on the wavelength ω will be a combination of the two charges (the old ω more than λ). Whereas the new charge on the length λ will be zero.

If a wavelength is not used in these two directions of communication between an OD pair in the ring, then it will be removed. Indeed, with this method we optimize the number of channels between each pair of OD in the network.

5.6. Operations

We present here the essential steps of the use of the tool and we illustrate the approach in the next section with some simple examples. It must be clear that we have used small models here for the sake of readability. As the tool is based on the numerical analysis of some Markov chains, we can analyse very large networks. The tool has been written with Glade and Gtk+ for the graphical user interface and C for the computations. We do not present details of the GUI.

1. The first step is to ask the user to choose the network topology to be analyzed. We assume here that the topology is a ring as we are mainly concerned with optical rings in the CARRIOCAS and ECOFRAME project. But the tools can be extended to other topologies as well. However, it will use an algorithm to find the different paths between OD pairs whatever the network topology.

   Thus we just have to give a size of the ring. This is given in the start window where the different ring sizes already saved are presented. If none of these sizes is appropriate to the user, he can enter any desired size first then create the traffic matrix (H). The user must select the buffer size and the frame size. Then it must choose the type of probability distribution for the size of the batch.

2. The second step of the problem description occurs at a second window to read the traffic matrix and construct the matrix intensities (routing matrix). In these two matrices we handle the loads on each link transmission. Then, for each node, for each queue associated with a transmission channel we launch $\mathbf{XBorne}$. With $\mathbf{XBorne}$ we get the average latency and packet loss rates of the buffers. In this window we calculate the maximum number of aggregations that can be done, and the maximum number of possible optimizations. We recall that the maximum number of possible aggregations is determined by searching all channels having a load above a certain threshold, while finding the candidate channels associated satisfying the selection conditions. We use this same reasoning to calculate the maximum number of possible optimizations in the ring, except that we look at channels of small load. Knowing that these two calculations are independent, we noticed that the choice of the first mechanism to perform will have an impact on the second.

3. We turn now to the third stage where we propose the user to choose what mechanism he wants to start:
   - Aggregation Mechanism
   - Optimization Mechanism

   If the user starts with the aggregation, management window of this mechanism will open. Then, the user must choose the selection policy of the candidate channel. Then, the user click on the start button to launch the first aggregation to be performed.

   For each channel aggregation or optimization, we calculate the QoS parameters (waiting time and loss rate) using $\mathbf{XBorne}$. If the loss rate increases, we propose to modify the buffer size, then we recalculate these parameters again. If it does not improve, we refuse to do this. However, if the loss rate is improving following the change of the buffer size, we accept this transaction and we'll go to the following (aggregation or optimization).
   We accept any change (aggregation, optimization or elimination) if and only if the QoS parameters are better or unchanged. At the end of all transactions proposed by both our

improvement processes, we show the average waiting time of packet loss rates and routing matrix representing the initial and final states of different channels at different nodes

6. CASE STUDIES

We present two small examples. As the buffer size are small, the numerical analysis is easy and we obtain results much faster than using a simulator. For both models the threshold for aggregation is 0.6 and the threshold for optimisation is 0.2.

6.1. Example 1

We consider an unidirectional ring topology with 6 nodes denoted as S_1 to S_6. We assume that the traffic allows to give only one channel per OD. We assume that the buffer size is 50, the size of the frames is 20, and the maximum size of batch is 5. Thus we have 30 OD pairs. To simplify the first assignment we assume that the same channel is used between S_i and S_j in the first part of the ring and between S_j and S_i in the second part. Thus we only need 15 channels and the assignment matrix to describe the channels in use is:

$$\mathbf{R} = \begin{pmatrix} . & 1 & 2 & 3 & 4 & 5 \\ 1 & . & 6 & 7 & 8 & 9 \\ 2 & 6 & . & 10 & 11 & 12 \\ 3 & 7 & 10 & . & 13 & 14 \\ 4 & 8 & 11 & 13 & . & 15 \\ 5 & 9 & 12 & 14 & 15 & . \end{pmatrix}$$

As this step there is no channel in aggregation mode, and no elimination of channels. Let us now suppose that the traffic matrix is matrix H. Its entries are the load of the channels.

$$\mathbf{H} = \begin{pmatrix} 0 & 0.9 & 0.4 & 0.1 & 0.2 & 0.1 \\ 0.2 & 0 & 0.3 & 0.5 & 0.2 & 0.1 \\ 0.4 & 0.45 & 0 & 0.4 & 0.8 & 0.1 \\ 0.5 & 0.2 & 0.1 & 0 & 0.1 & 0.5 \\ 0.4 & 0.1 & 0.1 & 0.1 & 0 & 0.1 \\ 0.3 & 0.1 & 0.1 & 0.1 & 0.6 & 0 \end{pmatrix}$$

In this example, the number of possible optimizations detected before the start of the process is estimated at 8 and the number of possible aggregations is estimated at 2. We used a geometric probability distribution. The policy consists in selecting the best candidate channel (with the lowest load).

Assume that due to loads in OD pairs S_1S_2 and S_3S_5, the initial assignment is not a feasible solution. Indeed with this large load the average waiting time to enter the network is very large for these flows, since the size of the queue is 49,9. The initial assignment of load to channels is given in Table 1.

num	1→2	2→3	3→4	4→5	5→6	6→1
1	0.9	0.2	0.2	0.2	0.2	0.2
2	0.4	0.4	0.4	0.4	0.4	0.4
3	0.1	0.1	0.1	0.5	0.5	0.5
4	0.2	0.2	0.2	0.2	0.4	0.4
5	0.1	0.1	0.1	0.1	0.1	0.3
6	0.45	0.3	0.45	0.45	0.45	0.45
7	0.2	0.5	0.5	0.2	0.2	0.2
8	0.1	0.2	0.2	0.2	0.1	0.1
9	0.1	0.1	0.1	0.1	0.1	0.1
10	0.1	0.1	0.4	0.1	0.1	0.1
11	0.1	0.1	0.8	0.8	0.1	0.1
12	0.1	0.1	0.1	0.1	0.1	0.1
13	0.1	0.1	0.1	0.1	0.1	0.1
14	0.1	0.1	0.1	0.5	0.5	0.1
15	0.6	0.6	0.6	0.6	0.1	0.6

Table 1: *The load with the first assignment from matrix H.*

num	1→2	2→3	3→4	4→5	5→6	6→1
1	0.9	0.2	0.2	0.2	0.2	0.2
2	0.4	0.4	0.4	0.4	0.4	0.4
3	0.1	0.1	0.1	0.5	0.5	0.5
4	0.2	0.2	0.2	0.2	0.4	0.4
5	0.1	0.1	0.1	0.1	0.1	0.3
6	0.45	0.3	0.45	0.45	0.45	0.45
7	0.2	0.5	0.5	0.2	0.2	0.2
8	0.1	0.2	0.5	0.5	0.1	0.1
9	0.1	0.1	0.1	0.1	0.1	0.1
10	0.1	0.1	0.4	0.1	0.1	0.1
11	0.1	0.1	0.5	0.5	0.1	0.1
12	0.1	0.1	0.1	0.1	0.1	0.1
13	0.1	0.1	0.1	0.1	0.1	0.1
14	0.1	0.1	0.1	0.5	0.5	0.1
15	0.6	0.6	0.6	0.6	0.1	0.6

Table 2: *After one aggregation.*

Let us first improve S_1S_2. Channel 9 is used to connect S_6 to S_2 going through S_1 and its load is very small. It is a good candidate to improve the solution. Thus we make channel 9 an aggregation channel in S_1. Its destination is still S_2. With this aggregation, we found that the loss rate has increased. To avoid losing too much packets we increase the buffer size to 60. But despite this, we reject this aggregation because it degrades the QoS parameters of channel 9.

Now we look at origin-destination pair S_3S_5 on channel 11. First we look for a candidate for aggregation. Channel 8 is used to connect S_2 to S_5 by S_2, S_3 and S_4. Again its load is very low (0.2). We operate channel 8 in aggregation at node S_3 with destination S_5. With this aggregation, we obtained that the loss rate has increased by 50%. Fortunately enough increasing the buffer size to 60, the loss rate

num	1→2	2→3	3→4	4→5	5→6	6→1
1	0.9	0.2	0.2	0.2	0.2	0.2
2	0.4	0.4	0.4	0.4	0.4	0.4
3	0	0	0	0.5	0.5	0.5
4	0.2	0.2	0.2	0.2	0.4	0.4
5	0.1	0.2	0.2	0.2	0.2	0.3
6	0.45	0.3	0.45	0.45	0.45	0.45
7	0.2	0.5	0.5	0.2	0.2	0.2
8	0.2	0.2	0.5	0.5	0.1	0.2
9	**0**	**0**	**0**	**0**	**0**	**0**
10	0.2	0.2	0.4	0.1	0.2	0.2
11	0	0	0.5	0.5	0	0
12	**0**	**0**	**0**	**0**	**0**	**0**
13	0.2	0.2	0.2	0.1	0.1	0.2
14	0	0	0	0.5	0.5	0
15	0.6	0.6	0.6	0.6	0	0.6

Table 3: *After 4 optimizations and 2 eliminations.*

becomes reasonable again. Now we have in S_3 two channels to send packets to S_5 and the average load decreases to 0.5. The load on the various channels for all links are given in Table 2 after these two steps of aggregation. Note that the load of channels is not the same for all links in the case of channel 8 because of the aggregation process.

We reduced the buffer size to 50, then 35 since we have good results with smaller size. Following the execution of the optimization process, we eliminated channel 12 dedicated to S_6S_3 and channel 9 dedicated to S_2S_6. We also improved the channels 3, 11, 15 and 14 on a portion of the ring. For each of these acceptances, we watched two QoS parameters: the average waiting time of packets and the loss rate at the buffers. The distribution of the queue size for any buffer can be depicted at any step of the analysis.

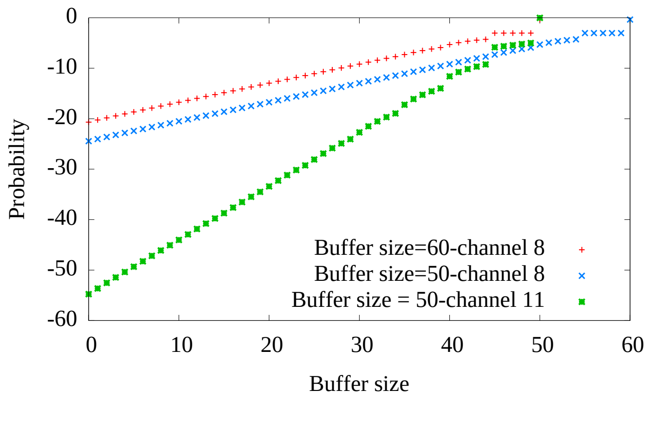

Figure 3: *Probability distribution of three queue sizes.*

Figure 3 illustrates the probability distributions of three queues corresponding to buffers of the channel 8 before and after optimization with channel 11.

6.2. Example 2

We set $B = 9$ and $K = 10$, and we choose a geometric distribution for the batch in this example. We consider a ring of 5 nodes denoted as S_1 to S_5. Thus we have 20 OD pairs and we only need 10 channels. The assignment matrix to describe the channels in use is:

$$\mathbf{R} = \begin{pmatrix} . & 1 & 2 & 3 & 4 \\ 1 & . & 5 & 6 & 7 \\ 2 & 5 & . & 8 & 9 \\ 3 & 6 & 8 & . & 10 \\ 4 & 7 & 9 & 10 & . \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 0 & 0.9 & 0.4 & 0.1 & 0.25 \\ 0.2 & 0 & 0.3 & 0.7 & 0.1 \\ 0.4 & 0.5 & 0 & 0.3 & 0.8 \\ 0.5 & 0.1 & 0.02 & 0 & 0.1 \\ 0.4 & 0.1 & 0.2 & 0.4 & 0 \end{pmatrix}$$

For this example we begin with Table 4 and we try to

num	1→2	2→3	3→4	4→5	5→1
1	0.9	0.2	0.2	0.2	0.2
2	0.4	0.4	0.4	0.4	0.4
3	0.1	0.1	0.1	0.5	0.5
4	0.25	0.25	0.25	0.25	0.4
5	0.5	0.3	0.5	0.5	0.5
6	0.1	0.7	0.7	0.1	0.1
7	0.1	0.1	0.1	0.1	0.1
8	0.02	0.02	0.3	0.02	0.02
9	0.2	0.2	0.8	0.8	0.2
10	0.4	0.4	0.4	0.1	0.4

Table 4: *The load with the first assignment from matrix H.*

optimise the network. We see through traffic matrix H, there are three overloaded channels: 1 (OD pair S_1S_2), 6 (OD pair S_2S_4) and 9 (OD pair S_3S_5). And we can improve channels 10 and 7 connecting OD pairs S_4S_5 and S_5S_2 respectively.

To improve channel 10 connecting S_4S_5, we watched the predecessors of node 4, which are nodes 1, 2 and 3. Channels 9 and 4 were not taken as candidates because their loads are high. Table 5 shows the buffer size (BS0 before optimisation and BS1 after), loss rates (LR0 and LR1 with the same syntax) and decisions associated with each channel having undergone a transformation of the transit mode in aggregation or optimization mode. From table 5, we note that the waiting time of a channel in transit mode are lower.

Let us now consider a second step of optimization, we propose to increase the buffer size (B) because the waiting time has decreased due to optimization channel 6 with channel 7, but the loss rate has increased (0.21). We see that with a buffer size equal

Oper.	NC	BS0	BS1	LR0	LR1	B	OK ?
Optim.							
	7	6.78	1.33	0.38	0.20	9	Yes
	6	2.49	1.33	0.03	0.21	9	No
			1.38		0.13	10	No
			1.45		0.05	12	Yes
Aggreg.							
	6	1.45	1.00	0.05	0.09	12	Yes
	3	3.95	1.17	0.11	0.24	12	No
			1.25		0.05	15	Yes
	7	1.33	1.22	0.2	0.06	15	Yes

Table 5: *Buffer size, loss rate and decisions*

to 12, the loss rate is reasonable (0.05). However, in the second aggregation we find that changing the buffer size is still required. Since we changed it to 15.

Note that this example is a toy problem. Real models will consider larger networks but also bigger buffer and much smaller loss rates. Note that his small loss rates (typically 10^{-9} or 10^{-10}) can only be computed by numerical algorithms or closed form solutions. Simulators, even if they are more flexible, do not allow to compute such values.

num	1→2	2→3	3→4	4→5	5→1
1	0.5	0.2	0.2	0.2	0.2
2	0.4	0.4	0.4	0.4	0.4
3	0.1	0.4	0.4	0.5	0.5
4	0.25	0.25	0.25	0.25	0.4
5	0.5	0.3	0.5	0.5	0.5
6	0.5	0.4	0.4	0.1	0.1
7	0	0.1	0.45	0.45	0
8	0.02	0.02	0.3	0.02	0.02
9	0.2	0.2	0.45	0.45	0.2
10	0.4	0.4	0.4	0	0.4

Table 6: *After 2 optimizations and 3 aggregations.*

Table 6 describes the routing matrix after the execution of aggregations and optimizations channels. We also report the distributions for the size of some buffers to illustrate the approach. Figure 4 shows the probability distributions of three queues associated to buffers of the channel 7 before and after aggregation with channel 9.

7. CONCLUSIONS

We have designed a tool to improve the performance of ROADM. This tool is based on numerical analysis of DTMC and some greedy algorithms to select channels to be aggregated. The efficiency of the aggregation is proved by stochastic comparison arguments. This tool will be used to design core

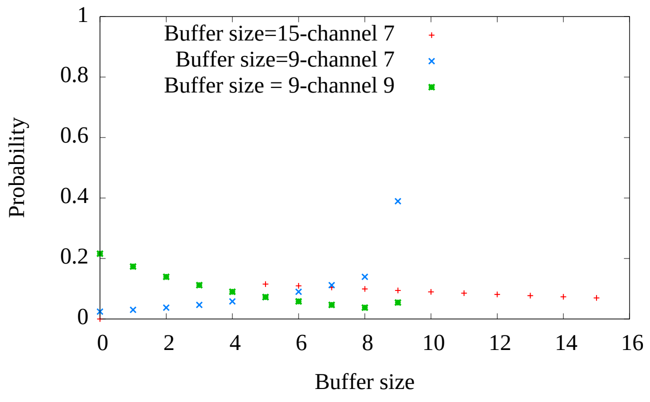

Figure 4: *Probability distribution of three queue sizes.*

networks with aggregation channels. Our tool is open source on the *URL* Izri and Fourneau.

Acknowledgement: this work was partially supported by project System@tic CARRIOCAS and by a GdR RO grant.

8. REFERENCES

O. Abu-Amsha and J. M. Vincent. An algorithm to bound functionals on Markov chains with large state space. In *4th INFORMS Conference on Telecommunications*, Boca Raton, Floride, E.U, 1998. INFORMS.

O. Audouin, A. Cavalli, A. Chiosi, O. Leclerc, C. Mouton, J. Oksman, M. Pasin, D. Rodrigues, and L. Thual. Carriocas poject: An experimental high bit rate optical network tailored for computing and data intensive distributed applications. In *Asia-Pacific Optical Communications conference*, 2007.

M. Ben Mamoun, N. Pekergin, and S. Younès. Model checking of continuous-time Markov chains by closed-form bounding distributions. In *Third International Conference on the Quantitative Evaluation of Systems (QEST 2006), Riverside, California, USA*, pages 189–198. IEEE Computer Society, 2006.

A. Busic and J.-M. Fourneau. A toolbox for component-wise bounds for steady-state distribution of a DTMC. In *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Viginia, USA*, pages 81–82. IEEE Computer Society, 2010.

A. Busic, J.-M. Fourneau, and N. Pekergin. Worst case analysis of batch arrivals with the increasing convex ordering. In A. Horváth and M. Telek, editors, *Formal Methods and Stochastic Models for Performance Evaluation, Third European Performance Engineering Workshop, EPEW 2006, Budapest, Hungary*, volume 4054 of *Lecture Notes* *in Computer Science*, pages 196–210. Springer, 2006.

C. Cadéré, N. Izri, D. Barth, J.-M. Fourneau, D. Marinca, and S. Vial. Virtual circuit allocation with QoS guarantees in the ECOFRAME optical ring. In *the 14th International Conference on Optical Network Design and Modeling (ONDM)*, Japan, 2010.

T. Dayar, J.-M. Fourneau, and N. Pekergin. Transforming stochastic matrices for stochastic comparison with the st-order. *RAIRO Operations Research*, 37:85–97, 2003.

J.-M. Fourneau, M. L. Coz, N. Pekergin, and F. Quessette. An open tool to compute stochastic bounds on steady-state distributions and rewards. In *11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003), Orlando, FL*. IEEE Computer Society, 2003.

J.-M. Fourneau, M. Le Coz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and Applications*, 386:167–185, 2004.

J.-M. Fourneau, N. Pekergin, and S. Younès. Improving stochastic model checking with stochastic bounds. In *2005 IEEE/IPSJ International Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops), Trento, Italy*, pages 264–267. IEEE Computer Society, 2005.

J.-M. Fourneau, N. Pekergin, and S. Younes. Censoring Markov chains and stochastic bounds. In K. Wolter, editor, *Formal Methods and Stochastic Models for Performance Evaluation, Fourth European Performance Engineering Workshop, EPEW 2007, Berlin, Germany, September 27-28, 2007*, volume 4748 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2007.

J.-M. Fourneau, N. Izri, and D. Verchère. Analysis and optimization of aggregation in a reconfigurable optical add/drop multiplexer. In *9th International Conference Next Generation Wired/Wireless Networking, NEW2AN, Russia*, volume 5764 of *Lecture Notes in Computer Science*, pages 120–131. Springer, 2009.

B. Haverkort, A. Bell, and H. Bohnenkamp. On the efficient sequential and distributed generation of very large Markov chains from stochastic Petri nets. In *Proceedings of the The 8th International Workshop on Petri Nets and Performance Models*, pages 12–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0331-4.

D. Heyman. Further comparisons of direct methods for computing stationary distributions of Markov chains. *SIAM Journal of Agebraic Discrete Methods*, 8(2):226–232, Apr. 1987.

N. Izri and J.-M. Fourneau. Maroco tool. Available in `http://www.prism.uvsq.fr/~noiz/MAROCO/Tool.html`.

N. Pekergin and S. Younès. Stochastic model checking with stochastic comparison. In M. Bravetti, L. Kloul, and G. Zavattaro, editors, *Formal Techniques for Computer Systems and Business Processes, European Performance Engineering Workshop, EPEW 2005 and WS-FM 2005, Versailles, France, September 1-3, 2005*, volume 3670 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2005.

N. Pekergin, T. Dayar, and D. N. Alparslan. Componentwise bounds for nearly completely decomposable Markov chains using stochastic comparison and reordering. *European Journal of Operational Research*, 165(3):810–825, 2005.

M. Shaked and J. G. Shantikumar. *Stochastic Orders and their Applications*. Academic Press, San Diego, CA, 1994.

R. Shankar, M. Florjanczyk, T. J. Hall, A. Vukovic, and H. Hua. Multi-degree roadm based on wavelength selective switches: Architectures and scalability. *Optics Communications*, 279(1):94–100, 2007.

W. Stewart. *Introduction to the numerical Solution of Markov Chains*. Princeton University Press, New Jersey, 1995.

D. Stoyan. *Comparaison Methods for Queues and Other Stochastic Models*. John Wiley and Sons, Berlin, 1983.

J. Tang and K. Alan Shore. Wavelength-routing capability of reconfigurable optical add/drop multiplexers in dynamic optical networks. *Journal of Lightwave Technology*, 24(11):4296–4303, 2006.

L. Truffet. Reduction technique for discrete time Markov chains on totally ordered state space using stochastic comparisons. *Journal of Applied Probability*, 37(3):795–806, 2000.

L. Truffet. Near complete decomposability : Bounding the error by stochastic comparison method. *Advances in Applied Probability*, 29:830–855, 1997.