

An OBS architecture for pervasive Grid computing

E. Van Breusegem*, M. De Leenheer*, P. Demeester*, D. Simeonidou[†],

M. J. O' Mahoney[†], R. Nejabati[†], A. Tzanakaki[‡] I. Tomkos[‡]

*Ghent University - IMEC, Gent, Belgium

[†]Essex University, Essex, UK

[‡]AIT, Athens, Greece

Abstract

Grid computing offers high levels of computational, storage and network capacity by bundling and sharing resources through a uniform interface. In its current form, Grids are restricted to only a small application area, and deployment proceeds mainly for specific problems. However, if Grids are to become as ubiquitous as electrical power delivery by making it available to the public at large, several network-related problems will have to be solved first. In this paper, we show the limitations of the currently deployed optical networks, by evaluating future applications for the Grid. Subsequently, we propose a novel architecture based on Optical Burst Switching, and clearly show the advantages of our approach.

I. INTRODUCTION

Over the past few years it has become evident that local computational resources cannot keep up with the demands generated by some users and application classes. Distributed computing, using the concept of a computational Grid, is proposed as the solution. This Grid, named after the analogy with electricity grid, consists of a number of resources (computing

and storage) interconnected through a network, to be shared amongst its users. Large computing endeavours (consisting of one or more “jobs”) are then distributed over this network to these resources, and scheduled to fulfill requirements with the highest possible efficiency.

Grids, as opposed to cluster computing efforts, consist of geographically distributed heterogeneous resources, interconnected through a variety of local area/wide area networks and offer a uniform interface to these resources (i.e. a Grid user can submit jobs to the Grid just as if he/she was handling a large virtual supercomputer). Grids also differ from clusters in that they do not have a central administration point. Instead they consist of resources across multiple administrative domains.

Due to the recent advancement of optical networking technology the feasibility of global distributed computing has significantly increased. Currently, important Grid activities are in the area of high performance computing, optimized for extremely large jobs (or large job batches), both from a bandwidth (constant streams of Gbit/s and up), storage ($\gg$ TB storage) and computational ($\gg$ TFlops) perspective, while the number of jobs is relatively low. Therefore, it is extremely important to optimize the job execution site. In combination with the limited number of job requests, this leads to centralized job management strategies. Current deployment efforts thus focus on dedicated optical infrastructures to support these Grid services, leading to a high performance, optical circuit switched (OCS), network (an example is [1]). Due to the static or semi-static nature of these type of Grids, long-lived wavelength paths between client and Grid resources are usually established. Typical current applications are found in Very Long Baseline Interferometry (VLBI), High Energy Physics (HEP), and High Performance Computing and Visualization [2]. It is not uncommon to have an entire Grid infrastructure dedicated to a single purpose.

Another class of Grids are inspired by the cycle sharing peer to peer applications (such as SETI@HOME), which we call the Peer-to-peer Grids. These Grids attempt to collect

resources from a variety of providers that are managed in a distributed fashion, and are able to handle relatively small jobs. The underlying infrastructure is however the standard Internet, offering no specific support for the Grid applications, resulting in a rather poor (network) efficiency and conflicts with "normal" internet traffic. Their strength lies in the capability of offering a generic service based on distributed resources, thus able to allow a wider variety of applications. Computing tasks are assigned to a Grid resource through a typical request/grant cycle handled by job schedulers. These job schedulers attempt to distribute the demands across the Grid as optimally as possible, based on application profile and resource availability. For a more comprehensive classification, we refer to [3].

This paper explores what future networks will need to provide if the Grid concept is to be extended to the general public. As we will show in the next section, neither class of current grids is capable of scaling up to a large user base, with a wide variety of applications and a plethora of user profiles. This will force network designers to look for alternative, more suitable architectures and infrastructures. No doubt these will be based on optics, and this paper continues by presenting an OBS based architecture that may be very well suited to the task: low processing, with high resource utilization and simple control.

II. MOTIVATION FOR AN OBS ARCHITECTURE, APPLICATIONS AND REQUIREMENTS

A. Introduction

In this section we describe potential applications, and clearly demonstrate that:

- local computing power, even with future desktop pc's, will not be sufficient, thus demonstrating the potential need for widely accessible Grid
- an OCS based optical network for transport between job origin and remote resource location will be very inefficient
- a grant/request based architecture may not be suitable for real-time applications

Since we are looking at a future infrastructure, some assumptions have to be made. For the average home user today the network cannot sustain Grid computing. With a home access bandwidth of only a few Mbit/s, to at most 100 Mbit/s download speeds, and an order of magnitude smaller upload speeds, transmission of jobs would simply take too long compared to local computing power. However, if the current trend holds and bandwidth availability (doubling each year) keeps growing faster than the computing power (at most doubling every 18 months) of an average home user, ubiquitous Grid computing becomes a valid option.

Here we assume a symmetrical access link speed of 2.5 Gbit/s. While this is certainly not available for the average home user today, history shows that in about 20 years such an evolution can be expected. Also, considering the fact that a high-end desktop PC today offers a computational capacity of 3-5 GFlops, we expect this to become in the order of 100 GFlops within the same time frame.

This kind of processing power will make it possible to execute extremely demanding applications (at least by today's standards) on an ordinary desktop PC. However, as the range of potential applications mentioned below will demonstrate, there will many cases where there are even larger computing requirements, making it unfeasible to execute them locally. Figure 1 shows some typical performance requirements for several types of applications. We see for instance, that online gaming should be able to scale to a large number of users, while providing real time operation and sufficient security. The trend, where local computing power is insufficient but a remote, shared resource can fulfill demand, can be reinforced when users step out of the current upgrade cycle of about 5 years. Indeed, most home computers have an average load far below 1%, and only very rarely need their full capacity.

This opens up a window for resource sharing. Multiple users have access via the network to the Grid, where they have shared computing power at their disposal, e.g. through connecting to the site of a provider (containing possibly a cluster or a Grid in itself), having a capacity

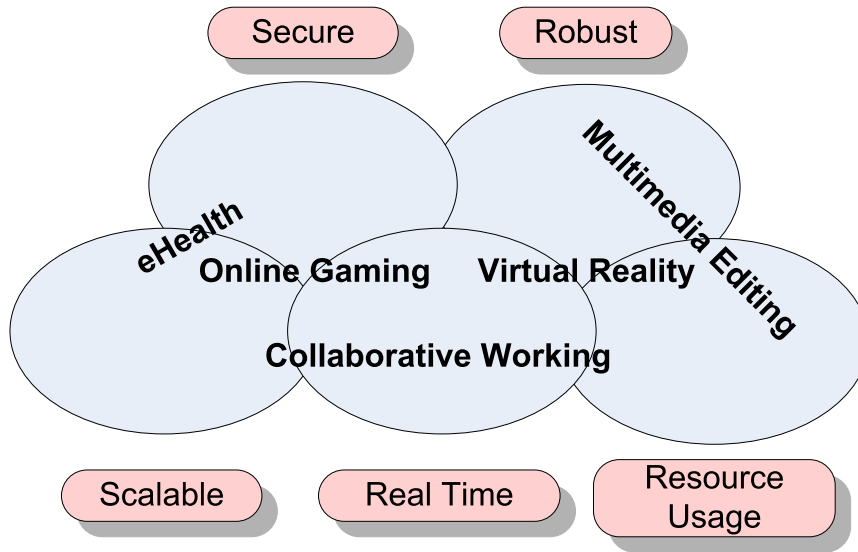


Fig. 1. Grid application requirements

which can exceed the local processing power by a factor of 100-1000, or even more (in case of devices with very limited processing power, like handhelds).

In the following subsections, we present some application scenario's, indicating existing Grid infrastructures will fail to cater for their needs.

B. Offline Application: Multimedia Editing

Video editing applications are already available today, and allow users to manipulate video clips, add effects, introduce new objects, restore old (family) films etc. Advances in recording and visualization technology will demand more computational and storage capacity, especially if we would like to perform the editing within a reasonable time frame (to allow e.g. for feedback to the user, to see whether the manipulation has the intended effect, and if desired change it).

More specifically, 1080p High Definition Television (HDTV) [4] offers a resolution of 1920x1080, which amounts to about 2 MPixel per frame. If we suppose that applying an

effect requires 1 floating point operation per pixel, then processing a one second clip already requires over 500 GFlop. This will take about 5s to complete locally, while execution on a provider's resource should only take 50ms (assuming the capacity of providers is a factor 100 higher). Transmission time of 10s of HDTV video (bitrate 20 Mbit/s or a 25 MB filesize) on a 2.5 Gbit/s access link is 80ms. Although response times do not satisfy real time constraints (for interactive video), it would certainly be unfeasible to use OCS directly in the access network in this case. Indeed, even assuming a relatively modest end-to-end wavelength path set-up time of 100 ms, a significant amount of resources would be wasted. Furthermore, the large amount of wavelength path requests may limit scalability. Thus, for an end-to-end solution, OCS cannot be used.

C. Online Application: Virtual Reality

A virtual environment is typically made up of various objects, which can be described by their shape, size, location, etc. Also, different textures are applied on these objects. The introduction of light sources in the scene makes it possible to mimic real-life locations and environments. A user should not only be able to visualize the environment by choosing different viewing angles, but also interaction with the objects should be possible.

Usually the description of a scene can be realized in limited storage space, the size of a texture being limited to a few Kilobytes. It follows that a scene can be stored in a rather small storage space, typically around a few Megabytes. However, rendering the scene is a different problem altogether: if we demand a performance of 300 million polygons per second, computational capacities as large as 10000 GFlops are required [5]. Clearly, the rendering of these scenes, preferably in real-time, is unfeasible using only local resources. Supposing a user has at its disposal an archive of different scene description, with a requested framerate of 25 frames per second. This amounts to a latency smaller than 40 ms between the submission

of the scene description, and the actual displaying of the scene. Assuming a scene is 2.5 MB in size, we obtain a transmission time of only 8 ms; this leaves us with about 30 ms for processing and retransmission of the final rendering, which should be possible with the given capacities of the (local) resource providers. Considering the delay associated with setting up an optical circuit, OCS can only be used when a user employs the same resource, hereby severely limiting the flexibility of the Grid concept. On the other hand, the lack of adequate QoS in the standard IP protocol makes it near-impossible to meet the real-time constraints.

D. Conclusions

When looking at such requirements the following conclusions can easily be drawn:

- a dedicated network for each application is economically unsound. Although there exist several high bandwidth and computationally intensive applications, building a separate network connected to each user seems unrealistic. The current convergence of phone networks and data networks clearly proves this point.
- Grid service requests will be, for the most part, highly unpredictable. This means that, basically, a dedicated static infrastructure is not the most optimal solution.
- The sheer potential volume of requests makes electronic processing highly complex. In other words, we need to simplify intelligence in the network as much as possible, as well as use optics wherever appropriate to deal with the huge bandwidth requirements.
- In many cases transmission times (job sizes) will be rather short (few 100 μ s to tens of ms). This means that using OCS end to end will prove to be too wasteful, as holding times of wavelength paths will be too small compared to their setup times.
- At the same time, for real time applications, waiting for a grant/cycle to complete simply takes too long. Combined with the high amount of jobs, the current peer-to-peer Grid systems may not be ideal.

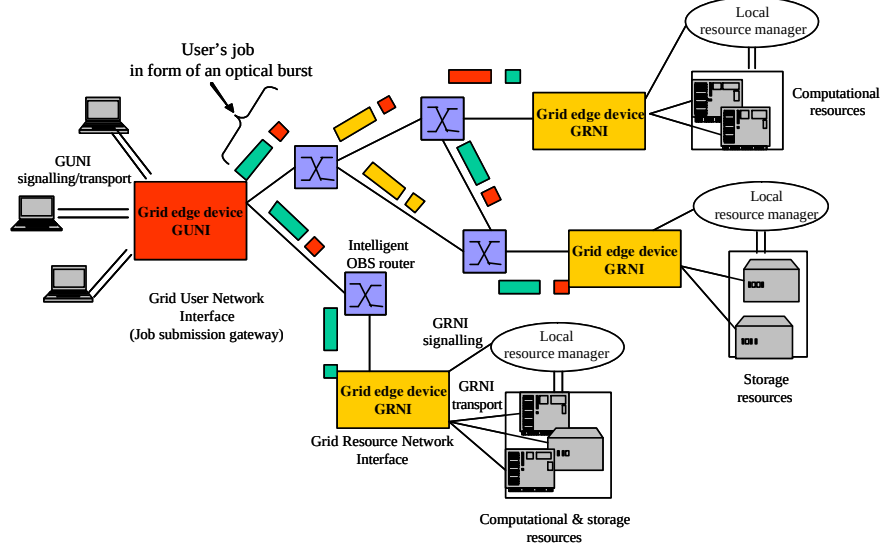


Fig. 2. Generic view of the proposed OBS architecture

These points make the current network solutions (OCS for computational and through the current internet for peer-to-peer) unsuited for providing Grid access to everyone. The dedicated infrastructure will be too wasteful and inflexible, while the request/grant based architecture with electronic management of Grid resources will be too complex. To overcome these problems, a new infrastructure will be required. To that end, we present an OBS based architecture, based on the philosophy of the very successful Internet model.

III. PROPOSED OBS ARCHITECTURE

In this section we describe step by step how and when a job is created in our architecture, how it progresses through the network and reaches its final destination, where it will be processed. First, we will explain the principle of anycast. Using this principle, the next section explains how a job is posted. After job completion, feedback to the user is very likely required. The reporting of this feedback is discussed in the subsequent section.

A. The principle of anycast

If one looks at a Grid from an end users point of view, there will probably be only one major concern: “Do the job I ask you to do, within the requirements”. In contrast, the internet is mainly used in the way: “I don’t care how you reach this information, just get it at this location”. There is a big difference: with a Grid (in general) one does not care where the job is executed, and in fact a number of (physical) sites can provide the required service. Therefore, the destination need not be known prior to submitting ! Indeed, we choose for a scenario here where the end destination is only known as soon as the job arrives, thus the designation “anycast” (as originally proposed in [6]). This is in stark contrast with the current internet, where every packet needs a destination address for it to be delivered. A good example of anycast today may be Freenet [7]. Freenet is an application, where a user shares a portion of its hard drive in a peer-to-peer network. However, the sharer does not know what the exact contents of that portion will be, nor where any file shared by anyone will end up in the network. Unlike other file sharing applications like kazaa, freenet autonomously and continuously optimizes the placement of shared files, without user intervention. The OBS network interconnecting the Grid would work very similar to that concept. Note that anycast in general can work on the application layer [8], as well as the network level [9].

B. how a job is posted

First of all, the user (or application) realizes that a locally generated computing task cannot be reasonably met with the local processing demands, and decides to post it on the Grid to accelerate processing. This job gets transformed in an optical burst, accompanied by a header indicating various parameters (e.g. processing, storage and policy requirements), as in Figure ?? . Note that we make a very important design decision here: in general, one job is contained within one burst. As discussed later in more detail, this will allow easier transport and network

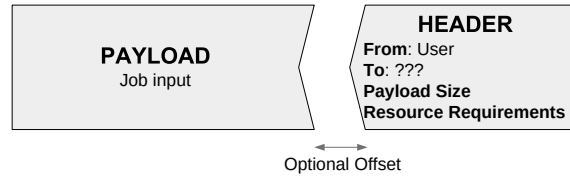


Fig. 3. generic structure of burst containing a job

operation. Of course, exceptions may have to be made for very large jobs which cannot be parallized in smaller ones.

Since we are dealing with an anycast situation, no destination address is needed. The burst is simply sent out into the OBS network. This burst then travels along a link, with the intermediate routers not being notified in advance of its arrival, much like JIT [10] or JET based schemes [11]. When this burst arrives at an intermediate router, it then decides on the fly where to forward the burst to, based on information contained in the preceding header, as also illustrated in Figure 2. This will be a local decision, based on the network and application information available in the intermediate router making the forwarding decision. Examples of such information are link load, delay requirements, estimated free computing power or storage space in a certain outgoing direction, and estimated required computing time or storage space. The end user doesn't specify the network location where the job will be processed, and hence it is implicitly scheduled through its progress in the network. This makes the Grid completely distributed and thus more scalable and robust, while transforming the resource allocation into the earlier mentioned anycast problem (no preset destination).

Each intermediate node belonging to the network connecting Grid resources with the end user goes through the same process, and eventually the burst arrives at a Grid resource. If this resource is able to handle the job contained within the burst, it will process it. If a burst arrives at a Grid resource which is deemed to be able to handle it, but cannot, a deflection

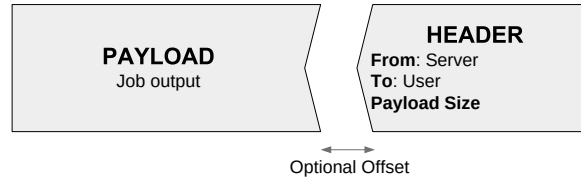


Fig. 4. generic structure of burst containing results

mechanism kicks in: the burst (job) is reposted.

Note that an intermediate router does not need a detailed view of where the resources are and how much capacity they have. As long as there is enough information to push the burst closer to a suitable destination a good decision can be made. This means, for example, that aggregation of the availability information may be a good way to reduce state.

C. returning the completed job

Once the job is completed, its result (e.g. the bitmap of a rendering job) may need to be delivered (most likely to the user who submitted the job). As Figure 4 illustrates, here the asymmetry of the network requirements connecting the Grid becomes very clear: Although the posting of the job is anycast (process it anywhere), sending results back most likely is not. There is a distinct return address, and more traditional forwarding solutions have to be used. In that case, “normal” burst switching with a fixed address can be used. A variety of options may be applicable and will depend on parameters such as: the requirement to report results back, the processing time, the storage availability and the size of results compared to the size and frequency of the jobs posted/processed. For example, in the case of scene rendering, the results reported may be larger in size than the original job, while a calculation of regression analysis based on extensive data may generate only small results.

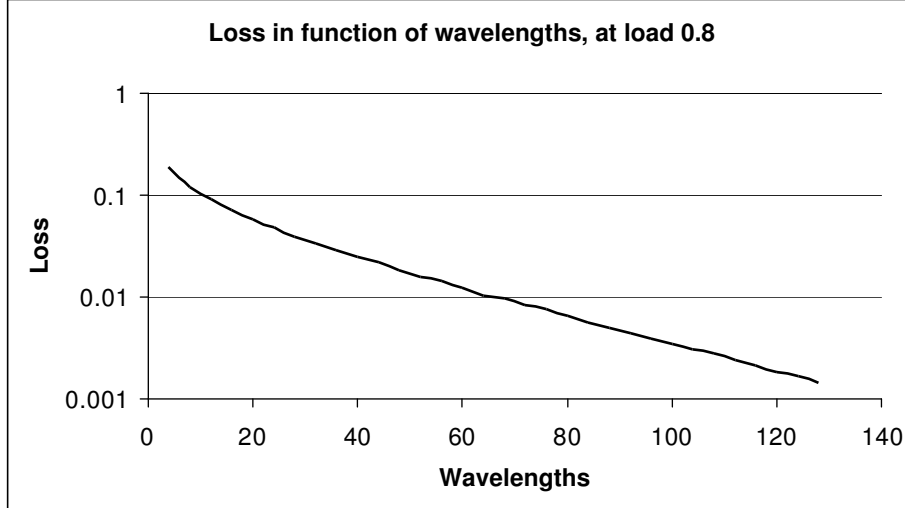


Fig. 5. Evolution of blocking behaviour in function of wavelengths

IV. ADVANTAGES OF USING OBS

An OBS architecture would be a good supporting infrastructure for the future widely available Grid we consider here, especially for the following reasons:

- native mapping between bursts & jobs. As we mentioned earlier, a single job will be mapped into one burst. This is a very straightforward and elegant solution, as a typical job will be in the order of magnitude of a few (tens) of megabyte, which at 2.5 Gb/s can be translated in a burst ranging from a few hundred *us* to several ms. The entire job either gets through, or not at all, which means the transport network can be simplified and waste is reduced (thus we do not consider segmentation [12]. Out of order delivery is also eliminated, and flow control can be simplified. On top of that assured delivery of a job becomes easier, since you only have to deal with 1 burst per job, reducing the requirements of maintaining state. Note that due to the anycast principle, TCP is no longer possible as transport protocol for posting a job: There is no known destination to setup a connection with. Since flow control is not needed (1 job = 1 burst), a simplified

protocol can be used.

- bufferless operation of a switch is not a problem at high wavelength count. A classical problem encountered in OBS and OPS is the highly complex buffering, usually in the form of bulky fiber delay lines. These buffers are necessary as otherwise throughput of the switch is rather low. As Figure 5 however clearly shows, the problem becomes smaller with an increasing wavelength count. It shows the blocking of a single node simulation: It depicts the loss in function of the load for a four fibre by four fibre switch without contention buffers. The simulation assumes uniformly distributed traffic employing Poisson arrival processes of 100 μ s long bursts (more details can be found in [13]). It doesn't consider segmenting, which is fine considering we want a job retained in its entirety, nor deflection routing. As the figure shows, as soon as more than 100 wavelengths in one direction are reached (which may or may not be carried by multiple fibers), an effective throughput of 80% can be reached (loss rate of $< 1\%$). Note that this doesn't consider deflection routing, which may lower the loss another order of magnitude. Thus, considering that the network will require hundreds of wavelengths in the future, bufferless operation may become an attractive option.
- control can be separated - advance sending. This is another classical argument for OBS, but holds especially true here. By decoupling header and payload, the processing delay line can be removed. If then also buffers aren't needed, as the previous argument suggest, we can do with a switch without delay lines at all ! At the same time, dedicating few wavelengths for headers only (i.e. out of band signaling) may prove to be very effective to reduce costly O/E/O conversions. Combined with deflection routing and the principle of anycast, probably a wider range of timing inaccuracies can be tolerated.
- circumvents electronic bottleneck. This is a classical argument in favor of OBS. As electronics evolve slower than optics in terms of speed, which will only become worse,

optics will have to carry most of the burden. Electronics are only used here for what they excel at: intelligent decisions. Pushing data bits around at high speeds is completely, end to end, left to the optics.

- low set-up time. This is mainly an argument against OCS. Since no real setup is involved, reaction time can be decreased drastically which may be a hard requirement for future applications.

As argued above, not only would OBS in general (and the architecture specifically) succeed very well in connecting a very large user base with a worldwide Grid, but due to specific Grid function the OBS switches themselves can become simpler: delay lines can be made obsolete and deflection routing is inherent to the algorithm. For the posting of the job, end-to-end transport is simplified by containing a job within one burst (recall: TCP is not possible).

V. CONCLUSION

As bandwidth availability increases access to remote computing resources will become easier, offering a great opportunity to open up these resources to an extremely large user base (ideally, every citizen). As we showed through some application scenario examples, new network architectures will be required for interconnecting resources and users. This may prove to be a great opportunity for OBS, since its transport format is ideally suited for the requirements posed. Our proposed OBS architecture, tailored to these requirements, has some added benefits, making the choice for OBS as supporting network architecture even more appealing.

ACKNOWLEDGMENT

This work was partly funded by the european commission through IST projects NOBEL, as well as e-Photon/ONe. The flemish government also providing partly funding through

the IWT-GBOU project 010058 "Optical Networking and Node Architectures". Erik Van Breusegem would like to thank the IWT for financial support through his Ph. D. grant.

REFERENCES

- [1] L. L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, P. M. Papadopoulos, "The OptIPuter," *Communications of the ACM*, 46(11):58–67, Nov 2003
- [2] D. Simeonidou (ed.) et al., *Optical Network Infrastructure for Grid*, Grid Forum Draft, Sept. 2003, <http://projects/ghpn-rg/document/draft-ggf-ghpn-opticalnets-1/en/1>
- [3] B. Volckaert, P. Thysebaert, M. De Leenheer, F. De Turck, B. Dhoedt, P. Demeester, *Grid Computing: The Next Network Challenge*, FITCE 2004, to be published in *The Journal of the Communications Network*
- [4] Li Zong and Nikolaos G. Bourbakis, *Digital Video and Digital TV: A Comparison and the Future Directions*, Proc. of the International Conference on Information Intelligence and Systems, Mar 1999.
- [5] Tsuneo Ikedo, *Creating Realistic Scenes in Future Multimedia Systems*, *IEEE MultiMedia*, 9(4):56–72, Oct 2002.
- [6] C. Partridge, T. Mendez, W. Milliken, *Host Anycasting Service*, IETF RFC 1546, Nov 1993
- [7] I. Clarke et al., *Protecting Free Expression Online with Freenet*, *IEEE Internet Computing*, Jan - Feb 2002, pp. 40-49
- [8] S. Bhattacharjee, M.H. Ammar, E. Zegura, V. Shah, Z. Fei, *Application-Layer Anycasting*, Proc. of IEEE Infocom'97, Apr 1997
- [9] E. Basturk, R. Engel, R. Haas, V. Peris, D. Saha, "Using Network Layer Anycast for Load Distribution in the Internet", IBM Research Report (RC 20938), Jul 1997
- [10] J.Y. Wei, J.L. Pastor, R.S. Ramamurthy, Y. Tsai, *Just-in-time optical burst switching for multi-wavelength networks*, 5th Int. Conf. on Broadband Comm. (BC'99), 1999. 339–352
- [11] C. Qiao, M. Yoo, *Optical Burst Switching - A new Paradigm for an Optical Internet*, *Journal of High Speed Networks*, Spec. Iss. On Optical Networking, vol. 8, no. 1, Jan. 2000, pp. 36-44
- [12] V. M. Vokkrane, J. P. Jue, S. Sitaraman, *Burst Segmentation: An Approach for reducing burst loss in optical burst switched networks*, *IEEE ICC '02*, New York, NY, April 2002
- [13] E. Van Breusegem, J. Cheyns, B. Lannoo, A. Ackaert, M. Pickavet, P. Demeester, Implications of using offsets in all-optical packet switched networks, ONDM 2003, Budapest, Hungary, february 2003