An Efficient Superscheduler Architecture and Job Migration Algorithm for Computational Grids over Light-trail WDM Networks: Invited Paper

Ashwin Gumaste[1, 2], Shakesh Jain[3] and Arun K. Somani[2]
[1] Indian Institute of Technology, IIT Bombay, India 400 076
[2] Iowa State University, Ames IA, USA.
[3] Akamai Inc. Santa Clara, CA. USA.

Abstract: **The merging, management and utilization of pervasive, idle computing devices using a dynamic communication infrastructure leads to the concept of computational grids. A computational grid enables enterprises to efficiently use distributed computing entities in a cost-effective setup for emerging compute-intensive applications. From a network perspective, the key requirement for a computational grid is the associated ability to provide dynamic bandwidth (on-demand). An optical networking solution based on the light-trail concept readily adapts to the requirement of dynamic provisioning of bandwidth, in addition to being a low-cost solution and providing for optical multicasting – the key to distribution of *jobs* amongst multiple nodes without the overhead of processing. We investigate a protocol based on a double auction mechanism for converting a light-trail based network into a computational grid using superschedulers at each node. It allows job migration to idle processors. We show how the auction scheme works for bandwidth assignment and in the technique of computing bids, as well as alterations in the virtual topology of the light-trail network. Results of simulation study show the methods' effectiveness.**

I. INTRODUCTION

Computational grids are a direct result of the merger of two premier research and development areas [1] – computation and communication. The latent processing power on desktops, often connected in network LANs and clusters and spread across high-speed networks, presents an opportunity to harvest perishable, valuable, spare processing power for exceedingly complicated tasks. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. The types of applications a grid supports include: distributed supercomputing, high-throughput computing, on-demand computing, data-intensive computing and collaborative computing. These applications may have one or more of the following requirements: high CPU time, great memory needs, *resulting in significant communications*, and requiring *bounded response time*. These applications view the grid as a large meta-computer consisting of multiple processors exchanging data across communication links. From the communication perspective, the central issue is how to efficiently share network bandwidth and reduce the potentially large communication latency between computation centers (nodes) in a grid. From a business perspective, computational grids enable IT-virtualization and its subsidiary benefits that facilitate an enterprise to achieve efficient high-density computing using low-cost and distributed resources. The combination of available desktops and other disperse computing resources over networks and managing such heterogeneous environments is a technological challenge, but the associated low investment is a motivation that enterprises would seldom forgo.

In this article, we propose a scheduling protocol, for grid computing in the wide area, typically for metro area networks – a prime application space for enterprises. Metro area networks for computational grids are based on optical networking paradigms. Optical networks facilitate high bandwidth, resiliency and low-operating costs. Despite the huge bandwidth available at the optical layer, for cost effective deployment, it is necessary that the bandwidth be maximally utilized. Efficient utilization using traffic grooming results in predictable network equipment and lower-cost. For a pragmatic computational grid implementation, we must also ensure dynamism in bandwidth provisioning. While efficient utilization is the primary cost differentiator, dynamism becomes the principle performance metric for converting a transport oriented network infrastructure into a computational grid. Contemporary electronic grooming in optical networks through SONET/SDH, RPR have been proposed and extensively studied but have known to have fundamental cost due to electronics and provisioning bottlenecks.

An optical networking solution that enables efficient utilization as well as provides for dynamic bandwidth provisioning is based on the *light-trail* [2, 3] concept. Light-trails as a transport infrastructure has been considered [4, 5], tested [6, 7] and to our knowledge this research thrust in networks has not been applied to computational grids. Using the light-trails paradigm to reap benefits in a computational grid involves tailoring light-trail infrastructure and associated control protocols for facilitation of grid tasks. In this paper we discuss about the modifications required for light-trails to support a computational grid. We discuss a protocol that facilitates efficient utilization as well as guarantees bandwidth provisioning in light-trails suited for grid applications. The protocol converts grid requests into networking requests and performs the task of scheduling bandwidth on an on-demand yet efficient basis.

In Section II we present light-trails from a conceptual and implementation perspective. The section also showcases the hardware modifications in terms of a superscheduler architecture that is added on to a light-trail node to enable the overlaying of a computational grid on a

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

light-trail WDM ring network. Section III outlines the protocol and abstracts the protocol for superimposition of the computational grid over an optical network. The proposed protocol that combines the superscheduler is also presented here. Section IV discusses numerical results obtained by testing this protocol on a light-trail simulator. Conclusions are presented in Section V.

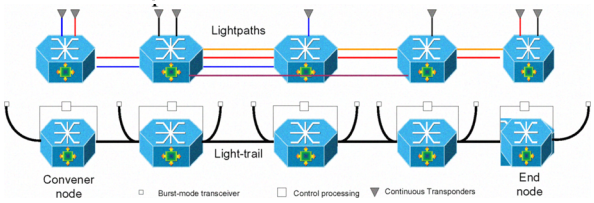

Fig. 1. Conceptual difference between lightpaths and light-trails.

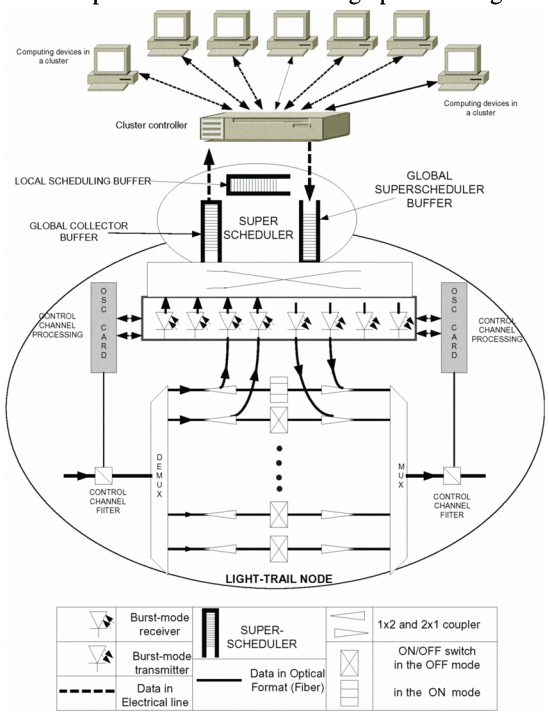

Fig. 2. Computational grids – light-trail node and superscheduler.

II. LIGHT-TRAILS: DYNAMIC AND EFFICIENT OPTICAL NETWORKING AND ADAPTATIONS FOR COMPUTATIONAL GRIDS

In this section we describe the light-trail solution and discuss its adaptation to a computational grid. The first generation of optical networking was based on the lightpath [8] or optical circuit paradigm – providing fixed, static wavelength connectivity (bandwidth) to a source-destination pair. Light-trails [1-2] have evolved from lightpaths. A light-trail is defined as a generalized lightpath [8] such multiple nodes are able to take part in communication along the path. A comparison of lightpaths and light-trails is shown in Fig. 1. A light-trail is a multi-point to multi-point unidirectional wavelength bus. Nodes time-share the bandwidth of the wavelength bus by arbitration through an out-of-band control channel. In light-trail networks there is a distinction between the data plane (wavelength buses) and control plane (out-of-band control channel).

A wavelength bus requires constituent nodes to support certain signal flow characteristics. A node in a light-trail allows optical signal to be dropped-and-continued and also allows passive-addition (without any switching) leading to the formation of the wavelength bus as shown in Fig. 1. The light-trail itself is a unidirectional bus of fixed size, with the head-end being called the convener node and the last node being called the end-node. Setting up and tearing down of the light-trails involves configuration of optical switches and is termed as hard-provisioning. Periodically light-trails can be grown or shrunk. Once a light-trail (bus) is set up nodes communicate to each other by establishing *connections* over the light-trail. Connections are dynamically set up as short duration circuits enabling data flow over the light-trail. Connection setting up and tearing down does not require any optical switch configuration and are controlled using an out-of-band control channel.

In a network, multiple data light-trails have a common control channel. The control channel is slotted in time and is optically dropped and electronically processed at every node (in the light-trail). The processing of control information in the electronic domain at every node makes the control channel synchronous. Typically, the control channel is a SONET/SDH signal (as defined in the ITU-T standard on *optical supervisory channels*) while the light-trail data-plane is based on technologies such as Gigabit Ethernet and 10 Gigabit Ethernet [9, 10].

Node architecture of light-trails: Fig. 2 bottom shows architecture of nodes in a light-trail while modifications to the architecture to facilitate computational grids are shown in the upper (smaller) bubble. The node in Fig. 2 consists of a client side architecture that supports computational clusters, and a network side architecture that supports light-trails. We first describe the network-side light-trail architecture and then describe the adaptations (proposed) as part of the computational grid.

Network side architecture (light-trail node specifics): A fiber line with multiple wavelengths (WDM) enters the node. The control channel is extracted from the fiber by a thin film (control channel) filter. The remaining (data) channels are demultiplexed by an Arrayed Waveguide Grating (AWG) into constituent wavelengths. Each wavelength goes through a Light-trail Optical Retrieval Section (LORS) that gives access of the wavelength to that node. Each LORS consists of two optical couplers separated by a slow optical ON/OFF switch. Couplers are passive devices that enable optical signal to be split or combined in a predetermined ratio. The two couplers in the LORS are in 1x2 and 2x1 configuration enabling the node to access and transmit (passively) data on the light-trail.

Trailponders: Since the nodes in a light-trail time-share the wavelength bus, conventional transponders are unable to transmit and receive data. To facilitate fast turn-ON/turn-OFF and to queue data when a node is not

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

transmitting, transponders in our system have specific burst-mode operation [11]. An implementation of such devices is found in [9, 10]. The burst-mode transponders (trailponders) are critical for the functioning of light-trails.

Client side architecture: The client side of the node is connected to processing entities enabling them to submit jobs for processing. Jobs are assumed to be adaptively divisible. The client side architecture consists of two elements: a cluster controller and a *superscheduler*. The cluster controller is connected to each of the processors at the node and it functions like a *1*x*m* switch (where *m* is the number of processors in the cluster). It is also responsible for scheduling of jobs within the cluster (locally) as well as sending and receiving jobs over the optical network (globally). To determine whether a submitted processing job is processed locally or globally the cluster controller uses a *superscheduler* subsystem. The superscheduler enables the cluster controller to decide where to schedule a particular job. The superscheduler functions between the client side and the network side. It consists of three buffers.

The first buffer is called the *local scheduling buffer*: it contains jobs that are to be processed locally.

The second buffer is called the *global scheduling buffer*: it stores jobs that are to be processed globally (to be sent to other nodes through the optical network).

The third buffer is called the *global collector buffer*: it stores jobs that the node receives from other nodes through the optical network. Stored jobs are then locally processed.

Working of the superscheduler within the light-trail system: A processor upon generating a job sends it to the cluster controller. The cluster controller stores this job either in the local scheduling buffer or global scheduling buffer depending on two factors: size of the job and available processing power in the cluster. If the job cannot be processed in the cluster, it is then stored in the global scheduling buffer. The global scheduling buffer will then send the job to clusters in other nodes via the optical network. The scheduling aspect is managed by the protocol.

The third buffer in the superscheduler is called the global collector buffer. When processors in the cluster are idle, they request for jobs from other nodes in the network. The proposed scheduling algorithm fetches jobs from other clusters via the optical network. These jobs are first stored in the global collector buffer and then transferred to the local scheduling buffer.

The central aspect of the computational grids over light-trail model is the protocol that enables jobs to *migrate* from a node to other node(s) enabling efficient utilization of distributed processing entities across the entire network. While utilization of processors is an important performance metric, it is also critical that the underlying bandwidth be well utilized, failing which we would require exorbitantly large node and buffer sizes especially at high-loads. Finally the protocol also ensures dynamism – low turn-around time that enables job migration to happen within the stipulated latency requirement of the computation tasks. The protocol is based on a *double auction mechanism* enabling *process surplus* and *process deficient* nodes to connect to each other. At another level the protocol also is responsible for configuration of the virtual topology of light-trails – which we term as the recourse model. The virtual topology configuration entails setting up, tearing down and dimensioning of light-trails. We now describe these two aspects of the protocol and then in the next section describe in detail the working of the protocol.

Process-deficient to process-surplus node connectivity: The protocol uses a double auctioning mechanism for arbitration of bandwidth to connect nodes that are process deficient to those that are process surplus. Double auctioning is done at a light-trail level. Each light-trail is assumed to have a controller node that arbitrates bandwidth and enables set up and tear-down of connections in the light-trail. Process surplus and process deficient nodes send bids reflecting their surplus/deficiency processing power to the controller of the light-trail. Bid calculation involves assigning a numerical value to reflect the level of surplus or deficiency of processing power at that given time.

Virtual topology configuration: A node in a light-trail may not be successful in being allocated the desired bandwidth or latency, or there may exist other light-trails connecting the same set of process deficient nodes to process surplus nodes. In such cases, it is important to reconfigure the virtual topology of light-trails. Reconfiguration of the topology allows nodes to maintain high utilization high while meeting the job access time requirements (latency) in the network. As the load imposed on the system grows, so does the number of light-trails in order to maintain a low queueing penalty (in the global buffer). We propose methods to configure the virtual topology by introducing a concept of *phishing*. Phishing involves nodes to advertise their surplus or deficient processing levels to other nodes in the network. Nodes with surpluses processing levels connect to nodes with deficient processing levels through the control channel, and form new light-trails. Likewise light-trails whose utilization falls below a certain threshold are torn down, enabling the affected nodes (of the torn-down light-trail) to phish for an existing light-trial. If however, the phishing does not result in success, then the nodes can form a new light-trail.

III. DOUBLE AUCTION AND RECOURSE ORIENTED PROTOCOL WITH PHISHING

Consider an *N*-node WDM ring network with 2-counterpropagating fiber ringlets each supporting *w* wavelengths. Let each of the *N* nodes support the architecture as shown in Fig. 1 i.e. at the client side support a cluster of computing devices (processors) and at the network side support light-trail technology.

We now define conventions and working of the job migration/assignment mechanism over light-trails. For convenience, we assume a time-slotted system. Time is

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

divided into slots of duration 2-5 ms. The slot size is upper bound by the tolerable latency specification and lower bound by the amount of processing and communication required to be done prior to scheduling data in the network for the next time-slot.

Cluster specifications and convention: We discuss the specification of the clusters at each node in the optical network in terms of their processing power as well as their relationship with the superscheduler.

r_a^i denotes the processing power of processor '*a*' at node *i*.

p_i denotes the number of processors at node *i*.

$\sum_{z=1}^{p_i} r_z$ denotes the processing capacity of the cluster supported by node *i*.

Let $\mu_a^i(t)$ denote if a job is generated at processor *a* connected to the cluster at node *i*.

$$\mu_a^i(t) = \begin{cases} 1 & \text{if a job is generated at processor } a \text{ at } t. \\ 0 & \text{if otherwise.} \end{cases}$$

Let $\omega_a^i(t)$ denote the size of the job generated at processor *a* connected to the cluster at node *i*.

Let $q_a(t)$ denote the residual (available) processing power at processor at time *t*.

Assumption: a job is adaptively and completely divisible. A job is defined in terms of its size rather than the number of processing cycles. This assumption is valid under the following conditions: Load is flexibly divisible and Processors are identical [12].

Then used power at processor *a* time *t* is $r_a^i - q_a^i(t)$. In each time cycle, under normal operation (no new load addition), the processor frees up power and the residual power corresponds to:

$$q_a^i(t) = q_a^i(t-1) + \delta[t-(t-1)] \tag{1}$$

where, $\delta[t-(t-1)]$ is the product of the time-slot duration [*t,t-1*] and δ is the processing power of a processor per unit time.

Let $C_i(t)$ denote the total processing power available in time-slot *t* at cluster supported by node *i* and is given by $C_i(t) = \sum_{a=1}^{p_i} q_a^i(t)$.

Eqn. (2) denotes the total load that enters the network in time-slot *t*.

$$\sum_{i=1}^{N} \sum_{a}^{p_i} \mu_a(t)\omega_a(t). \tag{2}$$

Each node has a superscheduler buffer *S* whose instantiated value $S_i(t)$ represents its size in bits.

Let S_{max} denotes the maximum size of superscheduler.

The load generated enters the superscheduler and then is either sent into the network or sent back to the cluster for processing. The total load going into the superscheduler in time-slot *t* for dissemination locally or globally is given by:

$$\sum_{a=1}^{p_i} \omega_a(t) \tag{3}$$

***Case 1**: Local (Cluster) Scheduling*

$$C_i(t) \geq \sum_{a=1}^{p_i} \omega_a(t) \tag{4}$$

Above relation implies that the total load generated at node *i* in the t^{th} time-slot is less than the available processing power at node *i*, and hence the load can be locally disbursed by the superscheduler. The residual power of the local network is then capable enough to process the job. The task is then to find a set of $\{\alpha_1^i, \alpha_2^i \ldots \alpha_{p_i}^i\}$ s.t.

$$\sum_{a=1}^{p_i} \alpha_a^i = 1 \text{ [13] and}$$

$$\alpha_a^i \sum_{a=1}^{p_i} \omega_a(t) \leq q_a(t). \tag{5}$$

The above problem has been solved in [13]. This also means that there is no requirement for job migration. However, even after assigning the load $\sum_{a=1}^{p_i} \omega_a(t)$ there is residual processing capacity in the cluster and hence the node *i* requests for jobs from other nodes in the optical network. The requesting part is shown in Case 2.

***Case 2**: Global (Network) Scheduling*

$$C_i(t) < \sum_{a=1}^{p_i} \omega_a(t) \tag{6}$$

implies that the load generated in time-slot *t* by the cluster at node *i* is greater than the processing power available at node *i* and hence some portion of the load must migrate to other nodes in the optical network. The migration is done in two steps. A portion of the load that can be processed locally is scheduled within the cluster, while the remainder load is assigned to a global buffer.

1. Assign $\sum_{a=1}^{p_i} \omega_a(t) - C_i(t)$ to the global superscheduler buffer at node *i* for scheduling within the cluster at *i*. The global buffer (superscheduler) then obtains the value:

$$S_i(t) = S_i(t) + \sum_{a=1}^{p_i} \omega_a(t) - C_i(t). \tag{7}$$

2. Assign load corresponding to the value $C_i(t)$ into the local cluster supported by node *i* using the equations for α as obtained in case 1.

When the load generated by the cluster is greater than what the cluster can process, it is sent to the superscheduler for submission to some other node in the optical network. Transversely, if the load generated by the cluster is less than what the cluster can process, then the node requests other nodes to send their loads to this node for processing. The process of sending or receiving load through the optical network is initiated by an auctioning algorithm. Nodes that have excess load float bids requesting nodes with excess processing capacities to accept their load. Likewise, nodes with excess capacity float a different kind

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

of a bid to seek other nodes to submit loads to them for processing. The bidding and connection assignment is done through the light-trail optical network and if needed the algorithm has provision to create new light-trails. The process of light-trail creation (and eventual destruction) reorganizes the virtual topology of the network.

Auction process: To allocate load from one node (cluster) to other node(s) (clusters) we use the dynamic provisioning and optical multicasting property of the light-trail. The unidirectional multipoint implementation of light-trail in [5] implies a node can be a source of a job or a sink or a job. Nodes in each time slot decide based on their superscheduler status whether they would be a source of a job or a sink of a job for the next time-slot. The decision is dependent on whether a node has surplus processing power or needs processing power from other processors in the network as shown in equations (4) and (6). The auction algorithm is tailored for bandwidth assignment in a way such that nodes are bidders for light-trail bandwidth (to send/receive jobs). However, a node may be seller for processing power, or a buyer of processing power depending on (4) and (6). In a given time-slot there can be several nodes vying for the same light-trail bandwidth. These nodes compete with each other. The competition spurs an auctioning mechanism.

We have a single unified method to arbitrate light-trail bandwidth such that nodes that require bandwidth (processing power) receive it in a "competitively fair" manner. We define the competitive fairness as the equality in the ratio of the bandwidth given to each node within a light-trail to the total bandwidth desired by the node. This notion of fairness is valid if the bids are a true reflection of the node's requests. The idea of the auction scheme is to connect process surplus and process deficient nodes in an efficient way. Below, we explain two sub-methods of the auctioning scheme: bid computation and bandwidth assignment. The result of the bandwidth assignment (within a light-trail) also leads to growth (reorganization) of the virtual topology of light-trails in the optical network.

Types of bids: A node in every light-trail is chosen as the light-trail controller, such that it helps in the arbitration process. Typically the controller can be the convener or the end-node of the light-trail. Nodes in the light-trail send their bids to the controller. Bids can either be requests for transmission or requests for reception of data in the next time-slot. Hence if we have a light-trail k with $n(k)$ nodes, then if $n'(k)$ nodes send bids for submitting jobs in the light-trail, then up to $n''(k)=n(k)-n'(k)$ may send bids for receiving these jobs through the light-trail.

Nodes that desire to send a job in a light-trail k send bids called *obids*, while nodes that desire to receive jobs from the light-trail send bids called *ibids*. Bids are sent through the control channel and upon computation of assignment nodes are intimated about their sending and receiving rights in the light-trail for the next time-slot. Bid computation is one of the key issues. Bid computation is a critical process that determines the behavior of the auction algorithm in being able to be fair and convergent in bandwidth assignment to nodes. Bid computation, as shown later on, is a normalized process, and bids take numerical values between 0 and 1 reflecting their requirement of the light-trail in terms of their processing power. We define $obid_{ik}(t)$ and $ibid_{ik}(t)$ as the bids sent by node i in light-trail k at time t desiring to send and receive, respectively, its job in the next time-slot.

Choosing light-trail for ibid and obid: For nodes that have a job to transmit into the network (i.e. job sizes > the cluster residual power) the value of superscheduler buffer $S_i(t)$ is non-zero. Such nodes will send an *obid*. As shown in Fig. 2 there is one superscheduler buffer at every node, while there are a maximum of w accessible wavelengths in each of the two fibers enabling each node to choose from a maximum of $2w$ wavelengths. We now discuss methods for a node to:

1. Determine whether to send an *ibid* or to send an *obid*
2. Determine light-trail to use to send the *ibid* or *obid*.

Each i create an array $ALT_i(t)$: that denotes all possible light-trails *existing* in the network and of access to node i.

Let $k \in ALT_i(t)$. For every $k : k \in ALT_i(t)$, create a set $DALT_i^k(t)$ that gives the set of nodes *downstream* of node i in light-trail k at time t and create a set $UALT_i^k(t)$ that gives nodes *upstream* of node i in light-trail k at time t.

From the definitions of $DALT_i^k(t)$ and $UALT_i^k(t)$ and the unidirectional characteristic of light-trail, a node i sends an *obid* to elements of $DALT_i^k(t)$ or sends an *ibid* to elements of $UALT_i^k(t)$. The decision to send an *obid* or *ibid* is dependent on the value of $S_i(t)$. If, $C_i(t) > 0$ and $S_i(t) = 0$ then node i will send an *ibid* because, it means node i has available processing power that can be dedicated to jobs from other nodes in the $t+1^{st}$ time-slot.

An important point to note is that jobs generated by the cluster at node i in the $t+1^{st}$ time-slot may actually be sent to another node for processing, since the available processing power at the cluster is utilized to get a job from some other node. The *ibid* is sent with an objective to receive jobs from any of the upstream nodes in the set $UALT_i^k(t)$. The choice of light-trail k for sending *ibid* is discussed later. The node sends an *ibid* for only one light-trail out of the possible $2w$. This is because if node i were to send the *ibid* on multiple light-trails, it would possible receive jobs from nodes in the multiple light-trails thereby overflowing the superscheduler buffer.

If $S_i(t) > 0$ then that there are jobs (generated at the beginning of the t^{th} time-slot) in the superscheduler buffer at node i that could not be scheduled into the local clusters (also implying that $C_i(t) = 0$); hence node i will send an *obid* to all the nodes downstream of itself in a *preferred* light-trail k. The choice of light-trail for sending *obid* is discussed later in this section. Upon selection of the

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

preferred light-trail k node i sends *obid* with an objective to secure processing power from nodes that are downstream of itself in the light-trail i.e. in the set $DALT_i^k(t)$.

Computation of obid and ibid: The next issue we consider is the computation of *obid* and *ibid*. The bid is a numerical representation of the requirement (or surplus) of processing power by the cluster. We define instantiation of *obid* and *ibid* as: $obid_{ik}(t)$ and $ibid_{ik}(t)$ depicting the value of the bid at node i in light-trail k in time-slot t. To compute $obid_{ik}(t)$ we define the following parameters:

$\sigma_{ik}(t)$: as the time elapsed since the last successful transmission by node i in light-trail k.

$$\Psi_{ik}(t) = \min_{h=1\ldots S}[\Delta_{S_h} - X_{ik}^h] \tag{8}$$

as the allowable time a packet or batch can wait in the superscheduler $S_i(t)$ before it must be processed in order to meet the parallelism of the load. Inhere by: In (8), Δ_{S_h} denotes the maximum allowable waiting time for process (or service) type h and $S_1, S_2, \ldots S_h, \ldots S_s$ are the s different process (or service) types. Note that X_{ik}^h is the time elapsed since a packet of process type h arrived in the superscheduler $S_i(t)$ and waiting to be allocated for processing. Based on the above definitions we define:

$$obid_{ik}(t) = \max\left[\frac{1}{1+\psi_{ik}(t)/\sigma_{ik(t)}}, \frac{S_i(t)}{S_{\max}}\right] \tag{9}$$

Explanation: obid is a ratio between the values of [0,1]. Either of the two terms on the right hand side of (9) equation passes on its value to *obid*. The first term is a measure of translating process criticality while the second term is a measure of translating the process size (occupancy) in the superscheduler at node i.

The value of $ibid_{ik}(t)$ is computed as a ratio of available processing power of a cluster at a node to the total processing power of the cluster. Thus:

$$ibid_{ik}(t) = \frac{C_i(t)}{\sum_{a=1}^{p_i} r_a^i} \tag{10}$$

Choosing the right light-trail at node i: A node that desires to send an *obid*, it computes the value of $obid_{ik}(t)$ for each $k \in ALT_i(t)$. Then, selects the light-trail that gives the highest value of $obid_{ik}(t)$ for all $k \in ALT_i(t)$ and name this as the *preferred light-trail* for node i at time t:

$$opref_i(t) = \arg\max[obid_{ik}(t)] \tag{11}$$

The node will send bids for seeking transmission bandwidth on this light-trail. The justification for choice in (11) is that, $opref_i(t)$ is the light-trail in which node i can send the highest bid, and hence the best probability of successfully attaining job migration. The assumption is valid *iff* the bid value is a true numerical reflection of the requirement of the node. Upon receiving all the *obids* for a particular light-trail, the controller chooses the node with the highest *obid* as the transmitting node for the next time-slot. The bid that the winning node sends is termed as $mbid_k(t\text{-}1)$ (for the $t\text{-}1^{st}$ time-slot).

For a node that desires to send an *ibid:* To compute which light-trail should a node send its *ibid* on, we have to first consider the assignment of bandwidth. Then, to compute which light-trail to send an *ibid* into, we note as:

$$prefgen_i(t) = \arg\max[mbid_k(t-1)]$$
$$\text{for all } k \in ALT_i(t) \text{ and } UALT_i^k(t) \neq 0 \tag{12}$$

where, $prefgen_i(t)$ indicates the preferred light-trail for node i to send its *ibid* in. This light-trail represents *healthy* bidding in the previous ($t\text{-}1^{st}$) time-slot; i.e. the node that won bandwidth in the previous slot, did so by placing comparatively a large bid with reference to other light-trails. This indicates that the winning node in the light-trail $prefgen_i(t)$ had to overcome other nodes whose own bids were of significantly high value. Hence by sending *ibid* into this light-trail the node has highest probability of finding a corresponding *obid* node that would send a job to this node. Once again, we assume that the bid values are a true reflection of the resources desired by each node for both *ibid* and *obid*. Hence node i naturally will be able to tap a job in the light-trail $prefgen_i(t)$.

Assignment of Light-trail Bandwidth: After computation of bids the next step is the protocol itself. After every time-slot the light-trail controller receives a set of *obids* and a set of *ibids*. If it does not receive either bids, then the controller cannot have a assignment. In such a case, the controller does nothing. In normal operation, the light-trail controller for a particular light-trail k receives a set of *obid*s and *ibid*s. Assuming identical superschedulers of size S_{max} it selects the highest bidder amongst the *obid*s and calls this bidder as the $dump_k(t)$. The node $dump_k(t)$ corresponds to the superscheduler that will schedule (dump) its job into the light-trail k in time-slot $t+1$. $dump_k(t)$ is computed as:

$$dump_k(t) = \arg\max_{i \in k}[obid_{ik}(t)] \tag{13}$$

In addition to deciding which node in the light-trail will send its job, the controller also determines which node(s) in the light-trail are prospective receivers for jobs. It should be noted that more than one node can be prospective destinations for a job (parallel processing). This is achieved through the optical multicast property of the light-trail, whereby when the node that wins transmission right sends its job to *all* nodes downstream of itself in the light-trail. A select group of these downstream nodes can be destinations for the transmitted data (job). Upon deciding the winning node amongst all *obids*, the controller sends back two messages to every node in the light-trail k.

Message 1: Saying that $dump_k(t)$ is the winner for transmission in time-slot $t+1$.

Message 2: Informing $dump_k(t)$ specifically about who its prospective destination nodes are from within light-

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

trail k to carry about job migration from $dump_k(t)$. The node $dump_k(t)$ then sends data to these nodes (through optical multicasting) by placing MAC address of these nodes at layer 2 level. The message that the controller sends to $dump_k(t)$ has two parts: first containing the prospective destination nodes, and second containing the ratio in which the job is to be sent to the prospective destination nodes. The first part of the message is called a set $collector_k(t)$, while the second part of the message is a set called $collector\text{-}ratio_k(t)$. The $collector\text{-}ratio_k(t)$ contains fractions that correspond to elements in the set $collector_k(t)$ (prospective destinations), such that the sum of these fractions is unity. It is also possible to divide the job based on some priority distribution considering value of *ibid* into account. The advantage of such a procedure would be the time saved in communication. We however, consider the process of dividing a job in a generic manner proportional to the ratio of the *ibids*.

$$\begin{cases} collector_k(t) = \{\arg[ibid_{ik}(t)], i \in k\} \\ collector-ratio_k(t) = \left\{ \left[\frac{ibid_{ik}(t)}{\sum_{i=1}^{n(k)} ibid_{ik}(t)} \right], i \in k \right\} \end{cases} \quad (14)$$

Macro-management functions: Periodically the light-trails are grown, shrunk or deleted depending upon their utilization levels. We now discuss the method for growing/ shrinking/ deleting or setting up new light-trails.

1. *Light-trail deletion:*
   Define a threshold value TH_{LOW} such that:
   if $mbid_k(t) < TH_{LOW}$
   and k' exists such that $s, d \in k'$
   perform
   light-trail (k) **deletion**
   endif
2. *Light-trail Advertising:*
   for *every light-trail k*
   advertise *all s,d* **possibilities** *within k such that* $s, d \in k$ *and* $d \in DALT_s^k(t)$
   endfor

IV. SIMULATION MODEL AND NUMERICALS

In the simulation model we consider a 2-fiber 16 nodes WDM ring network (with 40 channels in each direction of communication). Channel rate is assumed to be 1Gbit/s. The control channel is assumed to be time-slotted and runs 155Mbps or OC-3. The control channel is optically dropped (using a filter) and electronically processed at every node. Each fiber is unidirectional in communication. The control channel in both rings is dedicated for supervisory purposes only. Control slot duration is considered to be .001ms and data slot duration is 400 microseconds.

Each of the nodes of the ring has a system buffer which stores the jobs submitted for local as well as global scheduling. Size of the buffer is assumed to be 250 Mbits for simulation purposes. Processing units have a processing storage just enough to store jobs which the processing unit can execute in one data slot duration.

In real world some jobs have a higher precedence over others. Thus in simulation model we divide jobs into 18 classes, which create a precedence relationship amongst the jobs. Thus if there are two jobs in the system buffer one of class type X and one of class type Y such that X < Y, job of class type Y will be scheduled before class type X. Each of the processing unit is also assigned a class type since in real world not all processors can execute all kind of jobs. Processing unit with class type X can execute jobs of type X or higher.

All the processing units act as a job generators. The inter-arrival rate of jobs at each of the processing units follows a Poisson distribution. Table 1 shows the normalized load at each of the computing clusters. Each processing unit's job generation rate depends on the normalized load of the computing cluster of which the processing unit is part of. Size of the generated jobs at each of the processing units also follows Poisson distribution.

Cluster Unit	Normalized Load
C1	0.8
C2	0.8
C3	0.9
C4	0.3
C5	0.8
C6	0.8
C7	0.8
C8	0.4

We define a metric in the simulation to measure the efficiency of the proposed scheduling algorithm. We use this to compare the efficiency of scheduling algorithm as compared to when the scheduling efficiency was not used.

Depending on who is measuring the efficiency of the system, there are different metrics. For example the center system which manages the computing center will be interested in finding out how efficiently computing resources are being used while the end user who submits the jobs is more interested in metrics like Average Response Time and Average Wait time.

$$Average\,\mathrm{Re}sponseTime = \frac{1}{N}\sum_{j\in}^{jobs}(EndTime_j - SubmitTime_j)$$

$$AverageWaitTime = \frac{1}{N}\sum_{j\in}^{jobs}(StartTime_j - SubmitTime_j)$$

Average Response Time of a job is the turnaround time. *Average Wait Time* is the time a job entered the system and when its processing was started.

A good measure of the grid efficiency is how the system buffers are in use at each of the nodes. In Fig. 3 we plot the buffer usage for two cases, with and without global scheduling. With no global scheduling, the buffer usage reached a peak and after a while then there was a buffer at one of the system buffers. In case of global (along with the local) scheduling) the jobs were migrated across nodes and this overall buffer usage was almost constant.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593

Fig. 4 and 5 show the normalized average response time and normalized wait time, respectively, with and without global scheduling when the system is lightly loaded and when it is highly loaded. At low loads there is little difference in the response times and wait times, but communication latencies are an overhead. For the case of high loads there is almost 25% benefit in the average response times and almost 50% improvement in wait times. In Fig. 6 we show the light-trail utilization as a function of network load. It is clear that utilization increases with increase in load for different load sizes.

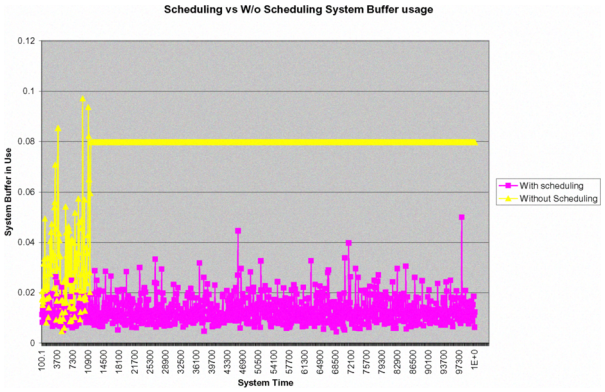

Fig. 3. System buffer usage.

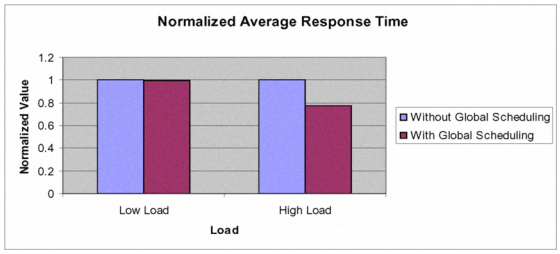

Fig. 4. Average response time vs. load.

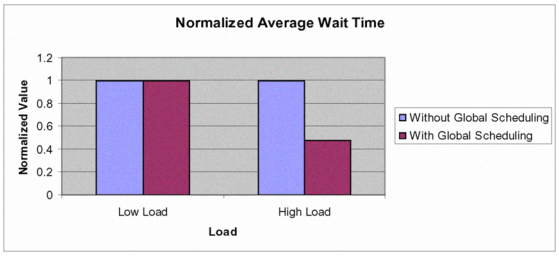

Fig. 5. Average wait time vs. load.

V. CONCLUSION

We have proposed in this paper a superscheduler architecture for computational grids using light-trail WDM networks. The proposed architecture is based on a "double-auction" model where by jobs are sent from process deficient nodes to process surplus nodes. Computation of an auction leads to efficient scheduling of jobs in the metro network – facilitating a computational grid. Performance results are shown for buffer utilization, response and wait times as well as for utilization of the network.

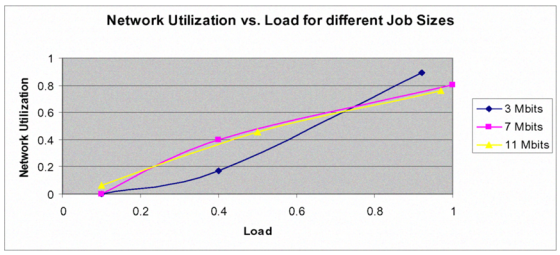

Fig. 6. Network utilization vs. load for different average job size.

Acknowledgements: The research reported in this paper funded in part by NSF projects CNS-0434872 and CNS 0626741, and the Jerry R. Junking Endowment and Cyber Innovation Institute at Iowa State University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or another funding agency.

REFERENCES

[1] Ian Foster and C. Kesselman, "Computational Grids" http://www.globus.org/alliance/publications/papers/chapter2.pdf

[2] A. Gumaste and I. Chlamtac, "*Light-trails: A Novel Conceptual Framework for Conducting Optical Communications*, 3rd IEEE Workshop HPSR 2003 Torino Italy June 2003

[3] J. Fang, W. He and Arun K. Somani, "Optimal Light Trail Design in WDM Optical Networks," in Proc. of ICC 2004, Vol. 3, June 2004

[4] A. Gumaste and I. Chlamtac, "*Light-trails: An Optical Solution for IP Transport*" *Invited paper* Optical Society of America, (OSA) *Journal of Optical Networking (JON)* May 2004. pp 261- 281

[5] A. Gumaste and S. Q. Zheng, "*Optical Storage Area Networks: The Light-trails Approach*," in *IEEE Communications Magazine* March 2005 pp 72-78 Vol. 21. No. 3.

[6] A. Gumaste, P. Palacharla and T. Naito, "Performance Evaluation and Demonstration of Light-trails in Shared Wavelength Optical Networks (SWON)", 2005 European Conf. On Optical Communication (ECOC).

[7] N. VanderHorn, S. Balasubramanian, Mani Mina, A. Somani, "Light-Trail Test Bed for IP-Centric Applications," IEEE Communications Magazine-Special issue on Optical Networking Testbeds: Experiences, Challenges and Future Directions, August 2005, pp. 11-16.

[8] I. Chlamtac, A. Ganz and G. Karmi, "Lightpath com- munications: an approach to high bandwidth optical WAN's," IEEE Transactions on Communications, Vol 40. No. 7. July 1992.

[9] P. Palacharla, A. Gumaste, E. Biru and T. Naito, "*Implementation of Burstponder Card for Ethernet Grooming in Light-trail WDM Networks*," 41st IEEE Int. Conf. on Comm. ICC 2006 Istanbul Turkey.

[10] A. Gumaste, N. Ghani P. Bafna, A. Lodha, A. Agrawal T. Das and S. Zheng, "DynaSPOT: Dynamic Services Provisioned Optical Transport Test-bed - Achieving Multi-Rate Multi-Service Dynamic Provisioning using Strongly connected Light-trail (SLiT) Technology," in *IEEE Journal of Lightwave Technology*, Jan 2008.

[11] Ethernet in the First Mile Standard: IEEE 802.3ah

[12] H. Shan, L. Oliker and R. Biswas, "Job Superscheduler Architecture and Performance in Computational Grid Environments," Proceedings of the 2003 ACM/IEEE Conference on Supercomputing.

[13] Y.-C. Cheng and T.G. Robertazzi,"Distributed Computation with Communication Delay," *IEEE Transactions on Aeropsace and Electronic Systems*, vol. 24, no. 6, November 1988, pp. 700-71

[14] R. Gupta, and A. K. Somani, "Game Theory As a Tool to Strategize as well as Predict Nodes Behavior in Peer-to-Peer Networks," in Proc. of the 11th ICPDS, 20-22 July 2005, Vol. 1, pp. 244-249.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7593
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7593