034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number: D.5.4

Support for Advance Reservations in Scheduling and Routing

Due date of deliverable: 2007-09-30
Actual submission date: 2007-11-15
Document code: Phosphorus-WP5-D5.4

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
Rheinische Friedrich-Wilhelms-Universität Bonn (UBN)

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract

The main objective of the Phosphorus project is to address technical as well as research challenges to enable on-demand e2e network services, i.e. heterogeneous network infrastructures connecting various high-end resources can be used in a dynamic and adaptive fashion. Phosphorus analysis, enhances, and demonstrates solutions that facilitate vertical and horizontal communication among applications, middleware, existing Network Resource Provisioning Systems (NRPSs), and the proposed Grid-GMPLS Control Plane. These developments are validated and demonstrated in a test-bed with a set of applications, which access new services via a Grid middleware.

Work package WP5 addresses a set of research topics to support the activities of other work packages. In particular WP5 analyses architectures and network services to successfully integrate the network as a first class resource into the Grid. This document focuses on advance reservations in the network domain. Based on the exploration of common and Phosphorus-specific use cases, models of advance reservations in the scope of networking are identified and embedded in a broader context of advance reservations of various Grid resources. Subsequently, detailed studies on particular topics are presented that cover topics like resource management for advance reservations, comparison of centralized and distributed advance reservations architectures, and malleable advance reservations.

List of Contributors

Christian de Waal	UBN
Uli Bornhauser	UBN
Alexander Willner	UBN
Markus Pilz	UBN
Christoph Barz	UBN
Emmanouel Varvarigos	CTI
Panagiotis Kokkinos	CTI
Konstantinos Christodoulopoulos	CTI

Table of Contents

Table of Figures

0 Executive Summary

The main objective of the Phosphorus project is to address technical as well as research challenges to enable on-demand end-to-end network services, i.e. heterogeneous network infrastructures connecting various high-end resources can be used in a dynamic and adaptive fashion.

Work package WP5 addresses a set of research topics to support the activities of other work packages. In particular WP5 analyses architectures and network services to successfully integrate the network as a first class resource into the Grid. This document focuses on advance reservations in the network domain. Based on the exploration of common and Phosphorus-specific use-cases, models of advance reservations in the scope of networking are identified and embedded in a broader context of advance reservations of various Grid resources. Subsequently, detailed studies on particular topics are presented.

In the scope of resource management for advance reservations, an efficient timeslot-based approach is introduced and evaluated. Furthermore, distributed and centralized advance reservations architectures are compared with a special focus on Optical Burst Switching (OBS) technologies. Within the scope of advance reservations for networks, a particular service has been identified that allows for a flexible usage of network resources. Strategies of the so-called malleable advance reservations (MARs) are presented and analysed in the scope of this deliverable.

1 Introduction

The main objective of the Phosphorus project is to address technical as well as research challenges to enable on-demand e2e network services, i.e. heterogeneous network infrastructures connecting various high-end resources can be used in a dynamic and adaptive fashion. Phosphorus analyzes, enhances, and demonstrates solutions that facilitate vertical and horizontal communication among applications, middleware, existing Network Resource Provisioning Systems (NRPSs), and the proposed Grid-GMPLS Control Plane. These developments are validated and demonstrated in a test-bed with a set of applications, which access new services via a Grid middleware.

Work package WP5 addresses a set of research topics to support the activities of other work packages. In particular WP5 analyses architectures and network services to successfully integrate the network as a first class resource into the Grid.

This document deals with the support of advance reservations in scheduling. Here, we focus on the interaction of high-speed network resources requested by a service take. A service take is either an end-user or a scheduling instance that has to orchestrate a set of Grid resources. Firstly, an exploration of common and Phosphorus-specific use-cases is presented in the following chapter 2. Starting from this basis, chapter 3 categorizes and specifies models of advance reservations in the scope of networking. As advance reservations are related to scheduling of jobs, existing models in the literature are presented. Chapter 4 embeds advance reservations in networks in a broader context. The interplay of advance reservations of computational and network resources is regarded. The aspect of resource management for advance reservations is addressed carefully in chapter 5. Particular topics that have been identified in the scope of this deliverable are evaluated in chapter 6. The contribution of this chapter is a comparison of distributed versus centralized advance reservations and the evaluation of various strategies to allow for malleable advance reservations (MARs). Both topics are addressed by means of simulation. The concluding chapter 7 summarizes the work of this deliverable and gives an outlook on next steps in the scope advance reservations for network resources.

2 Advance Reservation Scenarios

Large scale distributed applications and workflows need to access high end clusters of computational resources, storage systems, scientific instruments, visualization and network resources. The access can be performed simultaneously or chronologically coordinated respectively and can include different locations. Examples are distributed simulations and sensor data analysis that use the combined computational performance and data storage of multiple clusters, i.e. the workflow is not kept locally due to disabilities, security, or licensing reasons. Here, advance reservation capabilities and meta-scheduling allow for a coordination of different resources involved for a specific time slice.

To couple resources that span multiple administrative and network technological domains the Phosphorus project focuses on on-demand e2e network services that provide dynamic, adaptive and optimized connectivity across multiple domains. Consequently, we will first focus on use cases for Grid network services. In a second step applications of the Phosphorus test-bed are classified according to their Grid networking use case and analyzed concerning their advance reservation characteristics.

2.1 Grid Networking Scenarios

The Grid High-Performance Networking Research Group (GHPN-RG) [1] of the Open Grid Forum (OGF) [2] identifies in the informational memo [3] a set of use cases for Grid network services. The use cases are distinguished in two different groups: Path-oriented and knowledge-based. The path-oriented use cases are concerned with different types of network connectivity while the knowledge-based use cases deal with the collection and usage of network performance information. In the following we will concentrate on the path-oriented use cases as we see wide area network connectivity as a scarce Grid resource to which access has to be restricted. In order to guarantee a certain level of network QoS during a certain period of time and to allow for co-allocation, the GHPN-RG use cases have to be extended with advance reservation.

2.1.1 Parallel High Performance Computing

Peter Tomsu [3] identifies a trend for companies from monolithic super computers with a high cost of acquisition to a high number of standard PCs which are interconnected by a local area networking technology. As a consequence the author distinguishes between the cluster internal use of local area networks and the use of wide area networks for the connection of multiple clusters. In addition, Tomsu classifies communication requirements of the parallel high performance applications to the access and management of files and inter-process communication. The latter can be further partitioned due to their communication requirements:

(i) Parametric execution of applications which cannot be further parallelised. Different instances of the same applications run on different machines. While there will be in general no inter-process communication, there is a need for pre-staging and post-staging the jobs. The related data transfers may be bandwidth intensive but have no hard latency requirements.

(ii) Communication of loosely coupled applications which can perform their computation will have moderate to high bandwidth requirements but only low latency requirements. Note that there may be a graduation in the degree of coupling.

(iii) Tightly coupled applications with periodic need for communication with other processes require very low latency and high bandwidth network connectivity as the processes may be waiting for the communication messages.

Focussing on computational jobs that span multiple clusters, the type of network connectivity which has to be reserved depends on inter-communication requirements of the processes of the application running on different cluster locations. While there may always be a need for high bandwidth communication, the high latency requirements are highest with tightly coupled applications.

2.1.2 Visualization Sessions

One important instrument for the analysis of data sets is visualization. According to Gigi Karmous-Edwards [3] the analysis of local and remote data becomes more and more important in the field of e-science, medicine, engineering and digital art. These data sets may be as large as several terabytes or peta-bytes. The authors decompose the visualization use case into four components:

(i) The data may be real time or pre-processed. It can be accessed from storage devices, streamed from scientific instruments and sensors, or from an application (e.g. from a simulation).

(ii) Computation, which means the analysis of data as a preparation for the visualization session, which may include a rendering process.

(iii) The display that is used to visualize the data and may be supported by a local rendering engine.

(iv) Interactive commands which allow the scientist to select a section of the data, influence the simulation, or to recalibrate the instruments on a real-time or near-real-time basis.

Data, computation and visualization may be partly or fully collocated or reside at different sites. In addition to high bandwidth requirements depending on the size of the data sets and the resolution of the visualization, an interactive and collaborative environment requires round trip times of 150ms [5]. The author also identifies the need for an advance reservation service as the data acquisition devices and the visualization environments may be one-of-a-kind and have to be co-allocated with other resources, i.e. for collaborative environments with several visualization sites, a common time slot has to be identified.

2.1.3 Pre- and Post-Staging of Data for Jobs and Workflows

The use of distributed resources in workflows with dependencies is described as a use case by Volker Sander in [3]. In order to allow for a timely execution and an efficient usage of resources, the jobs constituting the workflow need to be synchronized. This can be achieved by integrating the capability of advance reservations into resource management systems, i.e. the services provided by such a system can be used by users and (meta-) scheduling systems. An example is a high-throughput file transfer with guaranteed deadlines permitting a synchronization of the pre-staging processes with the subsequent jobs to be executed.

Approaches to achieve these data transport services allowing for a guaranteed delivery of data with a certain deadline is analysed in section 6.3.

2.1.4 Structured Data Dispensation

High energy physics experiments – like conducted at the Large Hadron Collider by CERN – produce huge amounts of data. The amount of data of a single experiment is about 2 peta-bytes per year. This data is further analyzed by research groups around the world. For this purpose, a four tier hierarchical structure for data dispensation has been defined in the EGEE project [4]. Tier 0 is the experiment, tier 1 sites collect a full copy of the experiment data and tier 2 and tier 3 sites only have a subset of the data. Peter Clarke et al. [3] focus on three different use cases for file transfer:

(i) The retrieval of raw data from the experiment to a tier-1 site,

(ii) data reprocessing, and

(iii) file transfers needed for remote job execution.

For the transfer of raw data from a tier 0 to a tier 1 site, there is only a limited time window while the data is being kept on disks. After the migration of the data to tape at the tier 0 site, the retrieval to a tier 1 site will – in addition – involve staging the data to disks. As the backup service to tape is a constant process, there is only a limited time window, where an efficient retrieval is possible.

The data reprocessing is carried out at three different tier 1 sites (each processing 1/3 of the data set). The result of the data processing has to be redistributed to all tier 1 sites by a certain deadline (about two weeks).

User jobs are submitted to a resource broker that assigns the job to a computational element according to a specific cost function. The computational element itself or the job may request data replicas. The selection of the best replica may strongly depend on the network state information given.

To enhance this data transfer process a file transfer service for high priority files may be beneficial that provides guarantees for timely data delivery.

2.2 Advance Reservation for Phosphorus Applications

In deliverable D.3.1 [6] a set of use cases envisioned in the Phosphorus project is described. In the following section, they are analysed according to their networking and advance reservation aspects.

2.2.1 WISDOM - Wide In Silico Docking On Malaria

The WISDOM application allows the researcher to compute millions of compounds of large scale molecular dockings on targets implicated in diseases like malaria. In silico docking enables researchers to compute the probability that potential drugs will interfere with a target protein. The most important factor concerning the networking requirements is the pre- and post-staging, which may include the transfer of between 1 and 2 terabyte of data. A network reservation mechanism with a reliable data transfer service will be beneficial to speed up the process. Currently, the resource selection is done manually by the user.

Network capacity between the site with the input-data – the site hosting the data-base – and all computational sites should be reserved for the time of pre- and post-staging. In the time domain there might be restrictions on the duration and time window for computation. The typical time domain for this application is in the order of several minutes.

The WISDOM application can be seen as an instance of the pre- and post-staging of data for jobs and workflows use case described in section 2.1.3.

2.2.2 KoDaVis – Distributed Collaborative Visualization

The KoDaVis application is concerned with the visualization of large climate related data sets. These data sets with a typical size of about 1 terabyte are pre-computed and stored locally at a supercomputer site. The visualization environments are distributed at local laboratories. Using the KoDaVis application, scientists can collaborate in the exploration of the data set and share a common, interactive view of the climate data. Therefore the user selects the part of the data interactively that has to be streamed from the supercomputing

sites to all visualization environments. There is a parallel data-server that distributes fragments of data selected by the clients, and a collaboration server that synchronizes all clients.

The KoDaVis application can be classified as a form of visualization session described in section 2.1.2. While in the general visualization use case there may be three different types of sites, the KoDaVis application only involves a data site with pre-processed simulation data and a visualization site with local rendering facilities as the applications supports different client types. As KoDaVis is an interactive, collaborative application, it imposes end-to-end delay requirements on the network connectivity. The connectivity pattern is a star with the data site in the centre of the star. The bandwidth requirements are about 1 gigabit per second.

The use of advance reservation allows for the negotiation of common timeframe. This may include personal schedules of scientists, reservation system for visualization facilities and the inter-connecting network. A typical time frame is in the order of an hour during office hours.

2.2.3 TOPS – Streaming of Ultra High Resolution Data Streams

Similar to the KoDaVis application TOPS is a visualization service (e.g. for large scale simulation data). Raw data is transferred between the storage and a visualization service, which may be in close proximity to data due to security concerns, and is kept there for security reasons. The visualization service is then able to transfer a pixel stream to (tilted panel) display without distributing the raw data. Reservations are made for the graphics resources, the network connection between the visualization service, and the display. Corresponding to the KoDaVis application, the user is able to interact with the visualization and change the perspective. Currently, the resource selection is done manually by the user.

From the advance reservation perspective, requirements are similar to the requirements of the KoDaVis application. Reservations will also be in the order of an hour during office hours. However, the bandwidth requirement on the network connections is up to 10 gigabit per second.

2.2.4 DDSS – Distributed Data Storage System

The Distributed Data Storage System (DDSS) is a general data transfer service. It has two different use cases in the scope of the Phosphorus project. In the first use case, GridFTP is used as a protocol to move data between system nodes or sites. GridFTP is used for pre- and post-staging purposes as well as for the transport of intermediate data. While it is usually used without a network reservation service, such a service might guarantee transfer times. This allows for a tighter schedule in cluster jobs as the exact transportation time for the transport of data can be taken into account when constructing a schedule. In addition, a guaranteed network delay may help to speed up the transfer of files. The second use case is the transport of data for backup and archive applications. Such a service typically uses TCP/IP connections between clients and a server application (e.g. IBM Tivoli Storage Manager).

In both cases the communication structures are one-to-one or many-to-one. The size of the data to be transferred is dependent on the application and may vary. The files typically need to be transferred as soon as

possible. In the backup use case, there might also be a limited time window. If the data amount is small, the timeframe and management overhead to schedule a transfer via advance reservations might be unnecessary, i.e. the available capacity suffices. If the file transfer job lasts long enough and has to fulfil a dedicated deadline, a scheduling service is beneficial.

2.3 Conclusions

The scenarios in this chapter are presented with a focus on advance reservation in the scope of networking. Firstly, inter-connections to stream or exchange information at high data rates are identified in workflows of large scale Grid application. A network service should provide connections that can be scheduled beforehand and guarantee application specific QoS in terms of delay and capacity. These advance reservations allow for synchronizing the work to be performed in a distributed environment, i.e. the network assists geographically distributed computational, visualization, and data streaming jobs.

An additional service for various scenarios is the transfer of data for computational tasks. The process of pre- and post-staging jobs has slightly different requirements in contrast to the aforementioned requirements. It is important to guarantee that the data – to be processed – is delivered timely. But neither the end-to-end delay of the connections nor the used capacity is crucial. These types of advance reservations are concerned with the transfer of a data set in a user specified time interval.

Overall, resource management systems that allows for the services mentioned have to provide the capability to allocate network resources for a certain time interval.

3 Advance Reservation Models

In the following chapter an overview of advance reservations models is presented. Firstly, a general classification of jobs that can be submitted to a reservation system is presented. The classification is based on the use cases and scenarios presented in chapter 2 and existent terminology.

3.1 Job Classification

A first step to classify Grid jobs can be done by distinguishing them from fine-grained jobs (or tasks) performed by a multi-tasking operating system, which can be found in computers and workstations. A user submits a job to the Grid that may last for a longer period of time (multiple minutes or hours). Consequently, resource usage is usually admitted by a resource management system, which assists or incorporates a scheduling strategy.

In the scope of Grid computing, jobs can be classified in various application areas. For example, a job describes the execution of an application to start a simulation on a computational cluster. Although a job naturally belongs to a specific domain, similarities between application areas can be identified. This section compares job types present in the literature to the job types identified in Phosphorus project. While the former usually concentrates on computational jobs, the latter deals with network jobs. A particular difference between computational and network jobs is the resource management. While computational resources (e.g. machines, CPUs) might be scheduled as an independent set of resources, network resources have a dependency induced by the topological structure. For example, a user requests a path between two endpoints, which can only be established by a certain subset of links in the network. If applicable, these similarities and differences are discussed.

In the former chapter scenarios for advance reservations in the scope of networking are identified. These include streaming or pipelining of data (e.g. from sensors) and multi-cluster jobs, which require a specific QoS in terms of capacity and/or delay. Furthermore, pre- and post-staging (or pre- and post-processing) jobs to assist computational or storage jobs where identified. Before identifying types of reservations in section 3.3, a classification of jobs based on [7] is introduced.

- **Rigid Job**: The job describes a fixed set of constraints that have to be matched. For example, a computational job needs a specific number of CPUs, a specific software version, or a network job needs a dedicated amount of end-to-end capacity and has to meet a upper delay constraint.

- **Mouldable Job**: These jobs have a certain degree of freedom that can be used by the scheduling strategy to circumvent resource blocking or request rejection. For example, the job can be executed on a variable number of CPUs or the end-to-end capacity can be chosen from a given interval. Of course, the running time of the jobs is affected by the allocated resources. Following [1], a mouldable job's resources are fixed throughout the running time of the job. The time to determine the resources is restricted by the starting time.

- **Malleable Job**: In addition to the flexibility offered by mouldable jobs, malleable jobs are allowed to change their resource usage during execution, e.g. the resource allocation can be adapted due to failures or additional available resources. Malleable jobs are apparent in the domain of computational jobs, where jobs can be migrated on a different number of CPUs, as well as in the domain of network jobs, where a file transfer might be processed by allocating a variable amount of end-to-end capacity.

It is noteworthy that the declared classification of jobs differs in the literature. One example is the introduction of malleable network reservations in the work of Burchard [8]. Burchard's approach of malleable reservations corresponds to the classification of mouldable mentioned above.

3.2 Network Job Constraints, QoS, and Admission Models

Network constraints are often denoted as QoS. The implementation of QoS in networks is again a large field of technical details and research branches. Briefly, QoS can be viewed as "Better than Best Effort" service or a guaranteed service, which reserves a specified amount of resources. Better than Best Effort services are present in packet switched (e.g. DiffServ) and circuit switched networks (e.g. ATM's VBR). Guaranteed services are present in packet switched (IntServ), circuit switched (ATM, ISDN), and converged networks (MPLS, GMPLS). If not otherwise stated, QoS is seen in the sense of a guaranteed service.

In the scope of guaranteed services, it is possible to differentiate *link-constraints* and *path-constraints*. A constraint that has to be fulfilled on every link of a path is denoted as link-constraint. The available bandwidth is a typical link-constraint, as it must be fulfilled on every single link of the entire path. By contrast, the delay is a typical path-constraint, as the compliance can only be verified concerning the entire path. In case of a bidirectional reservation, the requirements in both directions are the same.

With the focus on guaranteed services, an admission model is also implied. A service requestor – either a human user or a higher level Gird service like a meta-scheduler – asks for specific service that can either be admitted, if enough resource are available to guarantee the service, or the request is rejected. Disregarding system failure like link or node outages, an accepted service request is enforced assuredly.

3.3 Types of Reservations

Advance reservation requests contain time related as well as resource related parameters. Depending on the information available, a reservation system has the ability to handle requests more or less efficiently. An example is the specification of a time interval in which a reservation can be scheduled with a certain degree of freedom: Although the point in time is known and needs to be considered as strict deadline, the reservation system might be able to reschedule the allocated resources as long as constraints are met. In the context of network jobs, the reservation system might be able to re-route or re-schedule an end-to-end path in order to allow for the admission of additional requests.

Obviously, there exist applications where the ending time is unknown when the resources are requested. In this case, the user cannot specify the point in time at which the resources are no longer needed. Common phone calls provided over a circuit switched network are a typical example. Here, a classification of reservation types including the associated parameters envisioned in the scope of Phosphorus is described. In the first place, the types of reservations are classified by the time related parameters.

3.3.1 Immediate Reservations

A reservation made to instantaneously use or (at least) to allocate resources is called an *immediate reservation* (IR). Two basic forms of IRs can be differentiated: Either the duration of the resource usage is known in advance or the ending time is not specified by the user or client system towards the reservation system. In the studies [9] and [10] for instance, the authors differentiate between immediate reservations with unknown and known duration also. An IR without ending time must be explicitly terminated by the user or kept alive in the context of soft-state mechanisms.

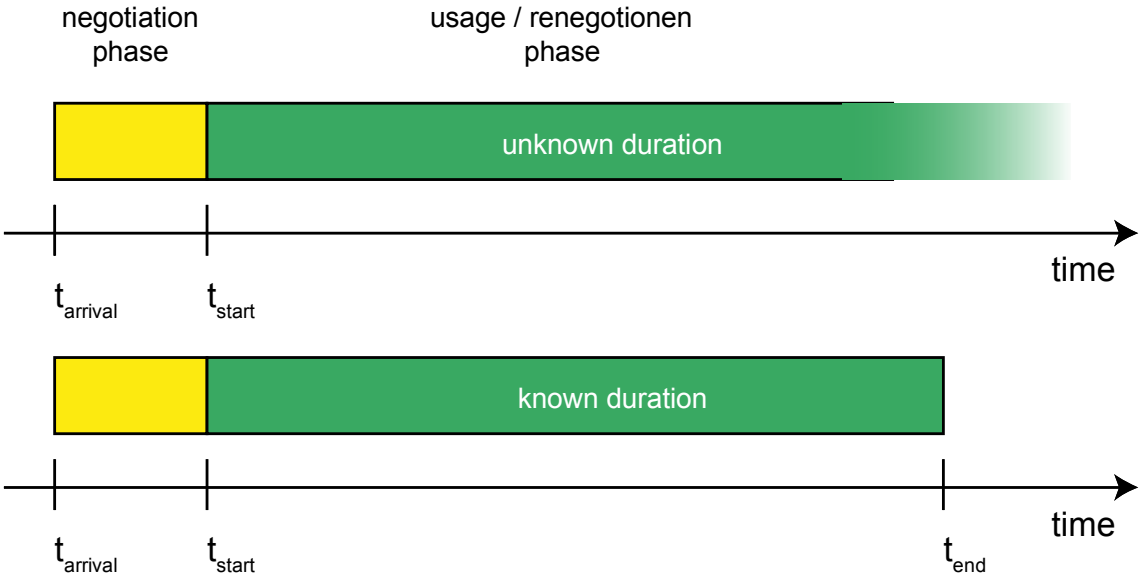

Figure 3-1: Immediate Reservation with unknown and unknown duration

The basic parameters of an IR request with unknown duration only concern the resource parameters. In the network context, this includes the source which identifies the source (e.g. ingress router) of the requested path and the destination (e.g. identifying the egress router). Additionally, a bidirectional-flag can be included, which enables the user to request a bidirectional path. If this flag is set, the order of the routers on the path from source to destination must be the exact opposite of the order of routers on the path from destination to source. Besides the source and destination of a path, the requirements on the connection must be defined as constraints. The necessary capacity and the maximal end-to-end delay are typical constraints of a connection. Formally, an immediate reservation request is defined as follows:

Immediate Reservation Request without Duration: These reservation requests are defined as $IM_1 = (C)$, where *C* denotes a set of requested constraints towards the managed resources.

An example for an immediate reservation request in the network domain is a tuple $C=(s,d,b,c)$, where *s* and *d* denote source and destination in a graph representing the network, *b* a bidirectional flag – which is either true or false – and *c* denotes the capacity needed.

Immediate Reservation Request with Ending Time: This reservation requests are defined as $IM_2 = (t_{end}, C)$, where t_{end} specifies the ending time and *C* denotes a set of requested constraints towards the resource.

A third alternative for IRs can be regarded, if the requested resources determine an ending time. An example is a request issuing a file transfer, which should start immediately and has a predetermined file size. In this case the reservation system might be able to map the file transfer request to a specific set of resource for a fixed time interval. In this case the specified type IM_1 and even IM_2 can be used as starting point.

In anticipation of the next sections, an IR request can be seen as a special form of advanced reservations, e.g. fixed advanced reservations discussed in section 3.3.2 can be used to specify IRs by specifying the current time as starting time. Thus, immediate reservations with known duration are not defined as a type of its own. A topic – differentiated in more detail in the upcoming section – is the split between time related parameters and resource related parameters, i.e. resource related parameters can be used to specify time related parameters depending as considered in section 3.3.4. Consequently, strategies to handle IRs are not sketched explicitly here.

Immediate reservations are present in various systems. The most prominent protocol to implement an IR in a packet switched network is IntServ, which uses RSVP to establish a service in an IP network. In converged technologies (MPLS, GMPLS) IRs can be established in a distributed approach using path selection with RSVP-TE at the ingress node (endpoint to which the request was issued). Furthermore, current activities in the IETF (PCE Working Group) allow for a centralized entity that handled the path selection process. The mentioned approaches allow for IRs without ending time, i.e. the path or circuits established at the starting time are terminated by the request issuer.

Noteworthy, the type IR is called “Grid-fast” circuit setup in the context of [11].

3.3.2 Advance Reservations

In addition to a reservation starting immediately – as described in the previous section – the starting time for the resource usage can be in the future. This type of reservation is called *advance reservation* (AR) in the following. The terms *in advance reservation* and *advanced reservation* can also be found in the literature. It is envisioned that ARs are required by various applications in the Phosphorus project and other Grid enabled infrastructures (cf. chapter 2).

In [12] and [13] ARs are specified and used to reserve resources. Following Zheng and Mouftah [12] ARs are split in three types:

- STSD: specified starting time and specified duration,
- STUD: specified starting time and unspecified duration, and
- UTSD: unspecified starting time and specified duration.

A 4th type is introduced by [14]. Following the described specification, the type UTUD (unspecified starting time and unspecified duration) is defined in conjunction with additional time constraints like earliest starting time and longest time. Although a similar notion is presented in [11], a different notion is used in the following. At first ARs are split in two types that consider the relationship of time and resource related constrains as follows:

- **ARs with independent time and resource constraints**: While validating the AR request and possible alternatives, choosing specific time related parameters does not influence the resource related parameters and vice versa. This type of reservations can be related to the rigid jobs introduced beforehand.
- **ARs with dependent time and resource constraints**: While validating the AR request and possible alternatives, the time and resource related parameters influence each other, i.e. choosing specific time parameters may influence resource parameters. This type of reservations related to the mouldable and malleable jobs introduced beforehand.

An example for independent time and resource constraints is a request that requires a fixed capacity in a specific time range. Although a different time parameter can result in selecting different paths in the network, the bandwidth parameter or requirement is fixed. A file transfer request is an example for the case of dependent time and resource constraints. The resource constraints can be specified by an amount of data (file size) that has to be transmitted. Here, choosing a time interval influences the capacity to allow for a completion of the file transfer.

Regarding the classification from Zheng and Mouftah [12], the following types with independent time and resource parameters are regarded in detail in section 3.3.3.

- *Fixed advance reservations* (FAR): A fixed AR has a specific time point as starting time and explicit duration value or unspecified duration. This reservation type relates to the STSD type.

- *Deferrable advance reservations* (DAR): A deferrable AR has a time range restricting the starting times and explicit duration value or unspecified duration. This reservation type relates to the UTSD type.

In addition *malleable advance reservations* (MAR) are introduced in section 3.3.4. The latter consider time and resource parameters interdependently, i.e. ARs with joined time and resource constraints. This type of reservation is also denoted as "Advance Cumulative Reservation" in [15]. To the best knowledge of the authors, Burchard [8] introduces the term malleable reservation and included this type in a simulative analysis. Although the term malleable is used slightly different in the context of scheduling – as mentioned in section 3.1 – this terminology is used in this document.

As mentioned in [16], it is also possible to introduce advance reservations with indefinite duration, meaning that the point in time when the reservations ends is not specified in the request. This corresponds to the STUD type mentioned above.

3.3.3 Fixed and Deferrable Advance Reservation

The nature of resource reservation in Grid computing environments differs from those dealing with spontaneous demand. Such reservations are usually made in advance and the duration or the amount of data to be transmitted is known beforehand. In contrast to immediate reservations, the points in time at which the reservation begins and ends are specified within advance reservations (ARs). The general service model of advance reservations has previously been described in other studies, e.g. in [17]. A basic form of an AR request is defined as follows and the life cycle is sketched in Figure 3-2: The request arrives at $t_{arrival}$, is admitted and starts at t_{start}. Furthermore, the usage phase (duration) is limited by t_{end}.

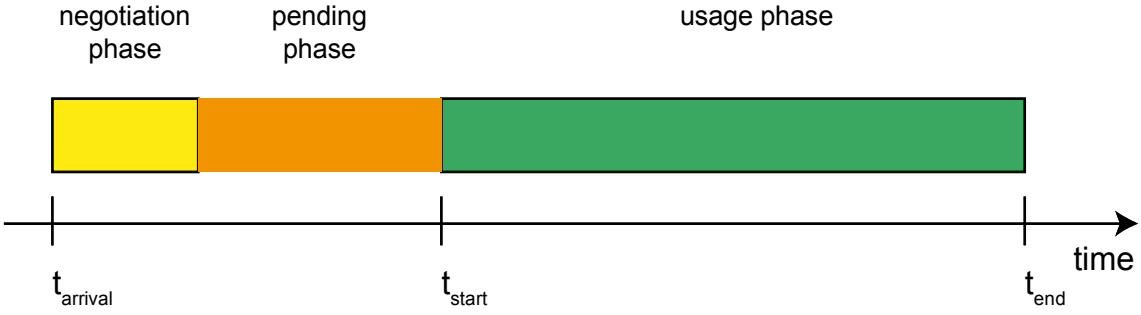

Figure 3-2: Arrival, start and end of a fixed advance reservation (FAR)

Fixed Advance Reservation Request: A fixed advance reservation request is defined as $FAR = (t_{start}, t_{end}, C)$ where $t_{start} < t_{end}$. The reservation starts at t_{start} and ends at t_{end}. The variable C represents additional resource constraints.

A particular admission model is to guarantee or reject an advance reservation request. In this case, whenever such a reservation request is received, the information about the allocation of resource to other reservations is accessed during admission control to determine whether a request is feasible or not. In case of advance reservations, if no restrictions are made, it would be possible to request resources boundlessly in advance. This

means that the reservation system has to manage resource allocation information for an open-ended period; but also similar difficulties as observed for immediate reservations can occur. Besides the high complexity of resource allocation management, serving and changing the network topology would become difficult as well. An implementable way to solve this problem is to limit the maximal interval of any given reservation. This interval is denoted as *book-ahead interval* (cf. section 5.2). Even if the usage of an unlimited book-ahead interval is also conceivable, its limitation restricts the memory consumption and avoids the problems mentioned above.

As mentioned in section 3.3.1, immediate reservations can be specified as advance reservations. Generally, it is possible to handle immediate reservations with known duration with an immediate- or advance-reservation scheme [18]. For the latter approach, which is followed here, the starting time of the advance reservation request is simply set to the point in time at which the request is transmitted. Hence, it is possible that the starting time of a request is before the point in time the request is received and therewith also outside the book-ahead interval. In this case the requested starting time can be set to the point in time at which the request is received by the reservation system ($t_{arrival} = t_{start}$).

The separation of the points in time at which a reservation request is received and at which it is supposed to begin results in a new degree of freedom for the reservation system. While immediate reservations must be replied directly, it is not necessary to reply an advance reservation request directly. However, it is assumed that the reservation system operates like an online system anyway; meaning that the reservation time is as short as possible. This is very important for the co-allocation of resources, as following reservations of non-network resources may depend on the admission control decision made by the network resource reservation system.

Beside the advanced reservations introduced as fixed advance reservations an additional type is presented. The main idea of the so-called *deferrable advance reservations* (DAR) is to introduce a degree of freedom in the time domain, i.e. time related parameters define a range of possible values to establish the reservation. This model is also described in [14] in the context of provisioning light-paths in a single or multiple domain. The life cycle of a deferrable advance reservation is given in Figure 3-3 and defined as follows.

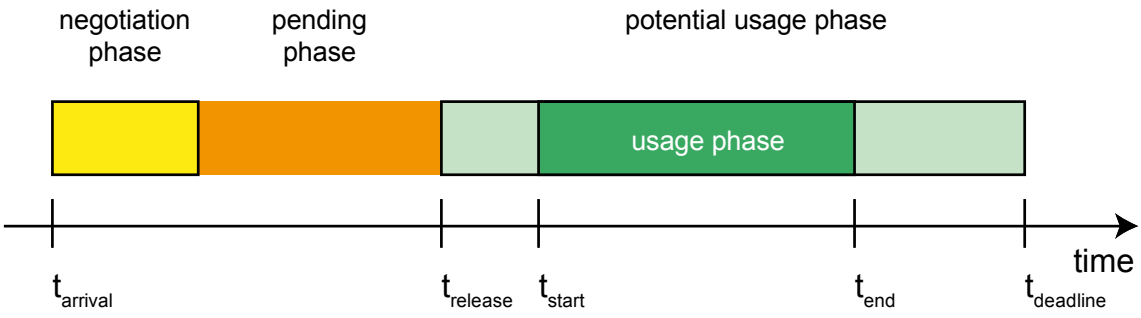

Figure 3-3: Timeline of a Deferrable Advance Reservation (DAR)

Deferrable Advance Reservation (DAR) Request: A deferrable advance reservation request is defined as $DAR = (t_{release}, t_{deadline}, d, C)$ where $t_{release} + d < t_{deadline}$. The reservation can start at $t_{release}$ and must end before $t_{deadline}$ the length of the usage phase is specified by duration $d > 0$. The variable C represents additional resource parameters.

Compared to a fixed advance reservation, the parameters t_{start} and t_{end} ($d = t_{end} - t_{start}$) can be determined by the reservation system. Depending on the flexibility of the client issuing the request and the reservation system handling the resource management, the admission result and start time can be sent at different points in time. For example, the reservation system might benefit to delay the transmission of the specific start and end time until the potential usage phase. In this case the system is able to modify a schedule and taking in to account additional requests.

An enhancement sketched in [14] is to allow for a duration interval, i.e. $d_1 < d < d_2$. Beside the flexibility of choosing the duration within the release time and deadline, an additional freedom is to choose the length of the usage phase. Again, various strategies can be introduced to handle this parameter range, e.g. choose a maximum value in the first admission control phase and reduce the usage phase down to the minimum value, if this allows for the acceptance of additional requests.

3.3.4 Malleable Advance Reservation

As motivated beforehand, the exact transfer rate for a connection is unimportant in situations where a fixed amount of data has to be transmitted. Only general capabilities of the sender and the receiver such as the maximal transfer rate and timing constraints for the transmission like a fixed deadline or an earliest stating time (release time) have to be regarded. An example in the area of Grid computing is given by the transmission of required input data to the involved nodes of a cluster, before the computation begins (pre-staging). For resource allocation purposes, it seems wise to implement the search for a suitable transfer rate and interval as a service of the reservation system, as shown in [8].

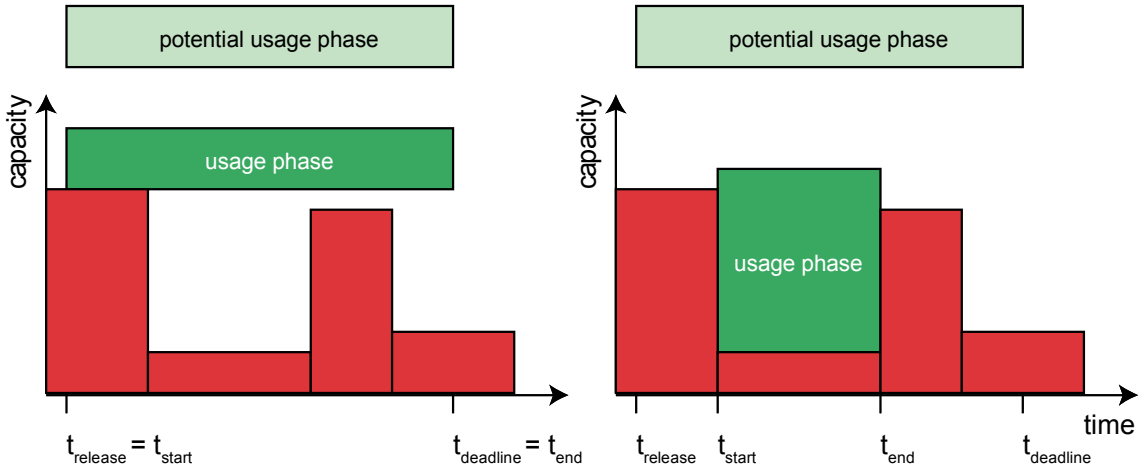

Figure 3-4: Examples of two a Malleable Advance Reservation (MAR) instances

By joining the time and resource constraints, the reservation system can find the most efficient solution for the requested transmission. This kind of reservation is denoted as *malleable advance reservation*. A motivating

example regarding efficiency is to fill gaps between allocated resources, which are caused by fixed advance reservations. These gaps can be filled with this type of service successfully (cf. Figure 3-4). On the left hand side, the system chooses an instance of the MAR request that reduces the used capacity and increases the duration. On the right hand side, the scheduler decided to increase the capacity for the request, which results in a reduced usage phase. The additional phases of an advance reservation are omitted in Figure 3-4. A malleable advance reservation request is defined as follows.

Malleable Advance Reservation (MAR) Request: A malleable reservation request is defined as $MAR = (t_{release}, t_{deadline}, C)$ where $t_{release} < t_{deadline}$. Furthermore, the variable C represents additional resource constraints that can be used to determine possible durations of the reservation.

Typical resource constraints for a MAR are a lower and an upper boundary for the transfer rate and a data amount to be transferred. Again, the admission decision and the transfer information can be sent to the requestor at different points in time. The transfer information contains at least a starting and ending time.

Generally, it is possible to adjust the transfer rate of a malleable advance reservation during the usage phase. This includes the possibility to interrupt a transmission for a while in order to find a new scheduling for the reservation request. It is up to the scheduling capabilities of the system in charge and the user capabilities, which strategies can be used. In section 6.3 a set these options is analysed in detail.

The admission decision for malleable advance reservations can become NP-complete. This can be proven, as shown in [15] by transforming the known NP-complete problem satisfiability (SAT) to a special case, the so-called *advance cumulative reservations problem*, a class of problems that also contains the problem for searching the first feasible interval for variable malleable advance reservations.

3.4 Conclusion

This section introduces three types of reservations that are regarded in the scope of the Phosphorus project. On the one hand, Fixed Advance Reservations (FAR) and Deferrable Advance Reservations (DAR) with independent time and resource constraints are defined and related to existent terminology in the literature. On the other hand, Malleable Advance Reservations (MAR) are defined. This type of reservation allows for an efficient resource scheduling, as time and resource constraints are dependent and allow for a flexible resource allocation.

4 Architectural Approaches towards Advance Reservation Systems

4.1 Compute Job Scheduling

Advance reservation of computational Grid resources plays a key role in delivering Quality of Service (QoS) to Grid users. The resource allocation involves resource selection, i.e. discovering and matching resources to each activity, finding time intervals on each resource during which an activity can execute (allocation), and finally making a combination of all available time intervals on all resources.

The advance reservation of computational Grid resources isn't a straightforward task, because of various reasons:

- The dynamic nature of Grids.
- There are resources of different types and capabilities, e.g., varying CPU speeds.
- Resources belong to different trust domains.
- Finding feasible workflow schedules and reserving appropriate resources while satisfying application's set of constraints (multiple constraints) is complex.
- There is a need for an agreement protocol between users and resources.
- There is a need for data structures for managing reservations.
- The advantages and disadvantages of centralized versus distributed advance reservation schemes should be evaluated.

In general the advance reservation of computational resources has two stages. In the first stage the scheduling of the reservation takes place, based on the user requirements. For the first stage a number of frameworks and architectures have been proposed. In the second stage the actual reservation is performed. The actual reservation of the computational resources is performed quantitatively, either by reserving a number of CPUs in a resource or by reserving a percentage of a CPU's capacity.

4.1.1 Advance Reservations Architectures for Computational Resources

In [19] the authors propose a complete framework for providing QoS in Grid Networks. GARA is the oldest framework for supporting QoS in Grids. This framework provides guarantees to an application requesting specific end-to-end QoS characteristics. For this reason it supports end-to-end discovery, reservation, allocation, and management of a heterogeneous group of computers, networks, storage systems, and other resources under independent local control. GARA splits the task of advance reservation into two phases: reservation and allocation. In the reservation phase, a reservation is created, which provides some confidence that a subsequent allocation request will succeed; however, no actual reservation is performed, instead, a reservation handle is returned, that can be used to monitor and control the status of the reservation. The introduction of a distinct reservation phase has two important implications. Firstly, by splitting reservation from allocation, it enables us to perform advance reservation of resources, which can be critical to application success if a required resource is in high demand. Secondly, if reservation is cheaper than allocation (as is often the case for large parallel computers, for example), we can implement lighter-weight resource reservation strategies than if objects must be created in order to guarantee access to a resource.

In [20] an advance reservation is obtained for an application from the meta-scheduler (resource broker), on behalf of resource provider, through a negotiation process. The authors propose a 3-layered (allocation, co-allocation, and coordination, cf. Figure 4-1) cooperative negotiation protocol that is used to efficiently reach an acceptable agreement. The first layer deals with allocation of a single Grid node. The second layer deals with co-allocation of multiple Grid nodes. Contentions, if any, are eliminated at the third layer. At each layer, a set of possible options are proposed based on different QoS parameters. Furthermore, in order to deal with the dynamic nature of the Grid, they introduce a priority provisioning mechanism, in which the reservation system guarantees that a certain capability will be available in the future. The decision of the actual node allocation however is done or exposed later on, just before resource acquisition. The resources, no matter where they reside or who owns them, are automatically allocated on demand in order to maximize the global utility. The negotiation starts either on a client's request or when the state of the Grid is changed (some resources leave or join the Grid). A co-allocator accepts requests from the clients and generates alternative co-allocation offers, which can be used by the client to execute its application. A co-allocation request may consist of a set of allocation requests for each Grid activity along with a set of constraints. The co-allocator negotiates with clients as a resource trader and with allocators as a cooperative negotiation mediator. Cooperative negotiation enables the system to generate offers closer to clients' requirements so that negotiation interactions with client are minimized, resource utilization and other QoS constraints are optimized, and inter/intra-application contentions are eliminated. Resource contention is introduced when the same slot is simultaneously offered for multiple Grid activities. Co-allocators try to agree over sharing of scarce resources without losing their clients. This introduces the need for a cooperative negotiation mechanism, i.e. internal negotiation between components of the reservation system. The negotiation process implies multiple interactions between clients

(e.g. schedulers) and co-allocators until they reach an agreement. Resources are offered to clients, who can select the best suitable offer or can decide to re-negotiate by changing some of the constraints. The goal is to generate co-allocation offers as optimal as possible so that interaction between the requester and provider is minimized and resource utilization is maximized.

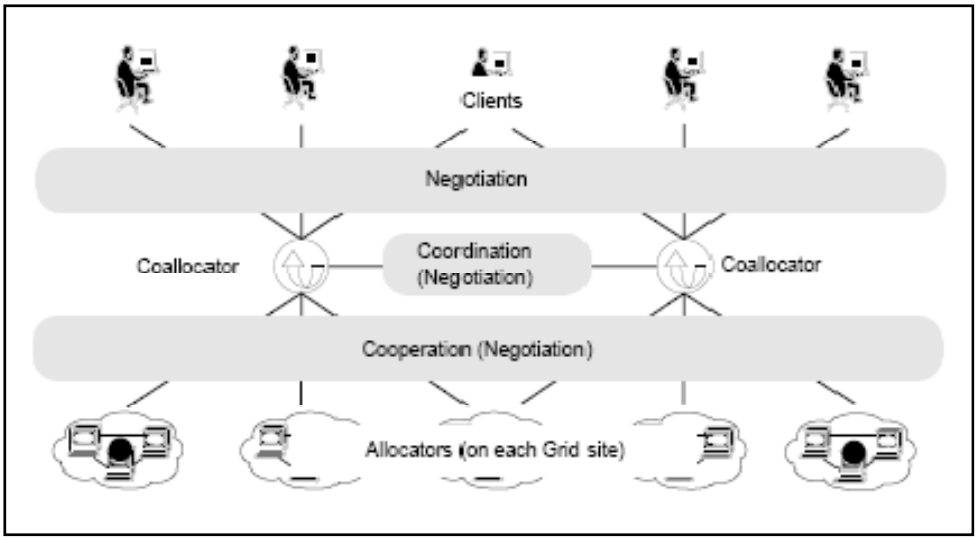

Figure 4-1: The advance reservation framework presented in [20]

In [21] the problem of finding feasible workflow schedules using advance reservations is examined. The authors analyze existing scheduling algorithms for heterogeneous environments and extended the well-known list scheduler HEFT in order to support advance reservations and co-allocation. The user gets a positive answer, if and only if a valid schedule was found. The resources will be reserved according to the schedule. Subsequently, the user will be informed about the guaranteed access.

Also in [23] the ICENI (Imperial College e-Science Networked Infrastructure) is presented, which uses a two tier system for reserving resources on the Grid. Individual resources may be reserved for periods of time potentially in the future. The Reservation service accepts a workflow from the Scheduling Service and then attempts to book appropriate reservations on the required resources to satisfy the workflow description.

The Maui [24] scheduler is an advanced job scheduler for cluster systems in which an advance reservation scheme enables future allocation of local resources. On a Grid site, Maui can be used in combination with other Local Resource Managers (LRM), such as PBS, SGE, LSF. Maui does not provide negotiation mechanism at resource and Grid level, and requires the Grid job submission service to be modified in order to be reservation aware.

4.1.2 Negotiation Protocols and Utilization Profiles

In order for the various entities of a Grid Network to coordinate the advance reservation of resources an agreement protocol must be used. Apart from the previously presented frameworks that (some of them) also include definitions of negotiation and reservation protocols, the Grid Resource Allocation Agreement Protocol-Working Group (GRAAP-WG) has specified the WS-Agreement [49]. The WS-Agreement defines a protocol based on Web-Services for establishing agreement between two parties, defines a language to specify the nature of the agreement, and provides agreement templates to facilitate discovery of compatible agreement parties. The WS-Agreement specifies three schemes:

- a schema for specifying an agreement,
- a schema for specifying an agreement template, and
- a schema for creation, expiration, and monitoring of agreement states.

The WS-Agreement itself is a framework; the details of terms to be agreed are domain-specific and is out of the scope of the WS-Agreement specifications, e.g. Job-submission using JSDL is a candidate to be used within the framework.

The information about the reservation of the resources as a function of time has to be kept by the reservation system in order to perform future scheduling decisions and actually allocate the resources to the corresponding users (or tasks). An efficient data structure for managing the advance reservations plays an important role in order to minimize the time complexity for searching available resources, adding new reservations, and deleting existing ones. In [22] the authors describe a modified version of the Linked List and Segment Tree data structures, which are capable of dealing with advance reservations in computational Grids. For Segment Tree, this entailed developing a new algorithm for finding a free interval closest to the requested reservation. Next, they describe how these operations can be performed on Calendar Queue [22], a data structure commonly employed for discrete event simulations. They also propose a Grid advanced reservation Queue (GarQ), which is a new data structure based on Calendar Queue and Segment Tree that improves some weaknesses of other data structures.

A detailed analysis of the resource management of the reservation system and in particular of the network resources is presented in chapter 5. For the context of this section we define the cluster availability profile, which is a general structure for storing the utilization of the computational resources. The cluster availability profile is used in deliverable D5.2 [48] for the implementation of various scheduling algorithms. The cluster availability profile is a general structure that can be used in a timeslot based reservation system with static or dynamic timeslot granularity (refer to section 5.2).

Specifically, we assume that a computational resource P consists of W_P number of CPUs that all have equal processor computation capacity C_P (million instructions per second). We will call the computation resource P a cluster. The utilization profile $U_P(t)$ of cluster P is defined as an integer stepwise function of time (cf. Figure 4-2), which records the number of processing elements that have been committed to tasks at time t relative to the

present time. The maximum value of this function is the number of CPUs W_P, while the stepwise character of U_P(t) can be explained by the discontinuities of height r_i (always integer number) between the starting and ending times of task i (that requests r_i CPUs). In the case that all tasks request a single CPU the steps are always unitary. We defined the *clusters availability profile*, which gives the number of CPUs that are free as a function of time, as $C_P(t) = W_P - U_P(t)$.

A meta-scheduler can use the information of the cluster availability profile to efficiently schedule the tasks. More specifically, when a task request i arrives (of computation workload L_i per processor and requesting r_i processors) the meta-scheduler can search the $C_P(t)$ of all resources for suitable placements of the task. A cluster P can serve the task i if in U_P(t) it can find $\frac{L_i}{C_P}$ (continuous) seconds that more than r processors are available. The task i can execute on cluster P by reserving in advance the corresponding resources.

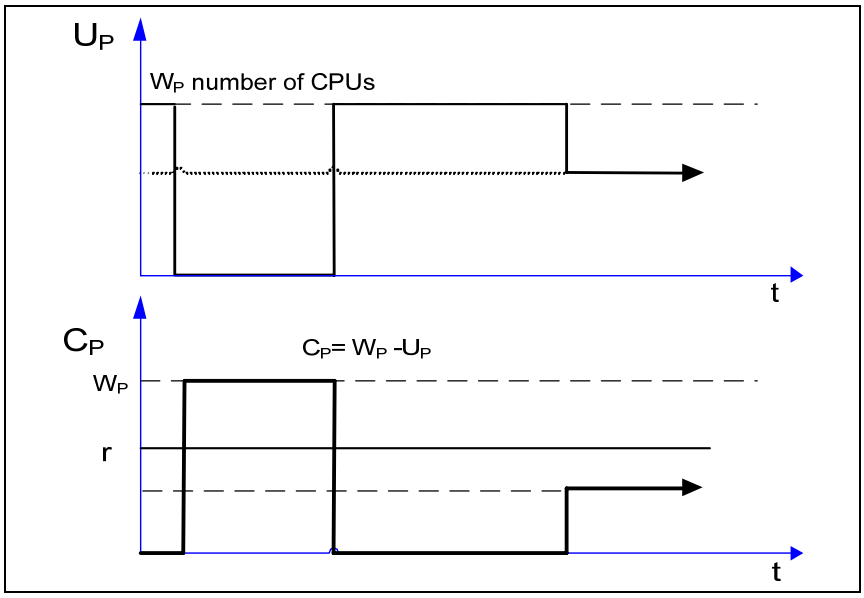

Figure 4-2: The cluster utilization profile $U_P(t)$ and the cluster availability profile $C_P(t)$ of a given cluster P with W_P processors each with equal processor computational capacity C_P. A task requests to be executed in r processors.

4.1.3 Distributed versus Centralized Advance Reservation of Computation Resources

One of the most important decisions in the design of advance reservation schemes is whether the decision and the planning of the advance reservation will be performed centralized or distributed. In both the scheduling and routing problems, there is the issue of whether we keep reservation utilization information at a central site (a

central scheduler) that is responsible for choosing the resource where each task will be executed (and possibly the path that will be used to send the data) or leave this job to be performed in a distributed way. In the distributed version each distributed scheduler keeps track of the resource utilization profiles (that can be outdated due to non zero propagation delays of the network and the update mechanism that is used) and uses this information to make the appropriate scheduling decisions.

In [46] the authors discuss the concepts of centralized, distributed and hierarchical task-scheduling in Grid computing and perform simulations to compare various alternatives of these schemes. The results show that due to the complexity of the problem the performance of the algorithms depend highly on the parameters, the computation machine configurations and the workload. The centralized vs. distributed problem has a number of other extensions apart from the task-scheduling and task-routing problems we consider in this deliverable. For example, the U.S. Department of transportation has examined the strengths and weaknesses of a distributed and a centralized telecommunications infrastructure and how these can enable Intelligent Transportation Systems (ITS) to function [47].

In order to give some answers to the above dilemmas we formulate the centralized vs. distributed problem and then perform simulation experiments in section 6.1.

4.2 Network Job Scheduling

In order to meet the requirements of coordinated computation in an interconnected world, two types of reservations were identified: Immediate and advance reservations. The latter are motivated by the fact that computational resources can already be booked in advance. The reservations specify certain QoS constraints that have to be mapped to services, which are provided by the network. In order to provide advance reservations, a concept of time is needed. While options for QoS are present in various network technologies, there is usually a lack of time dependent information and configurations within forwarding entities and control plane concepts. The challenge of advanced reservations is obvious: Without knowing the exact status of the network at future points in time it is difficult to decide whether a connection with a certain capacity can be accepted.

4.2.1 Network Advance Reservations Architectures

4.2.1.1 *Centralized Approach*

Network services to connect sites, nodes, or particular networks in a point-to-point, point-to-multipoint or multipoint-to-multipoint manner with a dedicated QoS are supported by different Network Resource Provisioning Systems (e.g. ARGON, DRAC, UCLP). Taking ARGON as an example, the NRPS interacts with protocol suites like MPLS and GMPLS. Both protocol suites support concepts to allow for the allocation of network resources by means of the routing protocol OSPF with traffic engineering extension (OSPF-TE) and the signalling protocol RSVP-TE. OSPF-TE provides a way of describing the topology with traffic engineering options, e.g. capacity and administrative weights. MPLS Traffic Engineering (MPLS-TE) nodes usually allow for

strict QoS guarantees, resource optimization, and fast failure recovery, but the current standard is limited to point-to-point connections. An extension to point-to-multipoint Traffic Engineering label switched paths is ongoing work at the IETF. While it is possible to provide multipoint-to-multipoint traffic engineering services using point-to-point traffic engineered paths, such an approach can be suboptimal, e.g. packet replication is performed at the ingress node. Depending on hardware capabilities, the deployment of multipoint-to-multipoint services is restricted to VPLS.

There are different approaches to realize the establishment of a path that was pre-computed by a centralized network resource reservation system for a reservation: It is possible to use explicit route objects which are supported by the MPLS and GMPLS control plane. If explicit routes are established in a network domain exclusively used by advance reservations, interactions with additional traffic are avoided. On the other hand, the deployment of a Path Computation Element (PCE) [14] is possible that consults the reservation system for the paths of reserved connections.

4.2.1.2 *Hierarchical Approach*

Hierarchical approaches allow the creation of end-to-end paths through multiple network domains. The domains can be under different administration and deploy various network technologies and standards. The main idea is to create an end-to-end network connection by stitching edge-to-edge paths in each domain. The process of path selection in these environments must also be technology aware, i.e. a stitching point between two domains must use a lowest common denominator to convey the network traffic.

In the scope of Phosphorus the Network Service Plane (NSP, [50]) is a hierarchical approach. The architecture is sketched in Figure 4-3. One the on hand, the NSP provides a central entity that can be used by the Grid middleware. On the other hand, different NRPSs (e.g. ARGON, DRAC, UCLP) can attach to the NSP be means of an NRPS Adapter that provides a common interface used by the NSP to check capabilities, query for topological information, and to perform reservations on behalf of the middleware. Further details about the NSP can be found in [50].

A similar hierarchical approach is used in GÉANT2's JRA3 activity. The fundamental entity is the so-called Inter-Domain Manager (IDM). The IDM is responsible processes incoming service request. Accepted requests are propagated to locally attached Domain Manager (DM) or to neighbouring IDMs. A DM is similar to a NRPS in the aforementioned scenario. Service requests can be propagated along a chain of domains until the service endpoint is reached. Multiple IDMs can act in a distributed fashion, i.e. the introduced hierarchy enables a distributed path computation at the IDM layer. Again, further details about the IDM can be found in [50].

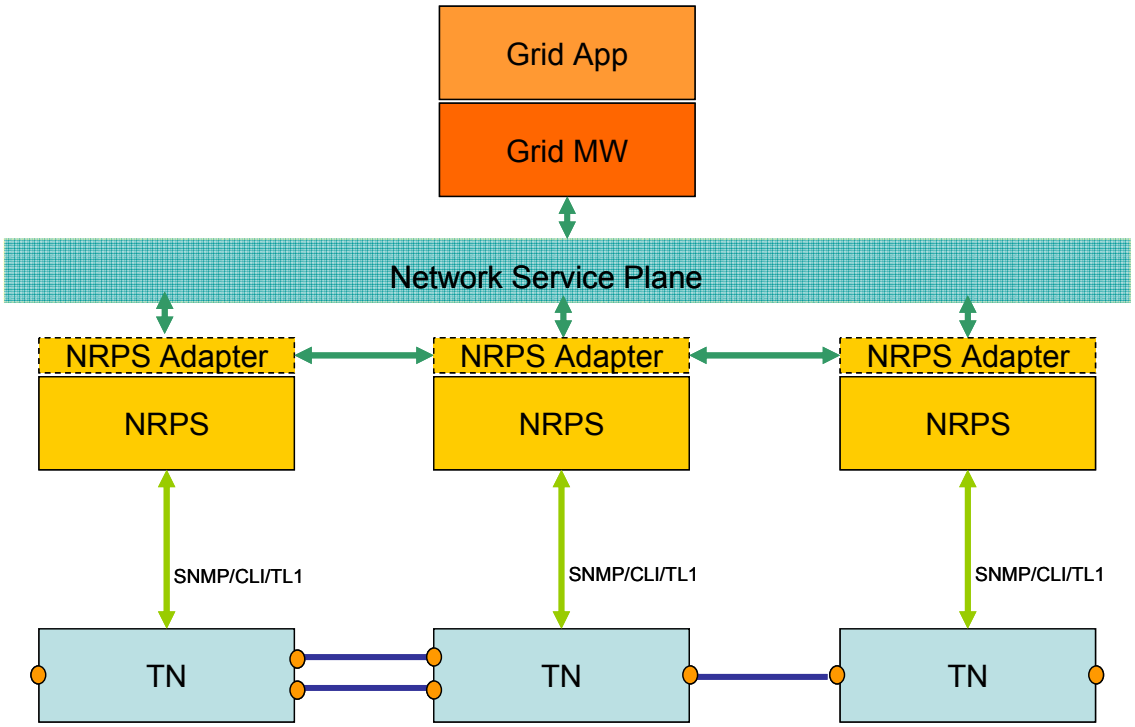

Figure 4-3: Client-Server architecture [50]

4.2.1.3 *Distributed Approach*

A distributed approach towards advance reservations in networks can be introduced by extending nodes with a concept of time. Current protocol suites like GMPLS are able to manage the current status of links, including the available bandwidth. This information is exchanged via the Traffic-Engineering extensions of the OSPF routing protocol. Furthermore, reservations can be implemented by setting up paths via RSVP-TE signalling. However, neither the routing protocol nor the signalling protocols described in current standards are able to deal with reservations or resources in future point in time. This gap in current standards is addressed in Phosphorus' WP2. Further details about the approach taken by WP2 can be found in [11].

4.2.1.4 *Optical Burst Switching Reservation Protocols*

As Grid applications evolve, the need for user controlled network infrastructure is apparent in order to support emerging dynamic and interactive services. In an attempt to address this problem, a draft for Grid Optical Bursts Switched (GOBS) was submitted to the Open Grid Forum [19]. In optical burst switched networks [26], [27] and [28] the data that has to be exchanged between the task originating (or data storage site) and the computational resource site is transmitted as a data bursts, and is switched though the network using a single label. This reduces the switching and processing requirements in the core network. When a burst is ready to be submitted, the source (ingress) node sends out a control packet to the destination (egress) node to setup the Label Switched Path (LSP) to be followed and reserve bandwidth for the burst (or bursts) that will be transmitted.

There are two types of burst reservation protocols [28], tell-and-go (one-way) and tell-and-wait (two-way). In tell-and-go protocols, the source transmits bursts without making any bandwidth reservations in advance. A control packet (called Burst Control Header, or BCH packet) is usually sent out-of-band and leads the burst by a small time offset (T_{offset}). The BCH packet contains information about the burst duration. At each intermediate link, the BCH packet reserves the appropriate resources for the burst that follows shortly after it. If the reservation at a node is not successful the burst is discarded, assuming there are no Fibre Delay Lines (FDLs) to store it. In tell-and-wait protocols, on the other hand, a source that has a burst to transmit first tries to reserve the appropriate resources from source to destination by sending a request message. Intermediate nodes receiving this message reserve the bandwidth on the desired outgoing links starting at the time the burst is expected to arrive at these nodes, and for duration equal to the duration of the burst. If this process is successful on all the links along the path, an ACK is sent back to the source, which then sends out the burst; otherwise, a NAK is returned to release the previously reserved bandwidth, and initiate the retransmission of the request message.

An example of a tell-and-wait protocol is the Efficient Reservation Virtual Circuit (ERVC) protocol proposed in [29], while recent research efforts include the Efficient Burst Reservation Protocol (EBRP) [35] and the WR-OBS [31]. The Ready-to-Go Virtual Circuit (RGVC) protocol [28] is an early attempt for a one-way reservation scheme in optical networks. In the recent years, one-way reservation schemes have received increasing attention. The Horizon protocol has been presented in [27] and Just Enough Time (JET) protocol has been proposed in [26]. The Just-In-Time (JIT) protocol proposed in [32] is a centralized protocol as it requires each burst transmission request to be sent to a central scheduler, which then informs the requesting node of the appropriate time to transmit the burst. Since centralized protocols are neither scalable nor robust a distributed version of JIT was presented in [33].

Of particular interest to us are the OBS protocols that employ a mechanism for “delayed” reservations such as Horizon [27] and Just Enough Time (JET) [26]. According to the classification presented in section 3.3.3, delayed reservations fall under the general category of “Fixed Advance Reservations”. Two way reservation schemes that employ “relaxed delayed” reservations such as ERVC and EBRP have a mechanism to negotiate the starting time of the reservation and thus (according to section 3.3.3) fall under the category of “Deferrable Advance Reservations”.

Various algorithms have been designed to harvest the advantages of advance reservations in OBS networks. The goal of these algorithms is to avoid the burst contentions in the core and balance the traffic load of the network.

The integration of contention resolution in the GMPLS framework is investigated in [39], where a load balancing technique is proposed, and is divided in two parts:

(i) traffic engineering is used at the edge of the network to reduce contention and
(ii) wavelength conversion and fibre delay lines are used for buffering in the core.

In [40] the authors introduce a dynamic Wavelength Routed OBS (WR-OBS) architecture where centralized control is employed to provide resource reservation efficiency, low delay, and QoS differentiation. In this algorithm the decisions are taken assuming advance reservations are employed in the underlying OBS

network. Taking a different approach, [41] presents a formulation of the path selection problem in OBS networks as an integer linear optimization problem by making some simplifying assumptions. This formulation tries to optimise the throughput of the whole network by scheduling the bursts accordingly, exploiting the concept of advance reservations to a great extend.

A recent approach in this field is a multi-cost routing and scheduling algorithm presented in [37]. The algorithm selects the paths to be followed by the bursts and the times when the bursts should start transmission from their source so as to arrive at their destination with minimum delay, addressing the burst routing and advance reservation planning problem jointly. The algorithm can function both with one- and two-way reservation schemes that have to employ advance reservations in order to enable the efficient scheduling of the bursts at the links.

4.2.2 Negotiation Protocols and Utilization Profiles

In order for the various entities of a Grid to coordinate the advance reservation of resources an agreement protocol has to be used. As in the case of the computational resources the WS-Agreement can be used between the scheduler and a resource for the negotiation of the reservation. For example, it is likely that a simple two-step protocol does not provide a solution for the negotiation phase. However, such a two-way protocol can be used in the last phase that is to actually grant the agreement. In this particular case the agreement has to do with the establishment of an end-to-end path, meaning the advance reservation of network resources that comprise the path. In the previous sections we have presented various ways to realize the establishment of a path according to the used architecture.

Similar to the utilization of the computational resources, the information about the reservation of the network resources as a function of time has to be kept by the reservation system. A detailed analysis of the resource management of the reservation system and in particular for the network resources is presented in chapter 5. For the context of this section we define the capacity availability profile, which is a general structure for storing the utilization of the network resources and can be used in a timeslot based reservation system with static or dynamic timeslot (section 5.2).

Assume that each connection reserves a constant amount of bandwidth for a given time duration. The *utilization profile* $U_l(t)$ of a link l is a stepwise function with discontinuities at the points at which reservations begin or end, and is updated dynamically with the admission of each new reservation. We define the *capacity availability profile* of link l as $C_l(t)=C_l-U_l(t)$, where C_l is the total capacity of the link l. In Figure 4-4 we show the capacity availability profile of a link.

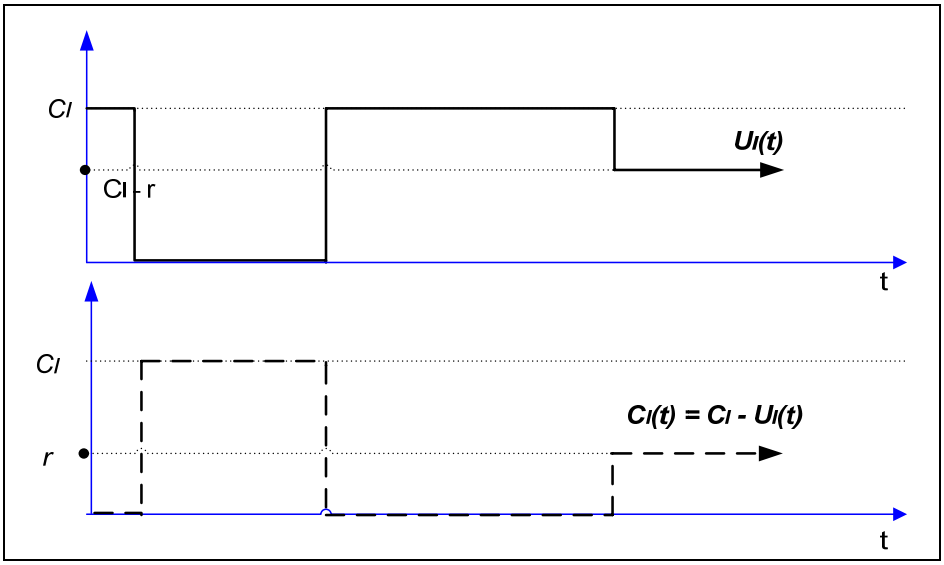

Figure 4-4: Step-wise capacity availability profile of a link.

4.2.3 Distributed versus Centralized Advance Reservation of Network Resources

As presented in section 4.2.1 there is the issue of whether we keep the network reservation utilization information at a central site (a central “scheduler” or a “bandwidth broker”) that is responsible for choosing the path that will be used to send the data or leave this job to be performed in a distributed way. In the distributed version each distributed scheduler keeps track of the link utilization profiles (that can be outdated due to non zero propagation delays and the update strategy) and use this information to take the routing decisions.

Centralized and distributed routing schemes have been compared, mainly, for WDM networks. In [44] the authors compare centralized and distributed connection management schemes under different traffic patterns in wavelength-convertible WDM networks. They show that the network blocking probability improvement of the centralized connection scheme over the distributed scheme is proportional to the reservation time and inversely proportional to the average connection inter-arrival time. For a high traffic load the distributed and centralized mechanisms yield the same blocking probability. In [45] the authors examine a novel distributed lightpath reservation mechanism based on multiple tokens in a WDM Ring topology, and compare it with centralized reservation mechanisms.

In section 6.2, we formulate the centralized vs. distributed problem in an Optical Burst Switched network and perform simulation experiments. An Optical Burst Switched network can be viewed as a special case of a network using timed and advance reservations.

4.3 Meta Scheduling

In the following two approaches to meta-scheduling are presented. The first approach is the MetaScheduling Services integrated in the UNICORE Grid middleware. For a more detailed view on this approach confer section 4.4 of [51]. Secondly, the Timed/Advance Resource Reservation (TARR) protocol is presented. The TARR uses timed and advance reservations in order to efficiently co-allocate network and computational resources, and is also extended in order to incorporate uncertainty in the process of co-allocation.

4.3.1 Advanced Reservations and the Hierarchical Scheduling model of the MetaScheduling Service

Resources in the Grid context may involve the aggregation of different computational and data resources. QoS in the context of aggregated resources means to ensure the availability of all required resources at the specified time for the certain period. Especially advanced applications benefit from the existence of different, heterogeneous resources available in Grids. For these advanced applications the required resources may also involve different sites. Resource management at these sites is done by local scheduling systems. To allow for a coordinated allocation of resources at different sites the local scheduling systems need to support an advance reservation mechanism. In the context of the Viola and Phosphorus project a central coordination entity (MetaScheduling Service) is used to negotiate a common timeframe for resource usage with multiple local schedulers.

4.3.1.1 *Co-Allocation and the MetaScheduling Service*

The MetaScheduling Service (MSS) [52] developed in the VIOLA project [53] allows the end-user to execute the individual components of its application using the most appropriate resources available. Examples of such applications are distributed multi-physics simulations where multiple resources are needed at the same time or complex workflows where the resources are needed with timely dependencies. Additionally, having distributed applications and data, there is also a need for dedicated QoS of the network connections between the resources to support efficient execution of the applications. Having reservation mechanisms allows to completely planning the execution of an application or workflow if the timely dependencies are given by the user.

The MetaScheduling Service developed in the VIOLA project is enhanced to cope with the requirements of the applications in the Phosphorus project. A detailed discussion on these requirements and the implications for the MetaScheduling Service can be found in [6].

4.3.1.2 *Architecture*

The MetaScheduling Service is integrated with the Grid middleware UNICORE. UNICORE is being developed and used in various projects and production environments. The MetaScheduling Service introduces capabilities

to make advance reservations or to provide synchronous access to distributed resources. Once a reservation has been made by the MSS, it is processed like any other UNICORE job. A job may consist of a number of sub-jobs one for each target system that is requested. Users can retrieve output, monitor, or cancel the job. In the current version, the plug-in is tailored to the needs of distributed simulations using the meta-computing enabled MPI-implementation MetaMPICH.

4.3.1.3 *Meta-Scheduling Service*

Once the MSS receives the agreement proposal with the necessary information on resources and QoS needed for an application from the UNICORE client, it starts to negotiate with the local resource managers (cf. Figure 4-5). The negotiation has four main phases:

(i) querying the local RMS for free slots to execute the application within a preview period

(ii) determining a common time slot

(iii) if such a time-slot exists: perform a reservation request of this slot on behalf of the user
otherwise: restart the query with a later start time of the preview period

(iv) check whether the reservation was made for the correct time slot on all systems,
if yes: we are done;
otherwise: restart the query with a later start time of the preview period

If no common time-slot within the local RMS's specific look-ahead times can be identified, an error is reported to the user. The successful negotiation and reservation is sent back as agreement to the UNICORE client, which then processes the job as usual. When the job starts at the negotiated common starting time the MSS collects the IP addresses of the participating machines (this information may not be available at an earlier time as the local scheduling system might assign the job to different nodes than planned at the time of submission) and communicates them to the network RMS which in turn is then able to manage the end-to-end connections with the requested QoS.

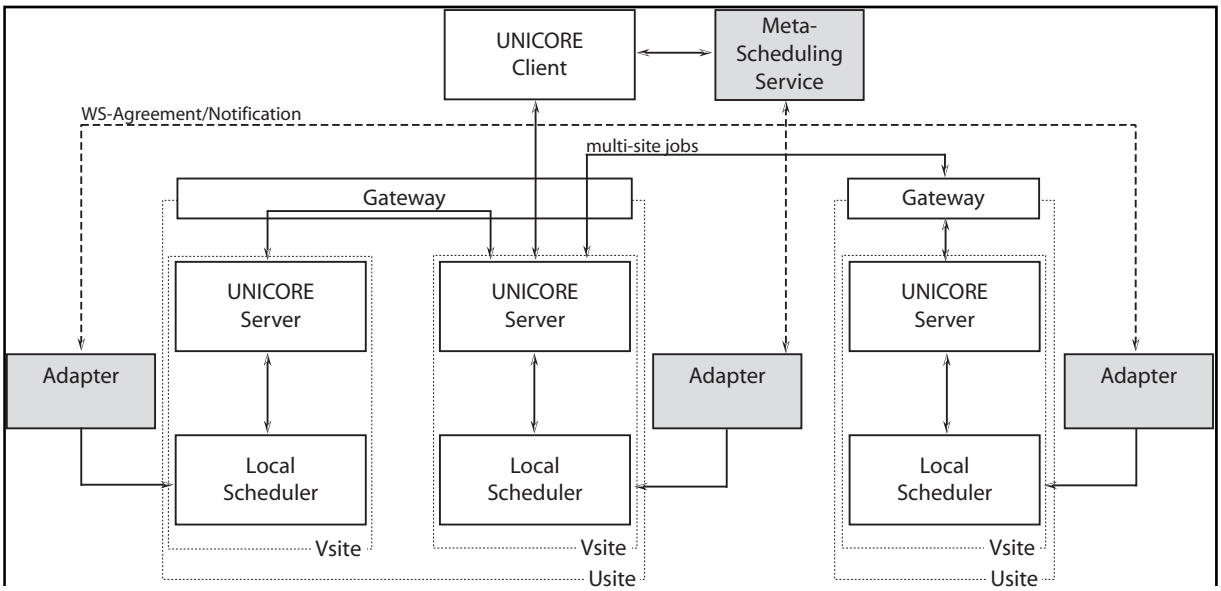

Figure 4-5 – Architecture of the VIOLA Meta-scheduling Environment

4.3.2 TARR - A Control Protocol for Joint Reservation of Communication and Computation Resources

The requirement of on demand and efficient use of resources implies that resources should be allocated to a task only for the time interval during which they are actually used, and should be available to other tasks for the remaining time. This is not accomplished by earlier resource reservation protocols, which usually reserve resources for considerably more time than the minimum required. To alleviate this problem, we introduce a new resource reservation protocol, called the TARR protocol [43], which uses the concepts of *delayed* (or *timed*) *and advance reservation* of resources. The protocol uses estimates of the communication delays and the task execution times in order to reserve (computational and network) resources only for the exact time periods during which they will actually be used by a task, and leave these resources free to be used by other tasks for the remaining of the time.

We propose a resource reservation scheme for Grids that uses the notions of advance and timed reservations. To better appreciate the advantages these two features offer, we first describe the usual approach where advance/timed reservations are not used. We refer to such a scheme as a *standard resource reservation scheme*, and we outline the reasons such schemes are inefficient.

4.3.2.1 *Standard Resource Reservation Schemes*

In standard resource reservation schemes, when the scheduler assigns a task to a resource, it records a reservation for that resource in its database. The resource is considered allocated to the task for an unspecified amount of time, starting at the time the task-to-resource assignment is made. As far as the scheduler is

concerned, the resource remains booked for the task until the scheduler is informed about the task completion. This situation is shown in Figure 4-6, where the scheduler waits for a *release message* to arrive from the resource before it can allocate the resource to a new task. To allocate the resource to a new task, the scheduler must send an *allocation message* to the application telling the user-side where the data and code required for the execution of the task should be transferred. The data and code are then transferred to the resource, which takes some additional communication time. During the transmission of the allocation message and during the transmission of the data, the resource remains (unnecessarily) inactive. When all the data have arrived at the resource, the task begins execution. When the task is completed, the resource remains again (unnecessarily) inactive until the release message arrives at the scheduler, which can then allocate it to another task.

We denote by $2t_p$ the average time that elapses between the time a resource sends the release message to the scheduler to inform it about the completion of a task and the time all the data required to execute the next task arrive at that resource. We also denote by x the mean execution time of a task on that resource. It is then clear that when a standard resource reservation scheme is used, the efficiency with which a resource is used is at most

$$e = \frac{x}{2t_p + x}$$

Note that the efficiency factor e may be considerably smaller than 1, and it decreases as x decreases or as $2t_p$ increases ($2t_p$ is at least as large as the roundtrip propagation delay). In order for the Grid to be useful for a number of different applications, we would like to be able to use fine grain computation (where x is small) and also be able to use remote resources (where $2t_p$ is large), both of which correspond to small values for the efficiency factor e. Thus, standard resource reservation algorithms fundamentally restrict the efficiency with which Grid resources are used. In the following subsection we show how we can use timed/advance reservations to make the efficiency factor e close to 1, independently of the communication delays in the network and the granularity of the tasks.

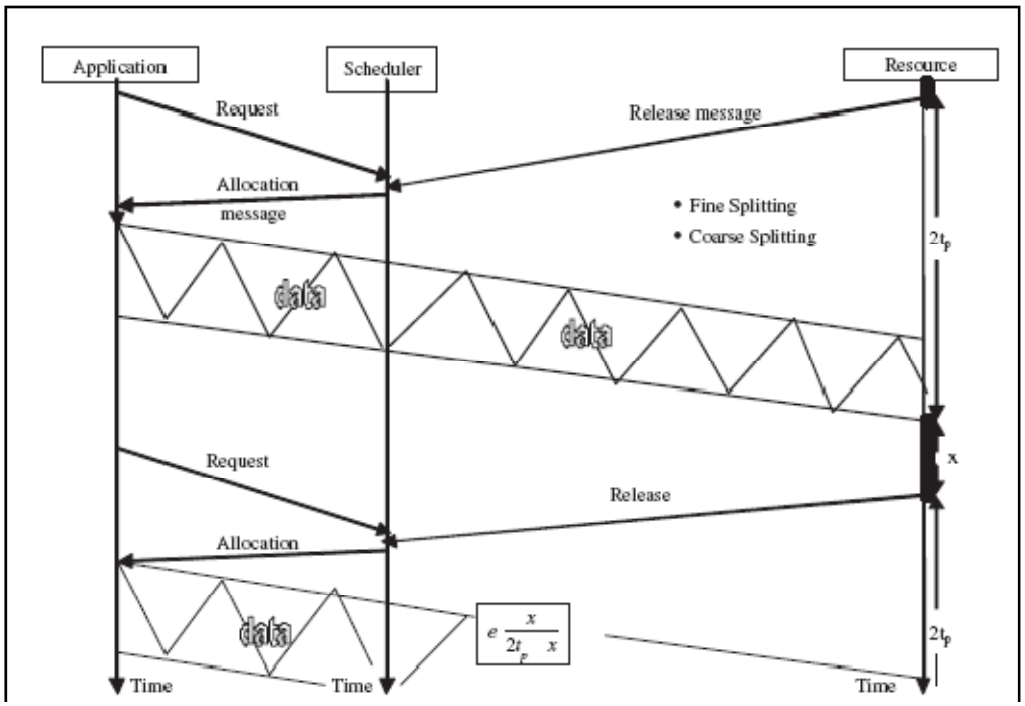

Figure 4-6: Possible scenario when a *standard resource reservation* scheme is used.

4.3.2.2 *The Timed/Advance Resource Reservation (TARR) Scheme*

In the Timed/Advance Resource Reservation scheme (abbreviated, TARR) that we propose, the scheduler maintains a *utilization profile* for each network and computational resource, which keeps track, as a function of time, of the future utilization of that resource. More specifically, the utilization profile of a computational resource records the (future) time intervals for which the resource has already been reserved for executing tasks. The resource may also be a communication link, in which case the utilization profile keeps track of the bandwidth that has been reserved on that link as a function of time, or it may be a memory resource or an instrument that is shared by users. When a resource is released, the scheduler is informed in order to keep the resource utilization information up to date. When a request for a task is sent to the scheduler, the scheduler allocates a resource to the task for a specific future time interval. In calculating the start time (t_{start}) of that time interval the scheduler takes into account the estimated communication delays, so as to make sure that all the data necessary to run the task will have arrived at the resource before time t_{start}. In calculating the end time (t_{end}) of the interval the scheduler takes into account the t_{start} and the estimated workload of the task. Therefore, the scheduler uses the resource utilization profile to allocate a future interval for each task scheduled on a given resource, so that when one task is completed, the data for the next task are already available at the resource for the new task to start execution immediately. This maximizes the efficiency of the resources and the efficiency factor *e* can get very close to 1.

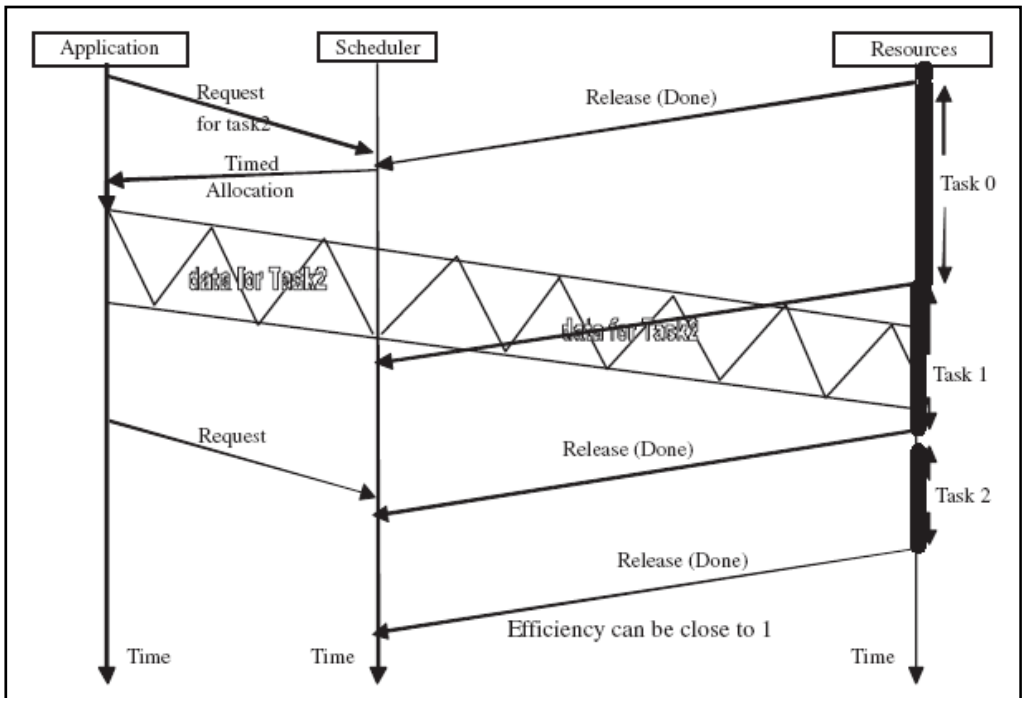

Figure 4-7: Resource reservation when advance/reservations are used. The resources are reserved only for the time during which they are actually used by a task.

An example of a timed/advance reservation made using the resource utilization profile is shown in Figure 4-7. The scheduler is responsible for assigning the tasks, each of which has a known (estimated) workload, to the appropriate resource using the utilization profiles of the resources. The information that has to be provided to the scheduler to make a decision is:

- the required CPU capacity (C),
- the required execution time (x) assuming capacity equal to CPU is allocated,
- the job completion deadline ($t_{deadline}$), and
- the communication delays δ_{ij} for the transfer of all the data involved in the execution of task T_i from the user to resource j.

These parameters are supplied to the scheduler by the user and possibly by a forecasting tool. The total estimated workload can be calculated as $L = C * x$. The scheduler takes into account the communication delay for transferring data from the user application to a resource, in order to find the earliest possible starting time of a task at that resource.

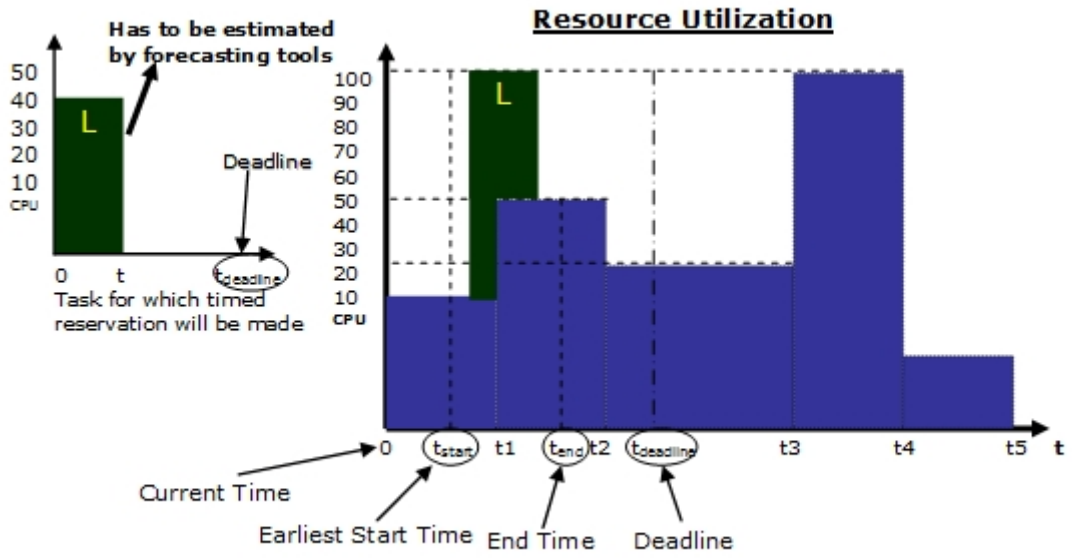

Figure 4-8: Illustrates an example of a timed/advance reservation using the resource utilization profile. The start time t_{start} of a reservation interval is chosen so as to ensure that all data necessary for the execution of the task have arrived to that resource by time t_{start}. The end time t_{end} of a reservation interval is chosen so as to ensure that the task is completed by time t_{end}. If we cannot allocate resources so that $t_{end} < t_{deadline}$, where $t_{deadline}$ is the deadline the job is rejected.

The next scheduling step is to reserve (a portion of) the resource for a specific duration for the execution of task. The reservation is performed by fitting the total workload in the resource utilization profile. This step may also involve a negotiation process, as shown in Figure 4-9. It is possible that the CPU capacity required for a task may not be available at the time it is requested, in which case a reservation for a time interval later than the time requested may be chosen. Also, the scheduler may negotiate/specify new parameters for the CPU capacity and corresponding time duration that will result in the same total workload. When the reservation is completed, the estimated end time of the task is calculated and the resource utilization profile is updated. In the resource utilization profiles shown in Figure 4-8 and Figure 4-9 the local resource manager is assumed to support time sharing, so that it can allocate fractions of the CPU capacity to the tasks. If this is not the case, and the resource has to be allocated a 100% of the time at a task, the utilization profiles and the start and end times for the reservation intervals can be easily modified.

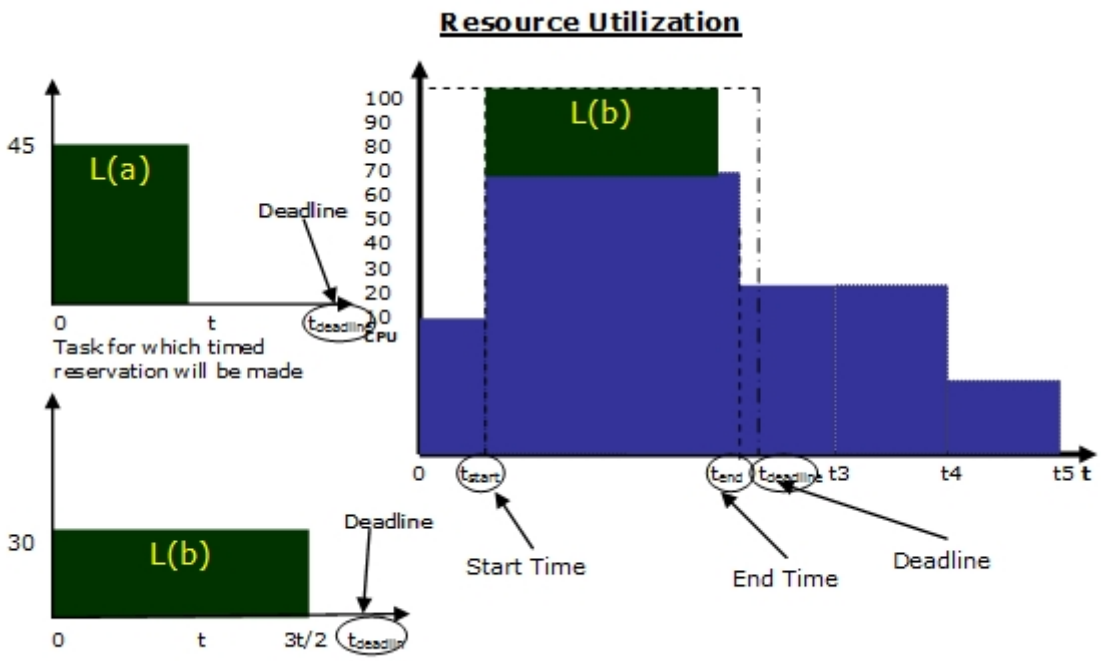

Figure 4-9: Illustrates a negotiation example. The applications requests 45 units of CPU capacity for duration equal to *t*, but gets 30 units of CPU capacity for duration equal to 3*t*/2 after the negotiation. (a) The workload as it was supplied by the application; (b) The workload as it was specified after the negotiation process.

4.3.2.3 *Timed / Advance Reservations under Uncertainty*

In the examples considered so far it was assumed that the task workload and the communication delays can be estimated with good accuracy by the forecasting tools and/or by the user. In practice, however, these estimates may not be accurate; the scheduler should be able to function correctly in the presence of such uncertainty. Figure 4-10 shows the utilization profile at a resource where a reservation for a given task has to be made. We assume that some rough estimates for the task workload and the communication delays involved are available, but there is a certain degree of uncertainty in these estimates. The degree of uncertainty can be expressed in terms of variance. Since the communication delays are not accurately known, it is not known when exactly the data required for the execution of the task will arrive at the resource. Also, since the workload is not accurately known, it is not known how much time it will take for the task to complete after it starts execution.

The rule to be followed by the scheduler in order to work correctly in the presence of such uncertainties is that the scheduler must underestimate the start time of the reservation (i.e., start the reservation earlier than its estimate) and overestimate the end time of the reservation (i.e., record the end time later than its estimate) by some small multiple of the standard deviation of these estimates. For example, if the mean transfer time for data is estimated to be δ seconds, the actual reservation on the resource can be made starting at time $\delta - 3\sigma_\delta$ after the present time. In Figure 4-10 the estimated start time is shown as t_{start}, while the time the reservation

actually starts is shown as $\check{t}_{start}$ (Underestimated Start Time). A similar rule is used when the workload is not known with accuracy. In that case the end time of the task has to be overestimated, in order to make sure that the task will have all the required resources for it to complete execution. The end time $\hat{t}_{end}$ (Overestimated End Time) used for the actual reservation is calculated by overestimating the workload ($\hat{L}$ = $L + 3\sigma wl$).

The rules mentioned above can be used separately or together depending on the accuracy the forecasting tools can provide. For example, when the the workload estimate is precise but the communication delays cannot be accurately estimated, we need only underestimate the start time of the reservation.

When the forecasting tools are unable to provide any estimates of the communication delays involved (or they provide no measure of how accurate the estimates are, that is, they provide no σ_{δ}), then the reservation should start immediately, as shown in Figure 4-11, as if the delay for the data to be transferred from the application to the resource were zero. In that case, the resource is reserved much earlier than required, and efficiency decreases. In Figure 4-11 the actual start time is shown with t_{start} and the task deadline is shown with $t_{deadline}$. If the workload is known, then when the task actually starts execution, a message is sent to the scheduler to allow the calculation of the end time, which is then recorded in the utilization profile.

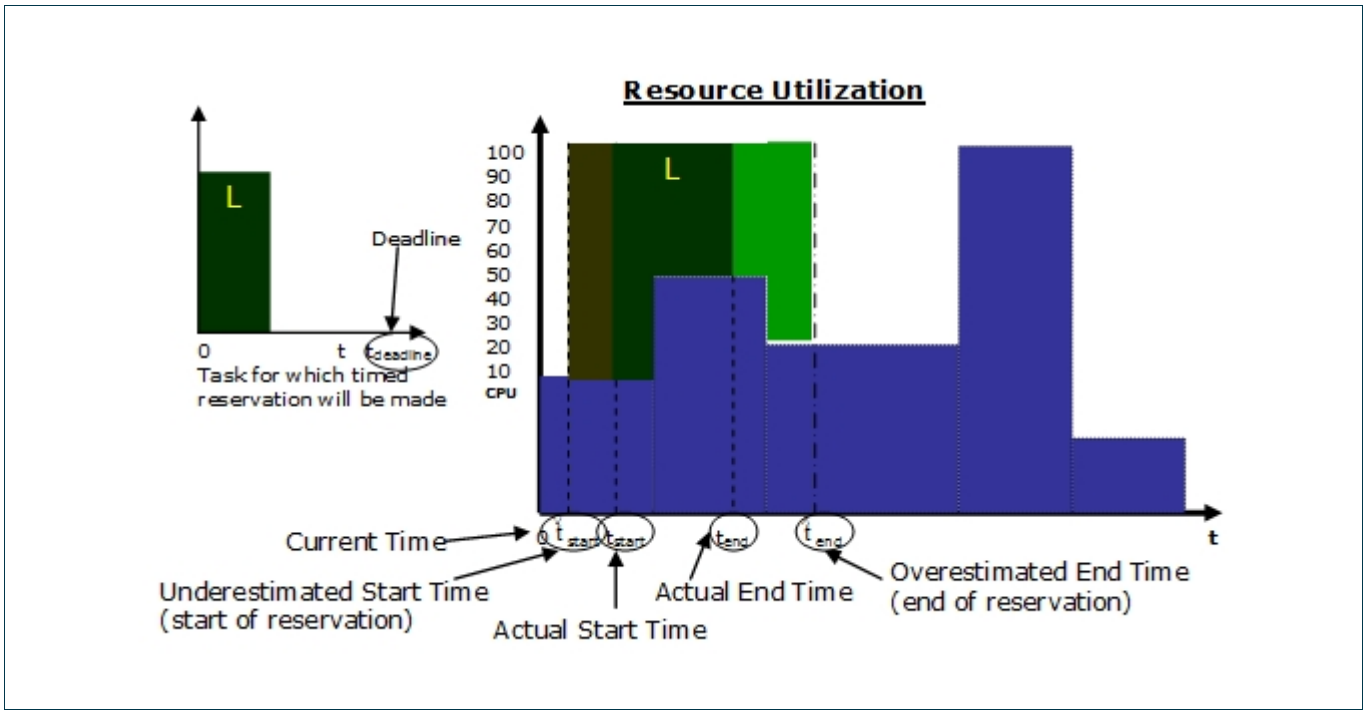

Figure 4-10: Illustrates the way timed/advance reservations can be made in the presence of uncertainty. The rule that should be followed is that, when in doubt, we should overestimate the workload and/or underestimate the start time in order to be safe. For example, when the variance σ^2_{wl} of the workload is known, we can assume that the workload is less than $L + 3\sigma_{wl}$ with high probability. Similarly, a lower bound that could be used for the start time is $t_{start} - 3\ \sigma_{\delta}$, with high probability.

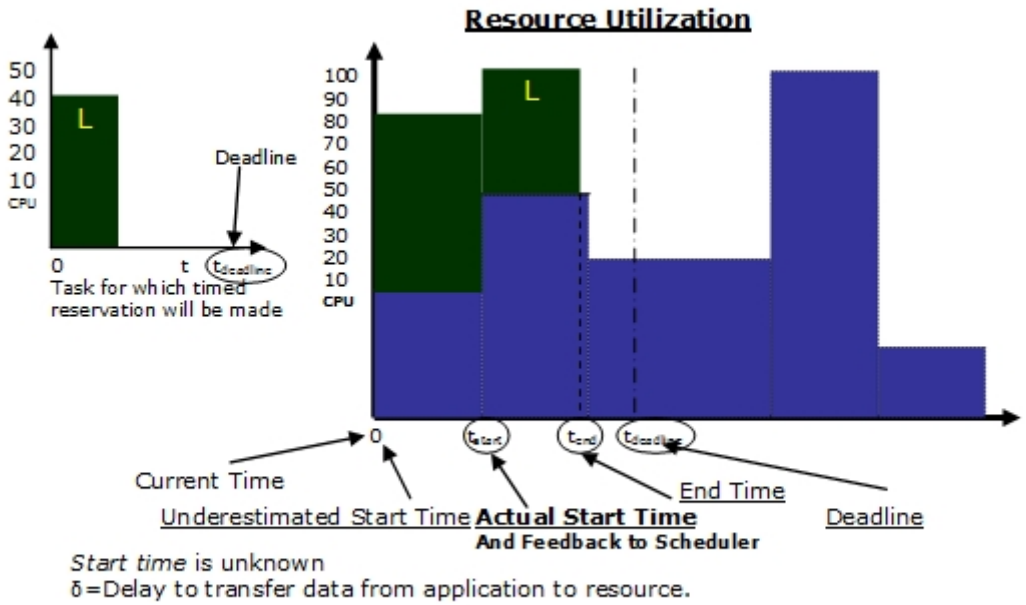

Figure 4-11: Advance reservations when the start time is unknown.

When no estimate for the total workload is available, the end time should be considered infinite. The resource is then released when the task has completed execution and a release message has arrived to the scheduler (Figure 4-12). The efficiency factor *e* is then the same with that of standard resource reservation protocols. Clearly, when the workload is not known, the task deadline cannot be guaranteed, and the user must be informed in advance that such no guarantee of the QoS can be provided.

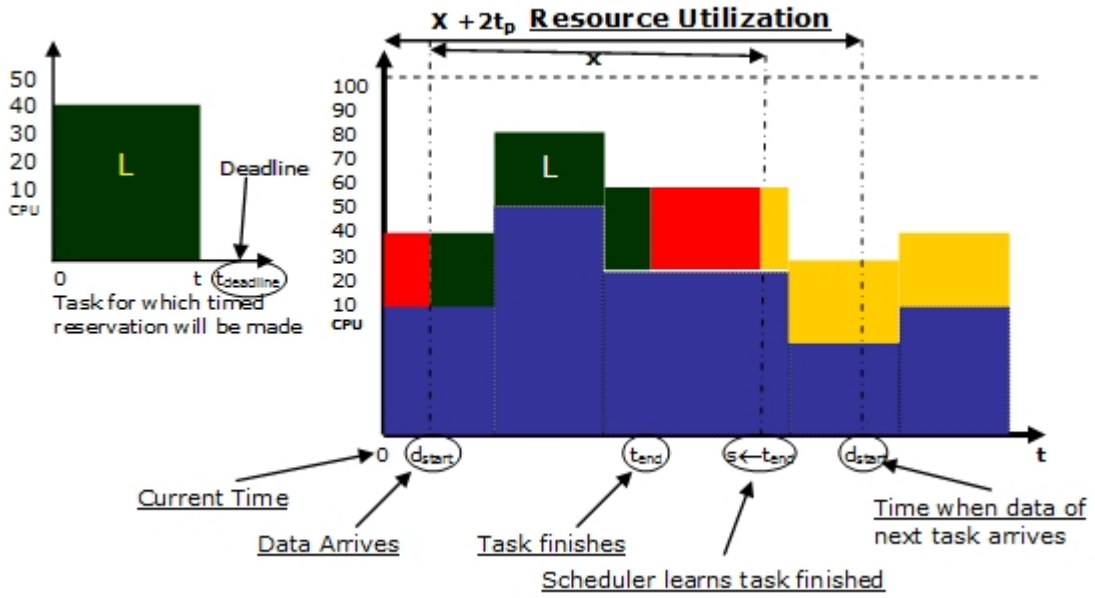

Figure 4-12: Advance reservation when no estimates for the workload and the communication delays are available.

5 Resource Management for Advance Reservations

Information about the reserved resources in the network is, besides the computed paths for accepted reservations, the most important status information that has to be kept by the reservation system. Obviously, the resource utilization information over time is needed for the admission decision of every request that overlaps the period for which the resources are reserved. The structure how this information is managed determines how much time is required for the pre- and the post-processing and how effective network resources are allocated to requests. Although this chapter focuses on temporal aspects in the scope of network resources, the approaches can be applied to other resources (e.g. computational resources) as well.

5.1 Definitions

Information about the reserved or scheduled resources has to be kept by the reservation system. Obviously, the resource allocation information of an accepted reservation is needed for the admission decision of every request that overlaps the period for which the resources are reserved. In order to provide a responsive reservation system, the period between receiving a request and replying to it is to be as short as possible. This period is primarily determined by the time that is required to prepare and to execute the resource planning. The first one, denoted as pre-processing time, depends mostly on the way the allocation information is managed. This part should be kept on a low level, independent of the number of accepted reservations. Following operations, denoted as post-processing, can be done after the transmission of the reply. In a straightforward manner, for example the information which is managed for the establishment and teardown of computed paths can be used to determine the available resources for a given period [10]. For instance, this information can be managed as an entity for every accepted reservation, which contains information about the resources which are allocated. Furthermore, the points in time at which the provisioning will be performed by the scheduler are managed as t_{start} and t_{end} in the respective entity.

Reservation Entity: A reservation entity e is defined as e = (t_{start}, t_{end}, R) where t_{start} and t_{end} denote the points in time the setup and teardown of the reservation begins and R specifies the resources to be used, e.g. a path in the network and the reserved capacity.

Every time the reservation system has to make an admission decision for a reservation or modification request, the entities which have a setup or teardown time that overlaps the new request are determined. These entities, denoted in respect of a request as *active entities*, have to be considered when the feasibility or the expansion of the accepted request is checked. Only reservations represented by an active entity can block resources for the new request. In the case of an advance reservation, the reservation system has to determine the minimal available resources before a schedule is computed. This is equivalent to determining the maximal reserved resources. Therefore, it is sufficient to consider the allocated resources of the active entities at their setup times. These are the only points in time the amount of reserved resources grows, so the maximal allocation of every resource must be reached at least one of these points in time.

5.2 Resource Management Approaches

There are various approaches to resource management for reserving resources in the scope of Grid computing and networking. This section introduces different approaches which are further analyzed in the subsequent sections. Without loss of generality, we focus on temporal aspects in the scope of allocating network resources.

5.2.1 Reservation-Based Approach

A basic reservation-based approach uses the set of already accepted reservations for admission control of incoming reservation requests. All accepted requests overlapping the requested time interval are identified [58], which allows determining if enough resources are available to fulfil the request. This concept has low memory consumption as it stores only accepted requests which are needed for connection establishment anyway. However, if one reservation request is handled after the other, up to $i-1$ accepted requests have to be considered in the worst case when the i^{th} request is handled. This means that the time to determine the available resources for n subsequent requests is $\sum_{i=1}^{n} i-1 \in \Theta\ (n^2)$. As a consequence, the approach is favourable if the number of requests is low. To cope with the complexity, a timeslot-based approach is introduced that maintains aggregated resource consumption information.

5.2.2 Timeslot-based Approach

The *timeslot-based* resource allocation management uses a global timeline divided into a set of *timeslots*. Each slot represents a period of time and holds information about the accumulated resource consumption for every link as depicted in Figure 5-1. The timeslot-based approach can be used to determine available resources independent of the number of accepted reservations. This can be achieved by evading the repeated access to reservation entities for the same period. Storing the summarized resource utilization for pre-defined periods reduces the processing time for new requests. Once a new request is processed, the reservation system has to

determine the *active timeslots*, a subset of all timeslots that cover the requested period. The maximal resource utilization of all active timeslots along with the resource capacity determines the resources available.

The timeslot-based management of allocated network resources is an established [54], [19], [15] and efficient way to manage utilization information. The approach is already used in different environments for advance reservations, e.g. denoted as *timeslot table* in GARA [13] – a component of the Globus Toolkit, which performs the resource management.

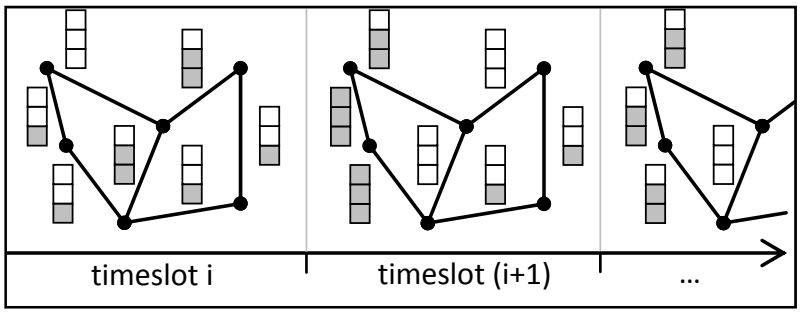

Figure 5-1: The timeslot-based resource allocation management manages the accumulated resource-utilization for every link for a given period.

5.2.3 Static and Dynamic Timeslots

A model with n_{ba} timeslots with a constant length c_s is denoted as static timeslots model. Here, the boundaries of a timeslot are indirectly defined by its index *i* and the slot length c_s. Using this model the number of timeslots which are managed by the reservation system is completely independent of the number of accepted reservations; it only depends on the length of the book-ahead interval $c_{ba} = n_{ba} * c_s$ and the length of a single timeslot c_s. While this approach is very easy to implement, it is inefficient if only a small number of reservations is managed by the reservation system.

Alternatively, timeslots of dynamic length can be used with a length of an integer multiple of c_s. For every new reservation, the existing timeslots are divided at the starting time $\hat{t}_{start}$ and the ending time $\hat{t}_{end}$, unless the timeline is not already divided at these points. Every time a new reservation is accepted, no more than two new timeslots have to be created. Even if the number of timeslots depends on the number of accepted reservations, it is limited by n_{ba}.

5.2.4 The Granularity of Resource Management

The mechanism of using time slots for a faster admission control is enmeshed with the *granularity* for the starting and ending times of reservation requests. This granularity is defined as the smallest Δt which is distinguished by the reservation system. While service takers generally specify starting-and ending-times with a limited precision which can be milliseconds, seconds, or even minutes(*user-specific granularity g*), a static timeslot based approach is also dependent on a granularity as it defines the size of the timeslots c_s.

With the reservation-or dynamic timeslot-based resource management approach the system can theoretically operate with *infinitesimal granularity*. In this case the starting and ending time of requests are not altered and in case of dynamic timeslots a boundary of a timeslot is set at any point in time a reservation begins or ends. However, this approach does not limit the maximum number of timeslots and therefore the processing time and memory consumption on the number of accepted reservations. A newly accepted reservation may result in up to two new timeslots. So, in the worst case, the reservation system has to manage (2n + 1) timeslots for n accepted reservations. This is shown in Figure 5-2.

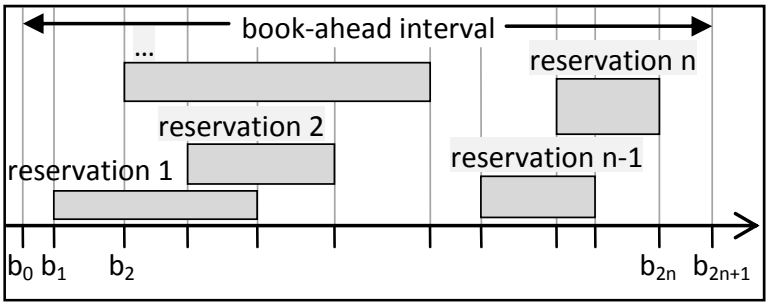

Figure 5-2: Up to (2n + 1) timeslots are needed to manage n reservations if an arbitrarily granulation is used for resource management.

This dependency can easily be avoided by introducing a *non-infinitesimal granularity* for the timeslots. Only a fixed set of usually equidistant points in time within the book-ahead interval is allowed as timeslot boundaries. Reservation requests need to be aligned correspondingly. Considering typical application areas, reservations are made in steps of 5 up to 15 minutes [10], so this also seems to be an adequate value for c_s.

5.2.5 Adjusting Reservations to Granularity

In general, the usage of a non-infinitesimal granularity makes it necessary to map arbitrary points in time specified in a request on a discrete set accepted by the reservation system. The approach described in [16] rounds off the starting and ending time of a reservation to the nearest slot boundaries. Starting and ending times within the interval $[b_i - \frac{1}{2}c_s; b_i + \frac{1}{2}c_s[$ are rounded to the slot boundary bi. A major advantage of this approach is that the average duration of the reservations will not grow compared to the original one, if a uniform distribution of the starting times and the durations, which are integer multiples of c_s, is assumed. This ensures that the average amount of resources reserved for a set of reservations does not increase by the mapping. As will be explained, this approach is inappropriate for reliable resource allocation, especially in the area of Grid computing. Changing the reserved time interval to one where the requested interval is not fully included requires a negotiation process. A Grid meta-scheduler [52] takes into account the reservation periods of multiple resources that are co-allocated with the network, e.g. it has to enforce real-time requirements of resources like sensors, and guarantee deadlines for data transfers. All starting and ending times of co-allocated reservations need to be aligned.

This reflection leads to another way of mapping arbitrary reservations to a certain granularity. Instead of rounding off requested periods, the reservation system maps reservation requests in such a way that the

requested period is entirely included in the assigned one. This can be done by rounding down the starting and rounding up the ending time to the next slot boundary. The main advantage of this approach is that service takers need not be notified and the co-allocation mechanism is not constricted. Furthermore, there exists no use case in which this approach is more inefficient than extending the required period by 2 c_s.

In the following, the original requested starting and ending times are denoted as t_{start} and t_{end}. The starting and ending times of the assigned period – after the mapping – are denoted as $\hat{t}_{start}$ and $\hat{t}_{end}$. Consequently, resources for a reservation res are allocated within the interval $[\hat{t}_{start}^{res}; \hat{t}_{end}^{res}[$.

A detailed study of the impact of the granularity on the responsiveness and the efficiency of the reservation system is presented in the next chapters.

5.3 Identifying an Adequate Granularity

The *granularity* defined as c_s for a timeslot-based resource allocation management has a direct influence on the efficiency of the reservation system. On the one hand, choosing a coarser granularity results in a higher probability that assigned periods of two accepted reservations overlap while the requested periods might not. On the other hand, a finer granularity results in a higher number of timeslots having to be managed by the reservation system, resulting in a slower handling of requests. In section 5.3.1 the overlapping probabilities for an infinitesimal and non-infinitesimal granularity are analyzed. The results are then used in section 5.3.2 to identify the overhead caused by a non-infinitesimal granularity. Section 5.3.3 gives a quantitative analysis for expected number of timeslots in a dynamic timeslot model.

5.3.1 Overlapping Probability

In the following, the assigned ending time $\hat{t}_{end}$ of a reservation in the book-ahead interval and its duration $\hat{t}_{end} - \max\left\{\hat{t}_{start}; \hat{t}_0 = \left\lfloor \frac{t_0}{c_s} \right\rfloor c_s\right\} =: d$ are considered. Let t_0 denotes the point in time at which the reservation request is received. The assigned starting time is indirectly given by $\hat{t}_{start} = \hat{t}_{end} - d$ which is obviously greater than or equal to the point in time the request is received. For infinitesimal granularity and requested starting and ending times t_{start} and t_{end} it holds that $\hat{t}_{start}$=max{t_{start},t_0}, $\hat{t}_{end} = t_{end}$ and $d = t_{end} - t_{start}$.

In the following analysis, the probability for an overlap between a new request *req* and an accepted reservation *res* is determined by means of the position of the ending time of the new request. The interval within the location of this ending time would cause an actual overlap between the request *req* and the reservation *res* is denoted as *overlap-interval* c_{ol} of *res* regarding *req*. The overlap interval can be divided into two parts. The first one is given by $]\hat{t}_{start}^{res}; \hat{t}_{end}^{res}[$ as a location of the ending-time within this interval leads to an overlapping between the rear part of the request and the front part of the reservation. It is denoted as *reservation-caused* part c_{ol}^{res}, while the other part is named *request-caused* part c_{ol}^{req} of the overlap-interval. It is defined through $[\hat{t}_{start}^{res}; \hat{t}_{end}^{res} + d^{req}[$. If the ending-time of the request is located in this part, the overlap appears between the rear part of the reservation and the front part of the request. Both parts of the overlap-interval are depicted in Figure 5-3.

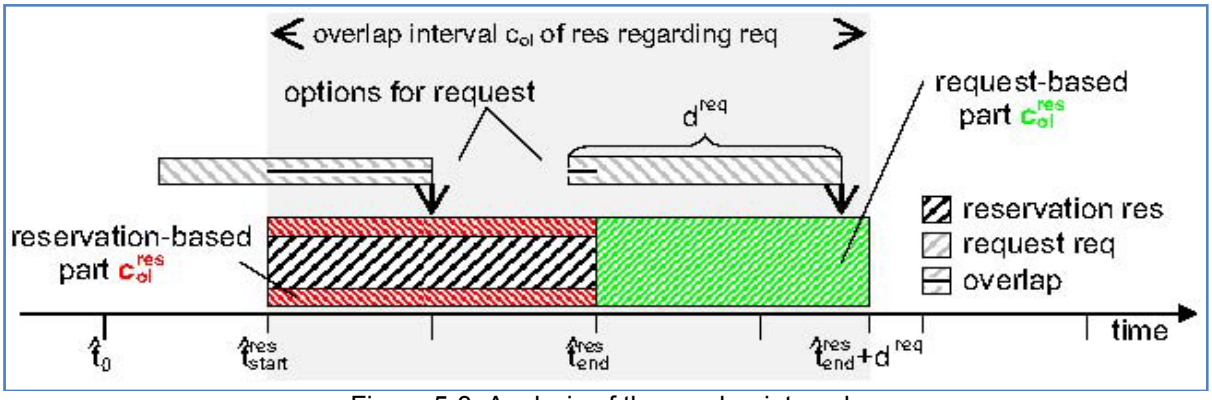

Figure 5-3: Analysis of the overlap-interval.

Assuming an independent and uniform distribution for the ending times of the reservations, the probability for an overlap between a single reservation and a new request is determined by the ratio of the overlap-interval to the book-ahead interval. To determine the expectation for a set of n reservations, it is sufficient to consider only reservations and requests with the average duration of $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d^{res_i}$ where the durations can be arbitrary distributed. Note that only reservations which are accepted by the reservation system are considered, so an ending-time has to be in the book-ahead interval. Furthermore, it is assumed that the average duration $\bar{d}$ of the reservations is an integer multiple of the slot length c_s while c_s is an integer multiple of the user-specific granularity g.

5.3.1.1 *Infinitesimal Granularity – A lower Boundary*

The lower boundary for the expected number of reservations a new request overlaps can be determined by analyzing a reservation system with an infinitesimal granularity. In this case, the accepted reservations are not adjusted, resulting in no unnecessary overlapping.

For the following analysis, two further helpful identifiers are introduced: The first one denotes the part of the overlap-interval in which the ending time of a new request can fall. It is named *limited overlap-interval*, written as $c_{lim.o}$. Analogously, the part of the book-ahead interval in which the ending time of a request can fall is denoted as *limited book-ahead interval*.

The probability for an overlap between a single reservation and a new request is given by the ratio between the length of the limited overlap-interval and the length of the limited book-ahead interval, as a uniform distribution of the ending-times is assumed. The former one is determined by the sum of the lengths of the reservation-caused part and the request-caused part of the limited overlap-interval. The latter one can be computed by $c_{ba} - \bar{d}$, as the earliest possible starting-time is the point in time the request is received. The assumption that all ending-times are independent allows for the use of a binomial distribution to compute the expectation that a new requests overlaps n accepted reservations. So, only the lengths of the reservation-caused and the request-caused part of the limited overlap-interval have to be determined.

To simplify the following analysis, without loss of generality $\hat{t}_0 = 0$. First, the possible ending-times for a new request with an average duration $\bar{d}$ are considered. The definition of d ensures that the ending-time has to be in [$\bar{d}$; c_{ba}], justified by the same argument which is used to determine the length of the limited book ahead interval.

With this information, the average lengths of the reservation-caused and the request-caused part of the limited overlap-interval (depending on the position of its ending-time) can be examined. Firstly, the length of the reservation-caused part is determined. For this, the book-ahead interval can be divided into three sections in which the ending-time can fall (Figure 5-4.a). Obviously, if the ending-time t_{end} of the reservation falls in [0; $\bar{d}$], there is no part of it in [$\bar{d}$; c_{ba}]. If $t_{end} \in]\,\bar{d};\, 2\bar{d}]$, the length is given by $t_{end} - d$, otherwise the length is determined by $\bar{d}$.

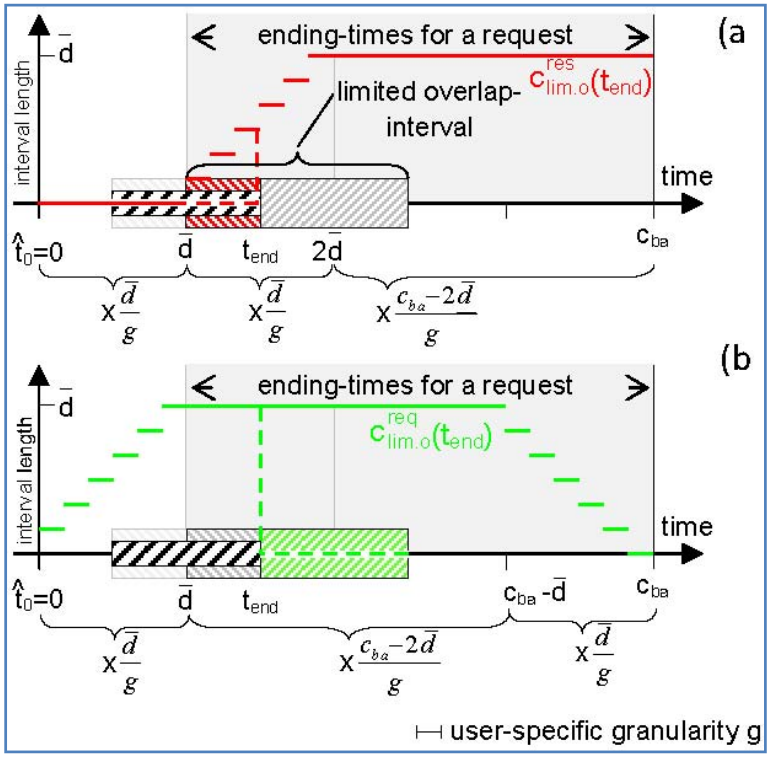

Figure 5-4: The length of the reservation-caused (red) and request-caused (green) part of the limited overlap-interval as a function of the position of its ending-time tend.

The possible ending-times for a reservation *re*s are limited to the discrete set of points in time which covers $(\frac{1}{g}c_{ba})$ elements given by $\hat{t}_{end} = ig \mid 1 \le i \le \frac{1}{g}c_{ba}$. This is caused by the user-specific granularity *g*. So, the average length of the reservation-caused part of the limited overlap interval for a single reservation can be calculated by

$$\begin{aligned}\bar{c}_{lim.o}^{res} &= \frac{1}{\frac{1}{g}c_{ba}}\left(\sum_{i=1}^{\frac{\bar{d}}{g}} 0 + \sum_{i=\frac{\bar{d}}{g}+1}^{2\frac{\bar{d}}{g}} (ig-\bar{d}) + \sum_{i=2\frac{\bar{d}}{g}+1}^{\frac{c_{ba}}{g}} \bar{d}\right) \\ &= \bar{d} - \frac{\bar{d}}{c_{ba}}\left(\frac{3}{2}\bar{d} - \frac{g}{2}\right).\end{aligned}$$

Now, the length of the request-caused part of the limited overlap-interval can be determined analogously to the reservation-caused part. The book-ahead interval can be divided in three sections, as well (Figure 5-4.b).

If t_{end} falls in [0; $\bar{d}$], the length of the request-caused part is tend. Within] $\bar{d}$; c_{ba} − $\bar{d}$], the length is $\bar{d}$, while the length for an ending-time in]c_{ba} − $\bar{d}$; c_{ba}] is determined by c_{ba} −t_{end}. So, the average length of the request-caused part of the limited overlap-interval can be calculated by

$$\begin{aligned}\bar{c}_{lim.o}^{req} &= \frac{1}{\frac{1}{g}c_{ba}}\left(\sum_{i=1}^{\frac{\bar{d}}{g}} ig + \sum_{i=\frac{\bar{d}}{g}+1}^{\frac{c_{ba}-\bar{d}}{g}} \bar{d} + \sum_{i=\frac{c_{ba}-\bar{d}}{g}+1}^{\frac{c_{ba}}{g}} c_{ba} - ig\right) \\ &= \bar{d} - \frac{\bar{d}}{c_{ba}}\bar{d}.\end{aligned}$$

The expectation for the entire length of the limited overlap interval is computed by

$$\begin{aligned}\bar{c}_{lim.o} &= \bar{c}_{lim.o}^{res} + \bar{c}_{lim.o}^{req} \\ &= 2\bar{d} - \frac{\bar{d}}{c_{ba}}\left(\frac{5}{2}\bar{d} - \frac{g}{2}\right).\end{aligned}$$

The probability for an average request to overlap an average reservation in a system with arbitrary granularity is determined by the ratio between the expectation for the length of the limited overlap-interval and the length of the limited book-ahead interval. As mentioned before, the latter interval is equal to [$\bar{d}$; c_{ba}]. So, the probability for an overlap between an average reservation and an average request can be computed by

$$\begin{aligned}p_{ol} &= \frac{\bar{c}_{lim.o}}{(c_{ba}-\bar{d})} \\ &= \frac{\bar{d}}{c_{ba}(c_{ba}-\bar{d})}\left(2c_{ba} - \frac{5}{2}\bar{d} + \frac{g}{2}\right).\end{aligned}$$

5.3.1.2 *Non-infinitesimal Granularity*

The studies made for a reservation system with an infinitesimal granularity are similar to those which are made for a system with a non-infinitesimal granularity. The main difference between them is the fact that the difference by which the requested starting and ending times are rounded to the assigned times has to be

considered additionally. This difference will lead to a larger overlap-interval compared to that of a system with infinitesimal granularity, which again results in a higher probability that an overlap occur. Obviously, the difference between this probability and p_{ol} depends on the grain size of the book-ahead interval, defined by the number of slots for a fixed book-ahead interval.

Analogous to a system with infinitesimal granularity, the book-ahead interval can be divided in three sections into which the ending time of a reservation can fall. If tend is in [0; $\bar{d}$], the assigned ending-time $\hat{t}_{end}$ is also in [0; $\bar{d}$], so the reservation-caused part of the overlap-interval does not contribute to the limited overlap-interval. An ending-time tend falling in] $\bar{d}$; 2 $\bar{d}$] results in a reservation-caused part of the limited overlap-interval with a length of $\left\lceil t_{end} - \frac{\bar{d}}{c_s} \right\rceil c_s$ as shown in Figure 5-5, as $\hat{t}_{end}$ was rounded up to the upper boundary of the slot that contains tend.

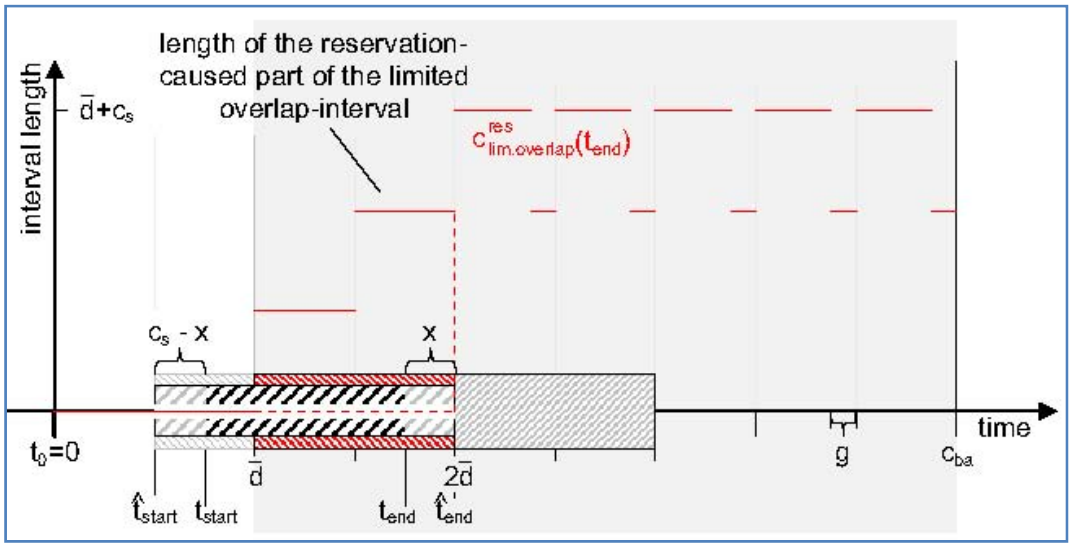

Figure 5-5: Length of the reservation-caused part of the limited overlap-interval for slotted granularity.

The assumption that $\bar{d}$ is an integer multiple of c_s leads to two different sub-cases that can occur, if $t_{end} \in]2\ \bar{d}; c_{ba}]$. If tend falls on a slot boundary, the starting-time t_{start} also falls on a slot boundary which means that the duration of the assigned period remains unchanged compared to the duration $\bar{d}$ of the requested period. Otherwise, the length is determined by $\bar{d}+c_s$, as t_{end} is rounded up about x and t_{start} is rounded down about c_s-x as depicted in Figure 5-5. Due to the uniform distribution of the ending-times, the ratio of occurrence of these two sub-cases is $1 : \frac{c_s}{g} - 1$. So, the average length of the reservation-caused part of the limited overlap-interval using a slotted granularity of c_s can be determined by

$$\bar{c}_{lim.o}^{res} = \frac{1}{\frac{1}{g}c_{ba}} \left(\sum_{i=1}^{\frac{\bar{d}}{c_s}} 0 + \sum_{i=\frac{\bar{d}}{c_s}+1}^{\frac{2\bar{d}}{c_s}} \frac{c_s}{g}(ic_s - \bar{d}) + \sum_{i=\frac{2\bar{d}}{c_s}+1}^{\frac{c_{ba}}{c_s}} \left(\left(\frac{c_s}{g} - 1 \right) (\bar{d} + c_s) + \mathbf{1}\bar{d} \right) \right)$$
$$= (\bar{d} + c_s) - \frac{\bar{d}}{c_{ba}} \left(\frac{3}{2}(\bar{d} + c_s) - g\frac{2\bar{d} - c_{ba}}{\bar{d}} \right).$$

Extending the requested period of the reservation request to the next slot boundaries does not affect the length of the request-caused part of the overlap-interval. The ending-time of a new average request req falls into the request-caused part of the overlap-interval only if $t_{end}^{req} \in]\hat{t}_{end}; \hat{t}_{end} + \bar{d}[$. Otherwise, t_{end}^{req} is greater than ($\hat{t}_{end}$ + $\bar{d}$), so t_{start}^{req} is also greater than $\hat{t}_{end}$. The fact that $\hat{t}_{end}$ is a slot boundary ensures that t_{start}^{req} is rounded down to a value equal or greater than ˆtend; this cannot lead to an overlap with the reservation req. The length of the request-caused part of the limited overlap-interval is considered in the following.

If tend falls in]0; $\bar{d}$], the length of the request-caused part of the limited overlap-interval is $\left\lceil \frac{t_{end}}{c_s} \right\rceil c_s$. If $t_{end} \in$] $\bar{d}$; $c_{ba} - \bar{d}$], the length is given by $\bar{d}$, while an ending-time in]$c_{ba} - \bar{d}$; c_{ba}] results in a length of $c_{ba} - \left\lceil \frac{t_{end}}{c_s} \right\rceil c_s$. Thus, the average length of the request-caused part of the limited overlap-interval in case of a non-infinitesimal granularity can be calculated by

$$\bar{c}_{lim.o}^{req} = \frac{g}{c_{ba}} \frac{c_s}{g} \left(\sum_{i=1}^{\frac{\bar{d}}{c_s}} ic_s + \sum_{i=\frac{\bar{d}}{c_s}+1}^{\frac{c_{ba}-\bar{d}}{c_s}} \bar{d} + \sum_{i=\frac{c_{ba}-\bar{d}}{c_s}+1}^{\frac{c_{ba}}{c_s}} (c_{ba} - ic_s) \right)$$
$$= \bar{d} - \frac{\bar{d}}{c_{ba}}\bar{d},$$

which means that it remains unchanged compared to a system using an infinitesimal granularity. The entire length of the average limited overlap-interval using a non-infinitesimal granularity is computed by

$$\bar{c}_{lim.o} = \bar{c}_{lim.o}^{res} + \bar{c}_{lim.o}^{req}$$
$$= (2\bar{d} + c_s) - \frac{\bar{d}}{c_{ba}} \left(\frac{5}{2}\bar{d} + \frac{3}{2}c_s - g\frac{2\bar{d} - c_{ba}}{\bar{d}} \right).$$

Analogous to the analysis for an infinitesimal granularity, the probability that a new average request overlaps an accepted average reservation is calculated by normalizing the average overlap-interval to the length of the interval, the ending-time of the request can fall:

$$p_{ol}^{c_s} = \frac{\bar{c}_{lim.o}}{(c_{ba} - \bar{d})} = \frac{\bar{d}\left((2 + \frac{c_s}{\bar{d}})c_{ba} - \frac{5}{2}\bar{d} - \frac{3}{2}c_s + g\frac{2\bar{d} - c_{ba}}{\bar{d}}\right)}{c_{ba}(c_{ba} - \bar{d})}$$

5.3.2 Influence on the Reservation Overhead

For a system using non-infinitesimal granularity, the limit $\lim_{c_s \to g} p_{ol}$ tends to p_{ol}, which means that a smaller slot length c_s results in a lower expectation for the number of reservations a new request overlaps with. This means that a small value for the slot length c_s is preferable, as a wasting of resources is avoided.

The assumption that the reservations are distributed independently results in a binomial distribution for the probability that a new request overlaps *k* of *n* accepted reservations. The expectation of this distribution is given by *E(n, p) = np*, where *p* denotes the probability that the request overlaps a single reservation. The factor o_{ol} between the expectations using a non-infinitesimal and an infinitesimal granularity describes the overhead caused by the usage of timeslots. This factor is equal to

$$o_{ol} = \frac{E(n, p_{ol}^{c_s})}{E(n, p_{ol})} = \frac{p_{ol}^{c_s}}{p_{ol}} \geq 1$$

and depends on the length of the book-ahead interval c_{ba}, the average reservation length $\bar{d}$, the slot length c_s, and the user-specific granularity g. A value close to one means that the overhead is low, while higher value means a larger overhead. The consideration of two different use-case scenarios will illustrate the overhead depending on the granularity, specified by the slot length c_s.

Assuming a user-specific granularity with a grain size of one minute, the first scenario (SI) has a book-ahead interval c_{ba} with a length of 30 days, while the second scenario (SII) has a one of half a year. In both cases, an average reservation length $\bar{d}$ of $1_{1/2}$ hours is assumed, while slot lengths c_s is between is 1 hour and five minutes are considered.

	considered value	$\times g\ [g = 1min]$
d	$1\frac{1}{2}h$	90
c_{ba}	SI: $30d$ or SII: $\frac{1}{2}y$	SI: 43200 or SII:259200
c_s	$1h$ down to $5min$	60 down to 5

Table 1: Values assumed for SI and SII.

An overview of the scenarios is shown in Table 1. The values chosen for the studies are based on statistics for Grid jobs from the EGEE project [57].

The left diagram of figure 5 shows the overhead o_{ol} for both scenarios depending on the slot length c_s for different user-specific granularities g. The first important observation is that the length of the book-ahead interval, which differs by the factor 6 between both scenarios, has nearly no influence to the overhead.

The overhead which is caused by the usage of a slotted granularity depends on the difference between the user-specific granularity (given by g) and the granularity which is caused by the reservation system (given by c_s). In the scenarios considered here, a difference of 60 minutes leads to an expectation for the number of reservations overlapped by a new request which is 33% higher compared to a system using an infinitesimal granularity. Choosing a slot length of 15 minutes results in an expectation that is increased by about 7.5%, while a slot length of five minutes leads to an increase of about two percent. This result is in line with the statement made in [12] which says that a slot length between five and 15 minutes is convenient. The overhead o_{ol} has a linear dependency on the granularity c_s

$$o_{ol}(c_s) = \frac{\frac{c_{ba}}{\bar{d}} - \frac{3}{2}}{2c_{ba} - \frac{5}{2}\bar{d} + \frac{1}{2}g} c_s + \frac{2c_{ba} - \frac{5}{2}d + g\frac{2\bar{d} - c_{ba}}{\bar{d}}}{2c_{ba} - \frac{5}{2}\bar{d} + \frac{1}{2}g},$$

which can easily be gained by solving $.o_{ol} = p_{ol}^{c_s}/p_{ol}$. An easy and good approximation is given if g approaches zero, if the user-specific granularity g is sufficiently small compared to the slot length used by the reservation system. In this case, $o_{ol}(c_s)$ can be approximated by

$$o_{ol}(c_s) \quad = \quad \frac{\frac{c_{ba}}{\bar{d}} - \frac{3}{2}}{2c_{ba} - \frac{5}{2}\bar{d}} c_s + 1.$$

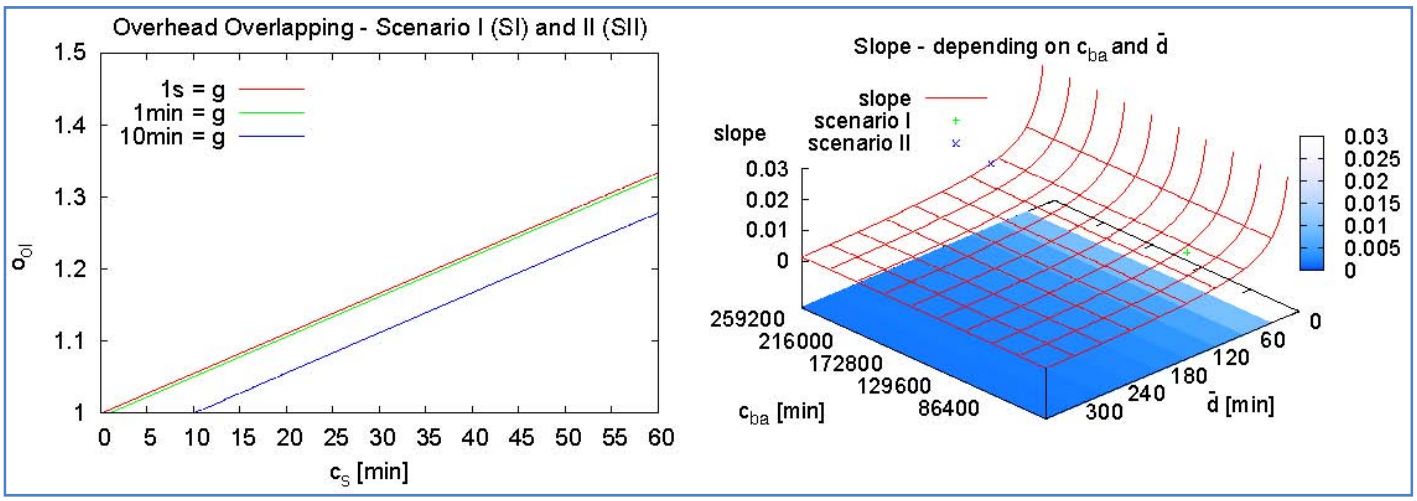

Figure 5-6: The overhead o_{ol} depending on the slot length (left) and the book-ahead interval (right).

The right-hand diagram of Figure 5-6 shows the slope of the function above depending on c_{ba} and $\bar{d}$. The observation made in the two use case scenarios that the length of the book-ahead interval has nearly no influence on o_{ol} can also be observed here: The slope for a fixed average reservation length is almost constant. Both of the two use case scenarios are plotted as blue and green dots in the diagram. By contrast, the influence of $\bar{d}$ on the slope, especially for small values of $\bar{d}$, is much greater. So, if the reservation system has to handle shorter reservations on average, shorter slot length c_s should be chosen, too. For trends and cyclic variations of $\bar{d}$, the slot length could be adapted dynamically. A detailed discussion on dynamic slot lengths can be found in section 5.3.4.

5.3.3 Influence on the Number of Timeslots

Obviously, the reservation system needs to manage a smaller number of timeslots for a coarser granularity. While this value can be directly set by the reservation system, the user also defines an upper boundary by the user-specific granularity. In the following analysis, it is not important whether a system with an infinitesimal granularity is considered. Either a granularity is given by the user-specific granularity g or by the maximum of g and the slot length c_s which is determined by the reservation system. The former is denoted as external granularity, the latter as internal granularity.

The terms and assumptions mentioned beforehand are also valid in this section. This means that the ending-times of the reservations are uniformly distributed while the durations of the reservations can be arbitrarily distributed. Furthermore, it is assumed that all reservations are independent of each other. The lower boundary of the timeslot S_i is denoted as b_{i-1}, the upper one as b_i. The boundaries of the slots which are defined by the chosen granularity are denoted as $s_j, 0 < j < \lceil c_{ba}/c_s \rceil$.

The probability that the assigned period of an accepted reservation res has the duration d is denoted as $p_{res}(d)$. Without loss of generality, a system with a granularity of $\max\{c_{clot}, g\}$ is considered.

The following analysis considers the number of timeslot boundaries of the book-ahead interval. If at a current point in time t_0 no reservation is managed by the reservation system, only a single timeslot $\left[\left\lfloor \frac{t_0}{c_s} \right\rfloor c_s;\ \left\lceil \frac{t_0}{c_s} \right\rceil c_s + c_{ba}\right[$ has to be managed. Generally, every inner slot boundary at which at least one reservation begins or ends leads to different utilized resources during the neighbouring slots. So, every inner slot boundary inside the book-ahead interval which fulfils this condition is a timeslot boundary as well. A set of n inner timeslot boundaries leads to $n+1$ timeslots that have to be managed by the reservation system. Thus, only the number of inner slot boundaries, at which at least one reservation starts or ends, has to be determined. This is done in what follows.

At first, a single slot boundary s_i and a single reservation res is considered. Using the assumption of a uniform distribution for the ending-times, the probability that the assigned ending-time $\hat{t}_{end}$ of the reservation res falls on s_i is determined by

$$p_i^{end} = \frac{c_s}{c_{ba}}.$$

The assigned starting-time $\hat{t}_{end}$ of a reservation falls on s_i, if its assigned ending-time falls on $s_{i+j}, 0 \leq j \leq \left\lceil \frac{c_{ba}}{c_s} - i \right\rceil$ and the duration of its assigned period is equal to $d = jc_s$. So, the probability that the assigned starting-time of the reservation res falls on the slot-boundary s_i can be computed by

$$p_i^{start} = \frac{c_s}{c_{ba}} \sum_{j=i+1}^{\frac{c_{ba}}{c_s}} p_r((j-i)c_s) = \frac{c_s}{c_{ba}} \sum_{d=1}^{\frac{c_{ba}}{c_s}-i} p_r(dc_s).$$

If a set of n reservations with an ending-time in the book-ahead interval is considered, the probability that at least one assigned starting- or ending-time falls on a slot boundary s_i is equal to 1 minus the probability that neither of them falls on s_i. The assumption that the reservations are independent from each other makes it possible to use the binomial distribution to determine this probability. It is given by

$$1 - \binom{n}{0} p_i^0 (1-p_i)^{n-0} = 1 - (1-p_i)^n.$$

This probability can be used to compute the expected value for the number of timeslots. The expectation is equal to 1 plus the expected number of inner timeslot boundaries. The expected number of inner timeslot boundaries is determined by the expectation for the number of inner slot boundaries where at least one reservation begins or ends. So, the expected number of dynamic timeslots $n_s(p_{req}(l))$ can be computed by summing up 1 and the probability that at least one starting- or ending-time falls on the slot boundary

s_i for all $1 \leq i \leq \frac{c_{ba}}{c_s} - 1$:

$$n_s(p_{req}(l)) = 1 + \sum_{i=1}^{\frac{c_{ba}}{c_s}-1} 1 - (1-p_i)^n$$

$$= 1 + \sum_{i=1}^{\frac{c_{ba}}{c_s}-1} 1 - \left(\frac{c_{ba} - c_s}{c_{ba}} - \frac{c_s}{c_{ba}} \sum_{d=1}^{\frac{c_{ba}}{c_s}-i} p_r(dc_s) \right)^n$$

5.3.4 Dynamic Adaptation of Granularity

In the following analysis, the expected number of dynamic timeslots is estimated by using the upper boundary n_s.

$$n_s(p_r(c_s) = 1) = \frac{c_{ba}}{c_s} - \left(\frac{c_{ba}}{c_s} - 1\right)\left(\frac{c_{ba} - 2c_s}{c_{ba}}\right)^n$$

As shown in Figure 5-6, a slot length c_s in the area of five up to 15 minutes leads to an acceptable overhead between two and 7.5% for the use-case scenarios described in Table 1.

As expected, shorter slot lengths result in a greater expected number of dynamic timeslots that have to be managed by the reservation system. The number of timeslots that are handled by the reservation system differs insignificantly for slot lengths between five and 15 minutes, if the number of accepted reservations which are handled by the reservation system is sufficiently small. The difference becomes more significant for an increasing number of reservations, whereas the maximal number of timeslots is limited by c_{ba}/c_s, which is shown for both use-cases in Figure 5-7.

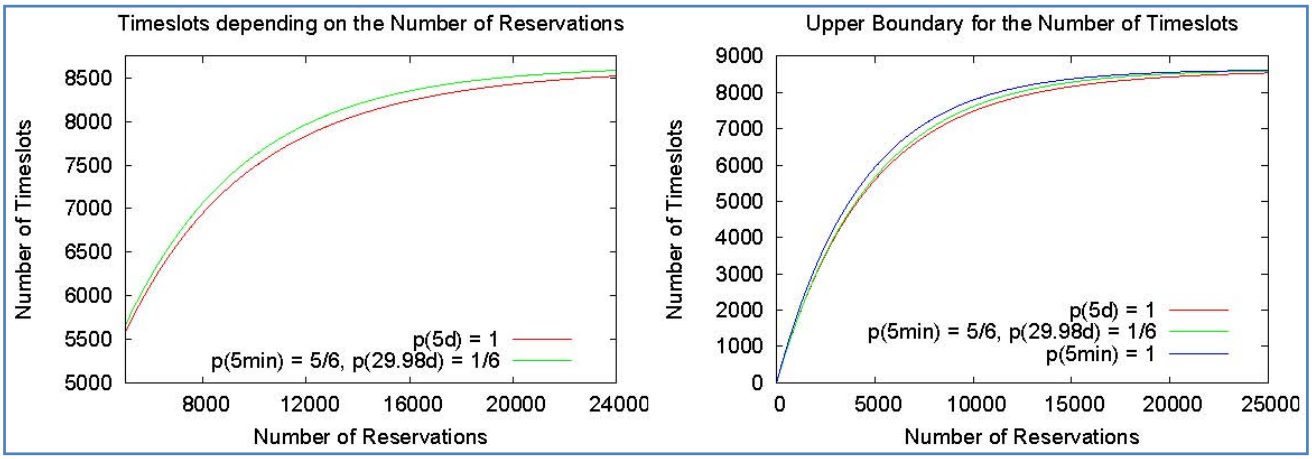

Figure 5-7: Number of timeslots for different slot sizes.

The run of the curves lead to the idea of a dynamic adjustment of the granularity for the resource management. If only a small number of reservations have to be managed, a small value for the granularity is chosen, which minimizes unnecessary overlaps. As soon as the number of reservations exceeds a well defined threshold, the reservation system increases granularity, which limits the number of timeslots which have to be managed from this point on. The reservations which are already accepted by the reservation system are not affected by the adaptation, so a remapping of accepted reservations is not necessary. The adaptation of the granularity can be performed several times for different thresholds. The precise values depend on the length of the book-ahead interval and the number of timeslots the reservation system is able to manage.

5.4 Simulative Validation

As described former section, the usage of a timeslot-based approach with a certain granularity for resource management leads to unnecessary overlaps between the accepted reservations. The equation for $o_{ol}(c_s)$ describes a linear relationship between the unnecessary overlaps and the chosen granularity. This section

validates this relationship by means of simulation. The system efficiency is analyzed by means of the bandwidth blocking ratio as well as the system responsiveness determined by the relative processing time for advance and malleable reservations.

5.4.1 Reservation System Efficiency

The bandwidth blocking ratio (BBR) [56], [8] is defined as

$$BBR := \frac{\sum_{res \in \bar{R}} a(res)}{\sum_{res \in R} a(res)},$$

where a(res) denotes the amount of data that can be transmitted, $\bar{R}$ the set of rejected and R the set of requested reservations. In case of advance reservations, a(res) is given by $a(res) = c_{bandwidth}(t_{end} - t_{start})$. Clearly, the BBR is only affected if the network load leads to an overbooking of resources. If the BBR increases due to a coarser granularity, the system efficiency decreases as requests overlap and must therefore be rejected by the admission control. Furthermore, if resources are heavily overbooked the impact of other factors diminishes. Therefore, the network loads in the simulation have been chosen accordingly.

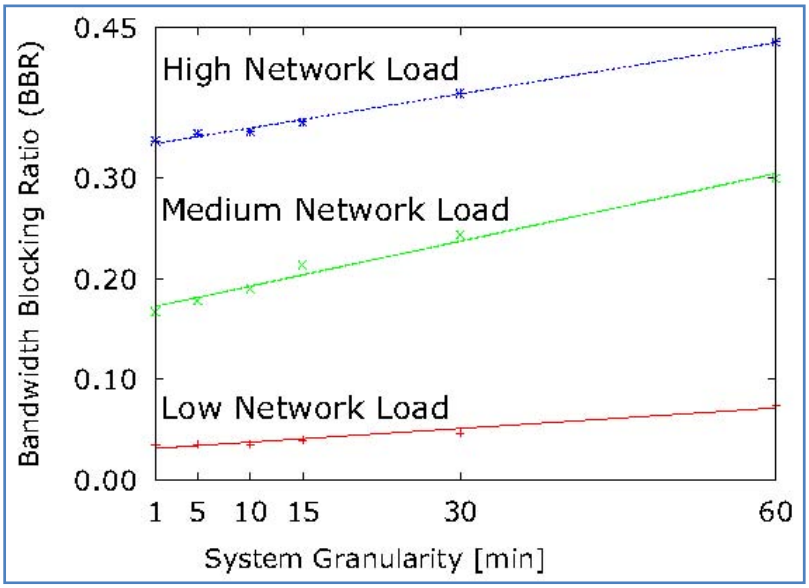

Figure 5-8: The Request and Bandwidth Blocking Ratio as a function of granularity.

Figure 5-8 shows the BBR for different granularities and network loads. The BBR is depicted for three different network loads as a function of the granularity. Here, the main issue is the slope of fitted lines. In case of the low network load, the BBR increases slowly. The performance degradation becomes more significant in the medium and high network load scenarios. As can be seen, the fitted line in the medium load scenario has the largest slope. This corresponds to the preliminary note. Overall, the diagrams confirm the linear dependency between the number of unnecessary overlaps and the slot length, as derived in section 5.3.2.

5.4.2 System Responsiveness

Following the former section, a finer granularity used for the resource management results in an increased number of timeslots to be managed by the reservation system. Increased memory consumption and processing time are typical consequences. This affects the responsiveness of a system that processes requests online. As the overall processing time depends on the implementation details, the relative processing time is analyzed.

In the pre-processing phase every timeslot covered by the requested time interval has to be considered. If a request is accepted, the allocated resources have to be reserved in the corresponding timeslots in the post-processing phase. Consequently, the processing time is analyzed for both phases. The diagrams of Figure 5-9 depict the average pre- and post-processing times as well as the average entire processing times of advance and malleable reservation requests for different granularities. The processing times are scaled to the average entire processing time for a slot length of five minutes. The results are depicted for a path computation based on shortest distance paths (a widest shortest strategy leads to similar results) [55]. They confirm that coarser granularities increase pre- and post-processing time. Of course, the exact relationship depends on the implementation of the resource management.

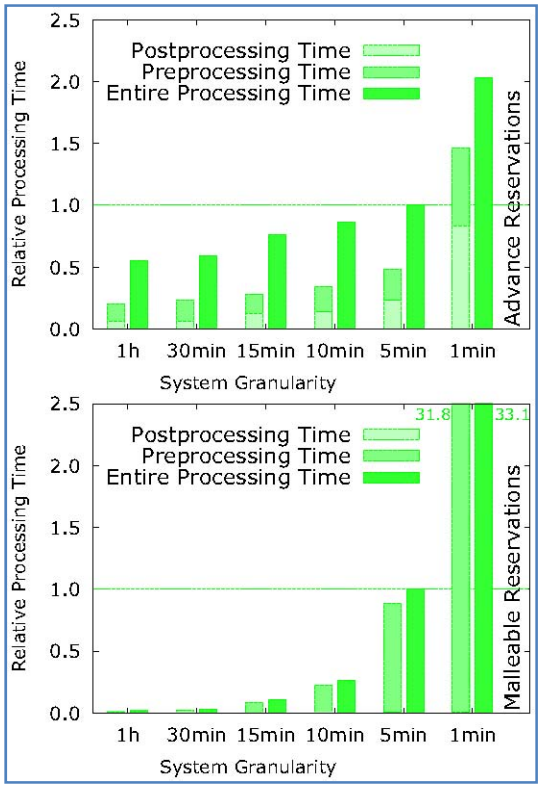

Figure 5-9: Pre- and post- and path processing times.

The overall processing time of malleable reservations depicted in the lower diagram is more dependent on the slot length than the processing time of advance reservations depicted in the upper diagram. The set of reasonable configurations of a malleable reservations request depends on the number of timeslots. This

analysis shows that the time needed to serve advance reservations is about two orders of magnitude smaller than the time needed for malleable reservations in general. If a granularity in the range of five and 15 minutes and networks being not too large and dense are considered, the time which is needed for the average path computation phase is relatively small compared to the entire processing time. This is also confirmed by studies performed for other topologies.

5.5 Conclusion

The main objective of this chapter was a detailed study of the impact of granularity on the responsiveness and efficiency of a reservation system with special focus on network resources for Grid environments.

It was shown, that an admission control mechanism based solely on the inspection of already admitted reservations is only suited if the number of reservation requests is low. A concept called timeslot is introduced to store accumulated resource usage information and thus to reduce processing time. However, granularity had to be introduced to decouple the number of timeslots needed from the number of accepted reservations. The granularity can be seen as a lower boundary of slot length. For static timeslots with a uniform size the granularity is equal to the slot size. For dynamic timeslots, the slot length is an integer multiple of the granularity. However, this concept needs a mechanism that adjusts reservations to granularity. This was done by rounding the request times to slot boundaries so that the requested interval is included in the reservation.

The resulting overhead was modelled mathematically in section 5.3.2 by comparing the overlapping probability of accepted requests in a system with infinitesimal granularity to the probability in a system with a non-infinitesimal granularity. For systems with a dynamic timeslot mechanism the expected number of timeslots was analyzed.

The efficiency of the reservation system with a non-infinitesimal granularity was evaluated by simulation in 5.4. Here, a linear dependency between the number of unnecessary overlaps and the slot length, as derived before, is confirmed. The section closes with a study of the impact of granularity on the processing times of advance and malleable reservation requests. For advance reservation requests, the efficiency in terms of Bandwidth Blocking Ratio is smaller when using a coarser granularity while the processing time of the requests is only slightly affected. However, for malleable reservations the overhead for coarser granularity is negligible, as data transmission rates can be adapted to the assigned interval while the processing time benefits to a high extent. This is because the affected pre-processing phase in general needs to be repeated for multiple configurations of the malleable reservation.

6 Routing and Scheduling Aspects

This chapter presents evaluation of selected topics described in previous chapters. In the next two sections the distributed and centralized approaches to advance reservations are evaluated by means of simulative evaluation. Subsequently, the type Malleable Advance Reservations is analysed in terms of various strategies to map a reservation request to resources.

6.1 Distributed versus Centralized Advance Reservation of Computation Resources

In order to examine the efficiency of a distributed or a centralized architecture for the advance reservation of computation resources we formulate the problem and then perform simulation experiments. In particular, we assess the performance of two typical meta-scheduling algorithms, namely: the Earliest Start Time and the Earliest Completion Time both extended in order to support advance reservations.

6.1.1 Problem Formulation

We consider a Computational Grid environment consisting of users, computational resources (clusters), connected through a network with links *l* of known propagation delays d_l. We assume that a user generates tasks in the form of requests and forwards them to the appropriate 'scheduler' (meta-schedulers, MS). There is one central or a number of distributed schedulers located at the users' sites (one scheduler per user). Users, computational resources and scheduler(s) are placed at nodes, so that a node may have all, some or none of these entities. In the distributed approach there is one distributed scheduler for every node that has a user, while in the centralized approach there is only one scheduler in the whole network to which all the users forward their task submission requests.

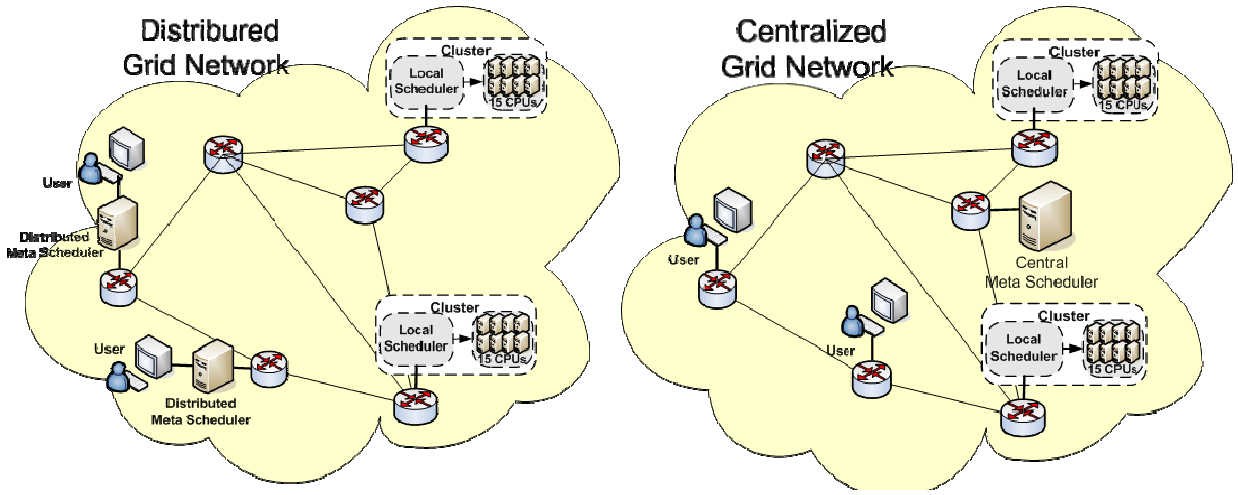

Figure 6-1: (a) Centralized and (b) Distributed Grid environment. Each node can have a user, a computational resource (cluster), a meta-scheduler (if we have selected centralized or distributed architecture we have one or more meta-schedulers), or none of these entities.

A User *U* generates tasks with probabilistic characteristics (workload, deadline, and requested number of CPUs) and probabilistic inter-arrival times. A task *i* requests to be executed on *r* CPUs (e.g. MPI parallel program) and has a computational workload L_i per CPU measured in millions instructions (MI), and a non-critical deadline $t_{deadline\ i}$, measured in seconds: by "non-critical" we mean that if the task's deadline expires the task remains in the system until it is executed. Note that we have assumed that we know the exact workload of a task before its execution. Although this assumption is quite strong the conclusions that we draw are general since our main goal is the comparison of centralized and distributed architectures.

Each computational resource (cluster) *P* contains a number of CPUs (W_P), each with the same processor computation capacity (C_P) measured in millions instruction per second (MIPS). The CPUs employ a space-shared (non-preemptive) architecture; so that once a task has started its execution on *r* CPUs it cannot be stopped and has to complete execution on those CPUs. Every computational resource has a local queue where arriving tasks are stored, and a local queue scheduler which assigns the tasks of the local queue to the available CPUs. We have assumed that the local queue scheduler serves tasks in the FCFS order and can schedule the tasks to be executed in the future (in advance). In this way, a task that arrives at the queue and request to be executed at some specific time (which we will call Starting Time - t_{start}) reserves *r* CPUs for the duration that it requests (fixed advance reservation). If another task arrives later and its requested execution period overlaps with the previously reserved period (assuming that no other CPUs are available), this new task is not served, even if it requests to start earlier. We will call this problem a scheduling "conflict", which we solve by running a rescheduling algorithm locally at the computational resource. Thus, the number of "conflicts" is a degree of the reliability of the advance reservation planning of the meta-scheduling algorithm.

In the case of the centralized architecture the central meta-scheduler keeps track of the utilization of all resources which is always up-to-date, by maintaining one *cluster availability profile* per computational resource (which is a data structure commissioned for this purpose – section 4.1.2). In the distributed architecture, each

distributed meta-scheduler maintains a "picture" of the utilization of all the computational resources. The locally maintained picture can be different among the distributed meta-schedulers and depends on their position in the Grid network, since we have considered a network with non-zero propagation delays. Update information (in the form of update messages as described in section 6.1.2) is communicated between resources and distributed meta-schedulers in order to synchronize the locally maintained resources' profiles with the actual utilization.

Whenever a user creates a new task he immediately forwards the task characteristics to the corresponding meta-scheduler (central or personalized-distributed meta-scheduler), in the form of a task request. The meta-scheduler, depending on whether we have a centralized or distributed architecture, has a complete or partial knowledge of the utilization of the CPUs of the clusters-sites. Based on this information, the meta-scheduler executes the corresponding algorithm and returns back to the user the assignment decision and an estimated Starting Time ($t_{start\ i}$) after which the task *i* is going to be executed at the selected computational resource. With the assignment decision and the calculation of the estimated $t_{start\ i}$ (we have assumed that the execution duration is known) a fixed advance reservation (section 3.3.2) is defined. The user immediately sends the tasks to the selected computational resource. We assume that shortest path routing is used in the network, and the communication delays consist of the propagation and transmission delays (without considering congestions).

In the centralized architecture the $t_{start\ i}$ estimation is always correct and the fixed advance reservation can be performed (is feasible), since the central meta-scheduler has assigned all the previous tasks to the computational resources and thus always has a complete knowledge of the resources' utilization, assuming that the workload of all the tasks is known in advance. On the other hand, in the case of a distributed scheduling architecture, the computed $t_{start\ i}$ can be incorrect; since the utilization profile of the destination may have changed by the time task *i* arrives at that resource ($t_{start\ i}$ calculation is based on outdated knowledge of the resources' utilization). This is a problem that cannot be avoided by any distributed algorithm in a network that has nonzero propagation delays.

When the task arrives at a computational resource it is locally queued and request to be executed at the estimated $t_{start\ i}$ (fixed advance reservation). If this is feasible the local scheduler performs the advance reservation. However, if we are using a distributed scheduling architecture and one or more other tasks have arrived to this specific resource after the task assignment decision, task *i* cannot be executed at the predicted $t_{start\ i}$, in other words the fixed advance reservation cannot be performed, and we have a conflict. When a task finishes its execution, the resource returns the results back to the originating user.

Intuitively, a centralized architecture will have good performance in small networks (where the delay to question and obtain the reply from the central meta-scheduler would be small) and in high loads (in which there would be no overhead to distribute the cluster utilization information and synchronize the distributed meta-schedulers). However, there are various drawbacks in a centralized architecture such as having a single point of failure in the Grid network and the fact that gathering all requests in a single computer would demand a quite strong and highly connected unit to tackle the workload. Therefore, distributed solutions have received considerable attention lately.

6.1.2 Cluster Utilization Profiles, Utilization Update messages and meta-scheduling algorithms

Clusters Utilization Profiles

We assume that the meta-scheduler(s) keep(s) track of the utilization profiles of the clusters by using the cluster availability profile data structure as presented in section 4.1.2.

In the case of the centralized architecture the central meta-scheduler maintains one *cluster availability profile* $C_P(t)$ for every computational resource *P*. These profiles contain always up-to-date information for the resources' utilization, assuming the execution times of the tasks are known in advance. In the distributed architecture, each distributed meta-scheduler maintains a "picture" of the utilization of all the resources by storing one $C_P(t)$ for every resource *P*. The locally maintained availability profiles are updated by exchanging update messages (as described in the next paragraph).

Utilization update messages

In the Grid network model that we have considered, the centralized architecture doesn't require the exchange of utilization information. However, in the distributed architecture, each distributed meta-scheduler needs to maintain a "picture" of the utilization of all the resources to efficiently assign the tasks to the resources. In order to synchronize the locally maintained "picture" with the actual resources' utilization, in the distributed scheduling architecture utilization update messages should be sent to each distributed meta-scheduler.

We have assumed an "exhaustive" update strategy. An update message can be sent at the time a task arrives to a computational resource. At that time, the local queue scheduler can calculate the time at which that specific task will start to execute, let's call it δ_i, by taking into account the tasks that have scheduled ahead of it or already being executed. The update messages are routed in the Grid network over the corresponding shortest path. However, in order to maintain the control plane overhead low, update messages are not sent to all distributed meta-schedulers, but to those that can use this information, i.e. a meta-scheduler is not informed about a task assignment in a computational resource if the task will complete its execution before the earliest time at which that meta-scheduler can send a task to that resource.

More precisely: consider a computational resource (cluster) *P* in which the task *i,* is going to execute after time δ_i. Resource *P* informs the distributed meta-scheduler *N* that a task is going to start to execute after time δ_i and for duration L_i / C_P if:

$$2 \cdot d_{P,N} \leq \delta_i + \frac{L_i}{C_P} \qquad (1)$$

where $d_{P,N}$ is the shortest path delay from *P* to *N*. Update messages are sent to all meta-schedulers that satisfy the above inequality (Eq. (1)). All other meta-schedulers are not informed about this particular fixed advance reservation since they can't use this information. This is due to the fact that the update packet will

reach *N* after $d_{P,N}$ and the earliest time after which *N* can assign a task at *P* is after an additional $d_{P,N}$, thus $2 \cdot d_{P,N}$ in total (assuming that the propagation delay of each link is the same in both directions).

A distributed meta-scheduler *N* that receives a utilization update message uses this information in order to update the cluster availability profiles ($C_P(t)$) that it maintains locally for the corresponding computational resource *P*.

Employed meta-scheduling algorithms

To simplify the presentation and the experiments, we will assume that each task requests one processor, that is r=1 (cf. section 4.1.2), which is the most usual case. For the scope of this study we have employed two typical meta-scheduling algorithms and extend them to support advance reservations. Specifically, the meta-scheduling algorithm decides the computational resource to assign the task and calculates the starting time of the execution, defining (planning) in this way a fixed advance reservation. Each algorithm was implemented in a centralized and a distributed version.

Earliest Start Time algorithm (EST)

The EST algorithm selects the computational resource (cluster) in which the task will start to execute earlier. When the algorithm computes the time at which the task will start to execute at a cluster it takes into account the propagation delay and the cluster's utilization information (accurate or outdated knowledge depending on the selected architecture). More formally: Assuming that user *U* wants to execute task *i* which has computation workload L_i, and deadline $t_{deadline\ i}$, the EST algorithm calculates for a cluster *P* with computation complexity C_P the earliest start time $EST_{P,i}$. $EST_{P,i}$ is defined as the earliest time, greater than the propagation delay from *U* to *P* ($d_{U,P}$), after which one processor is available for L_i/C_R seconds, given *P*'s availability utilization profile $C_p(t)$. The EST algorithm selects the cluster *R* with the minimum $EST_{P,I}$: ($R = \arg\min_{\text{all } P} (EST_{P,i})$), and the starting time of task *i* is defined as $t_{start\ i} = EST_{R,}$

Earliest Completion Time algorithm (ECT)

The ECT algorithm is similar to the EST but also takes into account the computation capacity of each cluster. Extending the previous definition: the ECT algorithm computes for a cluster *P* the earliest completion time $ECT_{P,i}$. $ECT_{P,i}$ is defined as the earliest time, greater than the propagation delay from *U* to *P* ($d_{U,P}$), before one processor is available for L_i/C_R sec, given $C_p(t)$. The algorithm selects the cluster *R* with the minimum $ECT_{P,i}$ ($R = \arg\min_{\text{all } P} (ECT_{P,i})$) and the starting time of task *i* is defined as $t_{start\ i} = ECT_{R,i} - L_i/C_R$.

When a task arrives at computational resource it requests to be executed at time $t_{start\ i}$ (fixed advance reservation request). In the distributed architecture, there is a probability that this is not feasible (i.e. another overlapping reservation has been made). In this case we say that we have a conflict. A conflict is solved locally at the cluster by applying a simple algorithm that searches for the earliest placement of the task (taking into account all previously scheduled tasks). The computed value δ_i – as presented in the previous section – defines a new fixed advance reservation. An update message is finally send that inform the distributed meta-schedulers for the finally performed advance reservation.

6.1.3 Performance Results

We have conducted full network simulation experiments. We have extended the ns-2 platform and incorporated the appropriate entities. More specifically, we have designed appropriate cluster utilization data structures; we have developed appropriate ns-objects for the users, computational resources and meta-schedulers and also defined a new ns-2 packet format for the processors update messages.

For the simulations we have assumed a 5x5 mesh with wraparounds topology. The distance of adjacent nodes was initially set equal to 400km and we also experimented with 100km and 1600km distances. In this mesh topology we placed 4 computational resources (clusters) with total number of 100 CPUs. Figure 6-2 shows the network topology that we used in our experiments. Based on this Figure cluster nodes were placed at nodes 7, 9, 17 and 19. When we use a centralized architecture the central meta-scheduler was placed at node 13. We have examined the performance when the cluster resources are homogenous and not: we have assumed that we have (i) equal number of CPUs per cluster all with equal processor computation capacity and (ii) uneven number of CPUs per cluster and uneven processor computation capacity between clusters (the CPUs of a cluster have equal computation capacity).

Users were placed at all the nodes of the network (25 in total) and tasks were generated probabilistically. When a task is created its owner is chosen with equal probability among the 25 users. The inter-arrival time between two tasks is a random variable (r.v.) that follows an exponential distribution with average λ tasks per sec. The computation complexity of a task is a r.v. that follows an exponential distribution with average L millions instructions. Finally, the deadline of a task is, also, a r.v. that follows an exponential distribution with average $t_{deadline}$ sec. However, in order to create always tasks that can execute in the Grid network, the deadline of a task is chosen so as the task can complete its execution without exceeding it in the resource with the highest processor computation capacity.

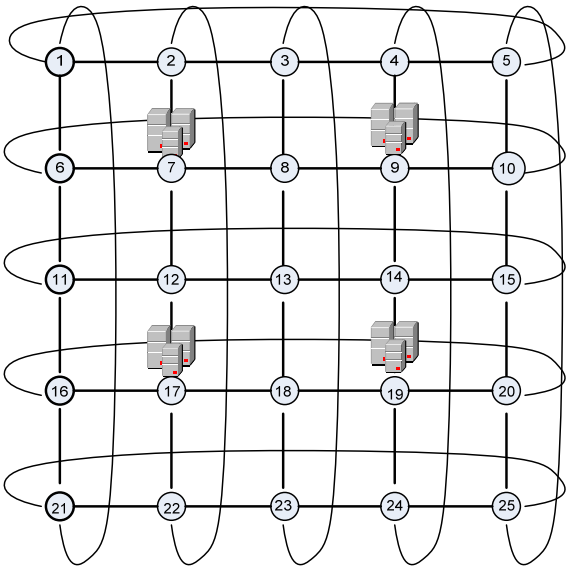

Figure 6-2: Experimented topology.

Unless explicitly stated the computation complexity of the tasks was L=50000 MI and the average deadline was $t_{deadline}$ =3 sec.

In order to assess the performance of the scheduling algorithms we have used the following performance metrics:

- Probability to miss deadline, in which we measure the tasks that completed their execution after their deadline has expired,
- Average total execution delay. The total execution delay of a task is defined as the time between the task's creation and the time at which the results have been returned to the originating user,
- Conflicts probability. This metric can be calculated only in distributed architectures and corresponds to the number of conflicts that we observed (we assume that we have a conflict when the predicted t_{start} by the scheduling algorithm is not met),
- Average number of exchanged messages, which measures the communication overhead of the employed distributed algorithms. In the centralized version the average number of exchanged messages is always 2 messages per task request (question to/reply from the central meta-scheduler). On the other hand, in a distributed scheduling architecture there is a considerable control plane overhead to "synchronize" the distributed meta-schedulers (exchange utilization update messages as presented in section 6.1.2). When a resource receives a task it informs the meta-schedulers that satisfy Eq. (1) about the period that the task is going to be executed.

6.1.3.1 *Clusters with equal number of CPUs and equal computation capacity*

In these experiments we have assumed that the four computational resources have equal number of CPUs and equal processor capacity. More specifically, each computational resource was a cluster with W_P=25 CPUs and each CPU had processor computation capacity C_P= 25000 MIPS (per CPU). It is worth noting that when all the resources have CPUs with equal computation capacity, the performance of the Earliest Completion Time algorithm (ECT) is identical to that of the Earliest Start Time algorithm (EST). Thus, in the figures of this section we graph only the performance of EST.

Figure 6-3 shows the performance of the EST algorithm when the average task arrival rate takes values between 5 and 50 tasks/sec. From Figure 6-3(a) we can observe that the distributed version of the EST algorithm exhibits smaller average total delay. The average total delay is dominated by the task execution time (L/C_P ~ 1.9 sec), while the task load affects the performance when the load takes large values. The difference between the distributed and centralized versions of the algorithm is clearer in Figure 6-3(b) which graphs the probability to miss deadline metric. In this Figure we can observe that the distributed version is always better than the corresponding centralized version. For high loads both versions seem to exhibit similar performance.

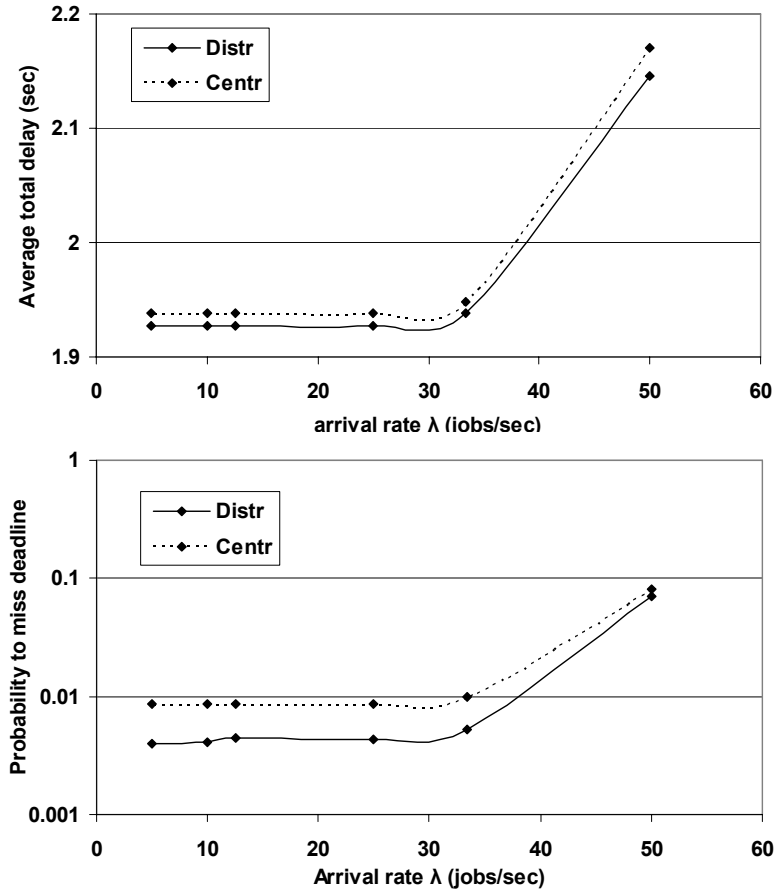

Figure 6-3: Effect of arrival rate λ: (a) average total delay, (b) probability to miss deadline.

Figure 6-4 shows the performance of the EST algorithm when the distance of the adjacent nodes was set to 100, 400 and 1600km. As expected, the increase in the distance increases the average task delay. From Figure 6-4(a) we can observe, that as the distance increases the difference between the distributed and centralized architectures increases as well. This can be explained by the fact that the increase in the network dimension affects vastly the centralized architecture by increasing the time that is required to question and obtain the reply from the central meta-scheduler. The average total delay is dominated by the task execution time (L/C_P ~ 1.9 sec), while the propagation delay plays a secondary role. From Figure 6-4(b) we can observe that for larger networks the improvement that we obtain by applying the distributed version of the algorithm is more significant. Figure 6-4(c) presents the conflict probability, only for the distributed version of the EST algorithm. As expected, the conflict probability increases as the arrival rate and the distance increases. This can be explained by the fact that the increase in the arrival rate or the network distance increases the

probability of having two (or more) distributed meta-schedulers deciding to schedule a task at the same resource at overlapping periods, before one scheduler learns the decision of the other (meta-scheduler).

Finally, it is worth noting that the average number of exchanged messages, obtained only for the distributed architecture, was almost in all cases constant and close to the upper bound 25 (we have 25 users and thus 25 distributed meta-schedulers), and for this reason we don't provide graphs for this metric. This can be explained by the fact that the average execution time of a task is L/C_P ~ 1.9 sec, which according to Eq. (1) is quite large and gives enough time for the update information to reach all the distributed meta-schedulers, even in the case that the distance of adjacent nodes is 1600km.

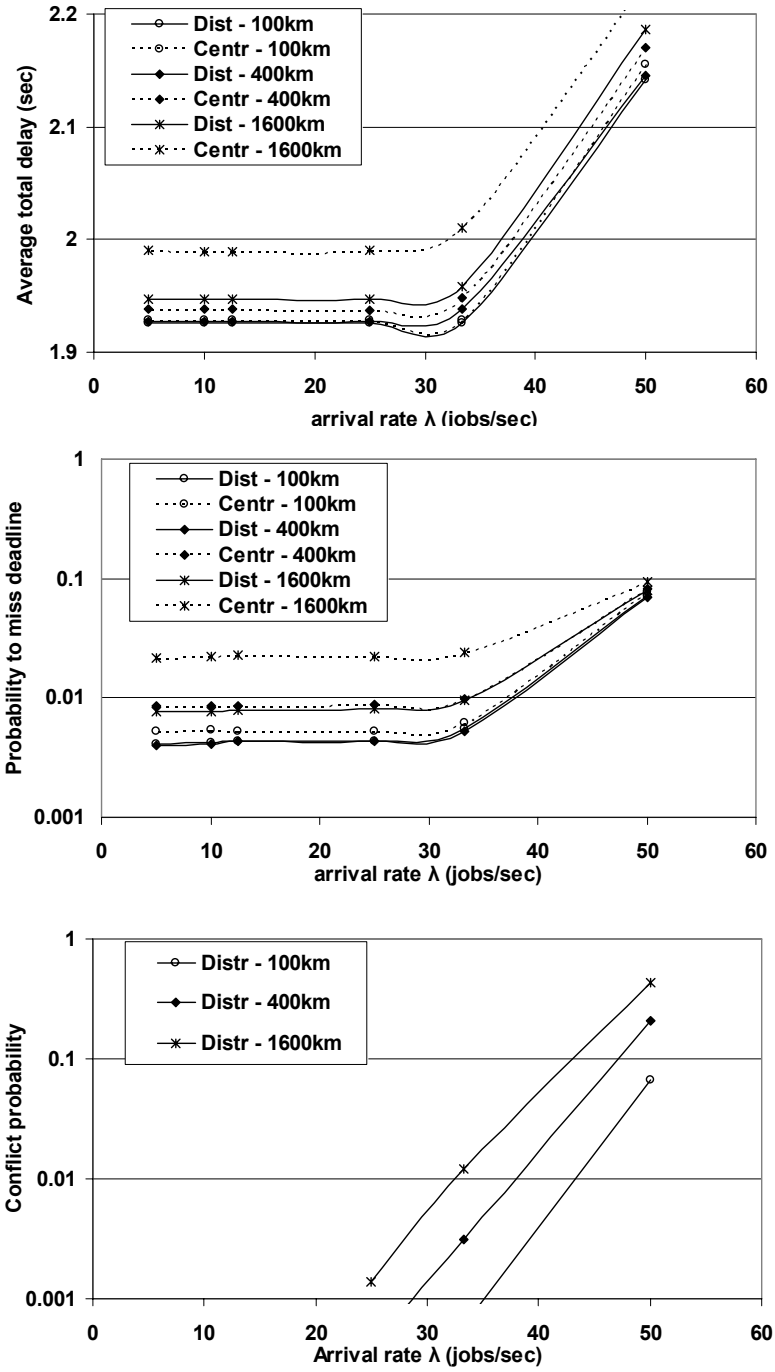

Figure 6-4: Effect of propagation delay: (a) average total delay, (b) probability to miss deadline, (c) conflicts probability (only in distributed algorithm) when the distance of adjacent nodes is 100, 400 and 1600 km.

6.1.3.2 *Clusters with uneven number of CPUs and uneven computation capacity*

In this section we present results when the four computational resources have uneven number of CPUs and uneven processor computation capacity. More specifically, with respect to Figure 6-2 the resources placed at nodes 7 and 19 are clusters with 17 CPUs and computational capacity 33000 MIPS (per CPU), and the resources placed at nodes 9 and 17 were clusters with 35 CPUs and computation capacity 19000 MIPS (per CPU). It is worth noting that the overall computation capacity of the Grid network was equal to that of the previous section.

Figure 6-5 shows the performance of the EST, ECT algorithms when applied in a distributed or a centralized scheduling architecture, while the average task arrival rate takes values between 5 and 50 tasks per second. With respect to the two algorithms, we can observe that the EST exhibits quite constant performance, while the ECT performs much better for low load, and deteriorates rapidly as the load increases. This can be explained by the fact that ECT is a greedy algorithm (when compared to EST) since it tries to schedule every task to the stronger cluster. If we imagine a series of tasks with various lengths, the EST algorithm will distribute them more even, while the ECT will try to achieve the best for each task separately, that would occasionally have a negative effect in the overall performance. It is worth noting that the performance of EST algorithm is almost similar to that presented in the previous section (when all the resources have the same number of CPUs and equal computation capacity).

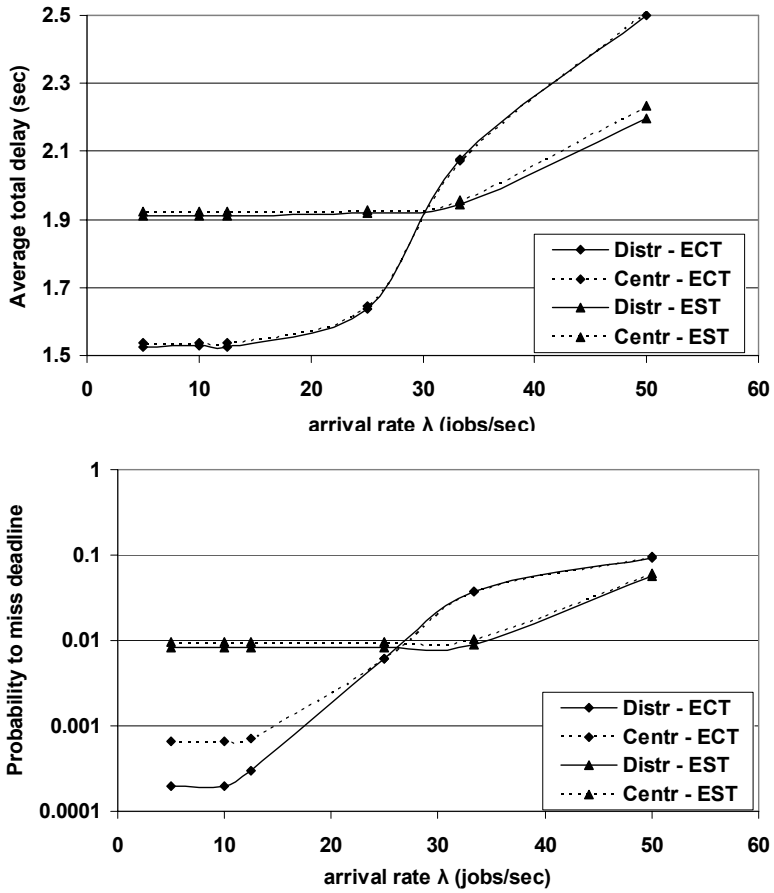

Figure 6-5: Effect of arrival rate λ: (a) average total delay, (b) probability to miss deadline.

Figure 6-6 shows the performance of the ECT algorithm when the distance of the adjacent nodes was set to 100, 400 and 1600km. We can observe that as the distance increases the improvements obtained by employing a distributed architecture are more pronounced.

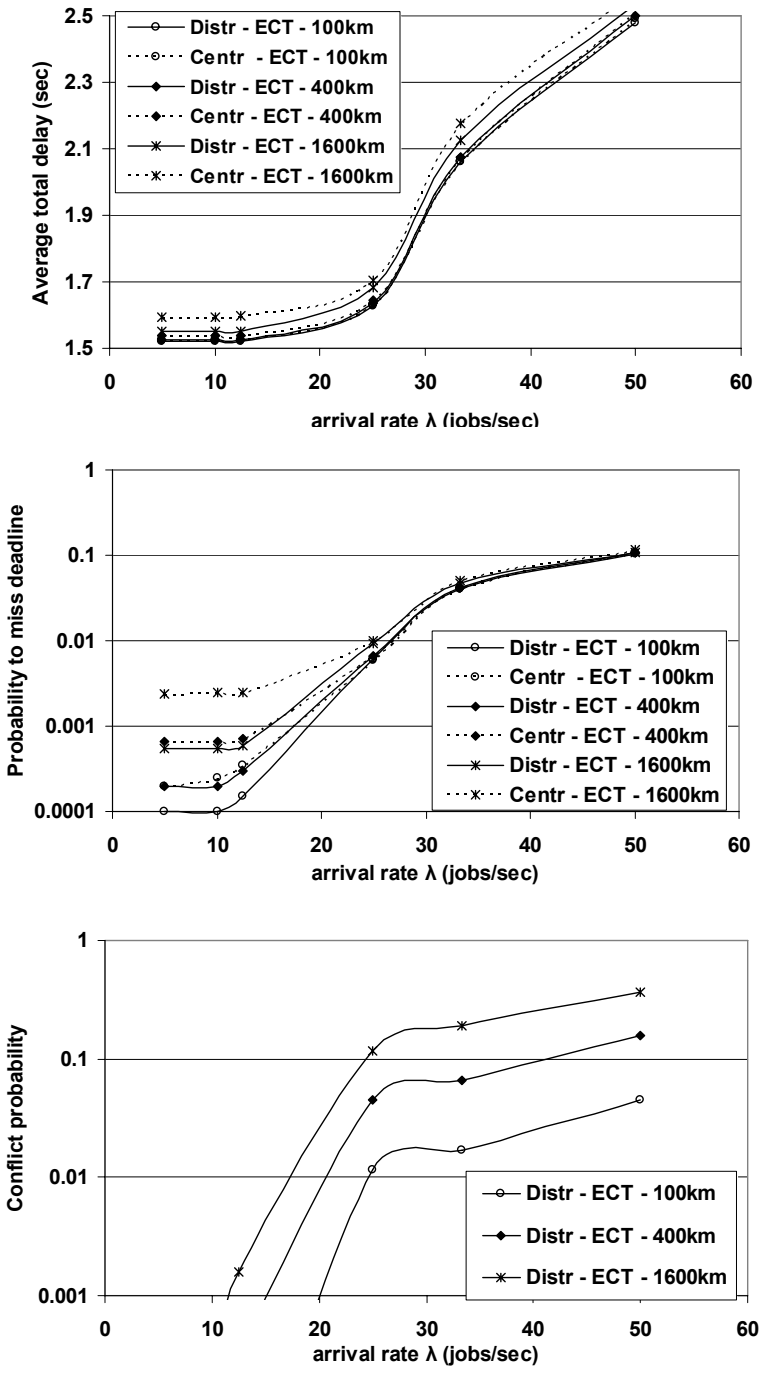

Figure 6-6: Effect of propagation delay: (a) average total delay, (b) probability to miss deadline, (c) conflicts probability (only in distributed algorithm) when the distance of adjacent nodes is 100, 400 and 1600 km.

6.2 Distributed versus Centralized Advance Reservation of Communication Resources in an OBS network

We formulate the distributed vs. centralized problem in a Grid Burst Switched network. Then we compare the performance of two routing and advance reservation planning algorithms implemented either in a centralized or in a distributed manner. The algorithms examined consider the routing and the advance reservation planning problems jointly. Thus, when the algorithms are executed they return not only the path to be followed by the burst but also the starting time after which the source should start the transmission. In this way the algorithms, apart from the path, define a sequence of fixed advance reservations to be performed at the links of the specified path. In particular, we have employed two burst routing and scheduling algorithms namely: a multi-cost heuristic algorithm (called Availability Weighting heuristic multi-cost algorithm) and a typical Dijkstra with contention avoidance algorithm (the distributed versions of these algorithms were presented and evaluated in [37]).

6.2.1 Problem formulation

The routing and advance reservation planning problem in a Grid burst routing network is defined as follows. We are given a network with links l of known propagation delays d_l and capacity C_l, and a source node S that requests to send a burst i of size equal to I_i bits and duration $B_i=I_i/C_l$ to some destination node E. We are also given the utilization profiles of all links l. We want to find a feasible path over which the burst should be routed and the time at which the burst should start transmission (this can be viewed as the plan of the advance reservations of the intermediate links), so as to optimize some performance criterion, such as the reception time of the burst at its destination. We assume that the used OBS signalling scheme employs delayed reservations that reserve the links in advance in order to enable the transmission of the burst at the decided time.

In the centralized scenario a node of the network plays the role of the centralized bandwidth broker (BB). We have adopted an architecture similar to the one presented in [38]. We have assumed that there is a single BB in each network domain which keeps information of the whole domain including topology, routing information, bandwidth reservation, existing data flows, etc. When a new burst wants to be transmitted, a request will be sent to the centralized BB, which will execute the routing algorithm and send the outcome (path and starting time of the transition) back to the source.

On the other hand, in the distributed scenario each node separately keeps information of the whole network. When the node wants to send a burst it locally executes the algorithm, calculates the path to route and the time to start, transmits the burst at the decided time and informs the other nodes of the network for this decision.

After choosing the best available path, a one- or a two-way reservation scheme that employs delayed reservations is used to transmit the burst. If we consider a distributed architecture, contentions may occur. Specifically, if a one-way scheme is used, the burst is not guaranteed to arrive at the destination, since the utilization profile at some intermediate link may have changed by the time the BCH packet arrives at that link, in

which case the burst may be dropped. Similarly, if a two-way scheme is used, the reservation of the path may fail (in this case the burst is blocked but not dropped). This is a problem that cannot be avoided by any routing and advance reservation planning algorithm in a network that has nonzero propagation delays.

If we consider a centralized architecture the central BB has a complete knowledge of the utilization of all the links, since it has routed and scheduled all the previous bursts, and thus no contentions occur. In the centralized version there is no particular reason for employing a two-way reservation scheme and thus in order to obtain smaller delays we assume that a one-way protocol is always employed in this architecture.

6.2.2 Link Utilization Profiles, Utilization Update messages and Burst Routing algorithms

Link utilization profiles

In the optical burst switching paradigm, each node needs to keep a record of the capacity reserved on its outgoing links and wavelengths as a function of time ([29] and [34]) in order to perform channel scheduling and reservation. The utilization profiles of the links have the form presented in section 4.2.2 (Figure 4-4).

In the case of the centralized architecture the central bandwidth broker (BB) keeps track of the utilization of all links by maintaining a utilization database for all the links which is always up-to-date. In the distributed architecture, each node maintains a “picture” of the utilization of all the links. This local picture can be different among the nodes and depends on their position in the network, since we have considered a network with non-zero propagation delays. Update information (in the form of update messages) is communicated between nodes in order to synchronize the locally maintained profiles with the actual utilization.

Link Utilization Update messages

Link utilization update messages are required only in a distributed architecture. Similar to the section 6.1.2 we have assumed an “exhaustive” update strategy. Link update messages are sent at the time a link is reserved. Assuming that link l is reserved after time σ_i with respect to current time, resource x informs the distributed BB that a burst is going to pass after time σ_i with duration I_i / C_l if

$$2 \cdot d_{x,N} \leq \sigma_i + \frac{I_i}{C_l} \qquad (2)$$

where $d_{x,N}$ is the shortest path delay from x to N.

Employed burst routing algorithms

For the scope of this study we have employed two algorithms, each in a centralized and a distributed version. The algorithms presented here consider the routing and the advance reservation planning problems jointly and thus return not only the path to be followed but also the starting time after which the source should transmit the burst. In this way, apart from choosing path, the algorithm defines fixed advance reservations that have to be performed on all the links that comprise the path.

The first algorithm that we used is the **Availability Weighting heuristic multi-cost algorithm** (AW) which was extensively presented in [37]. This algorithm consists of two phases. In the first phase it computes a set of candidate non-dominated paths for the given source-destination pair. For the purpose of the multi-cost algorithm each link is assigned a vector of cost parameters (including a discretised capacity utilization profile) and thus by defining appropriate "addition" operations for the parameters of the link's cost vector we can calculate the cost vector of a path. In order to compute the set of non-dominated paths for a given burst and source-destination pair the multi-cost algorithm also requires the definition of a "domination" operation between two paths. More specifically, we assume that a path p_1 dominates another path p_2 for a given burst if all the parameters in the cost vector of p_1 are better than the corresponding parameters of p_2. In the second phase of the algorithm we apply an optimization function to the cost vectors of the computed paths (that constitute the set of non-dominated paths). The optimization function takes into account the burst parameters and QoS requirements in order to select an optimum path. More specifically we consider as the optimum path the one that result in the minimum reception time of the data at the destination. The transmission starting time is the earliest time that the links of that path are available.

The second algorithm is the **Dijkstra shortest path algorithm with contention avoidance (D/CA)** that was also presented in [37]. In this algorithm the shortest paths of all source-destination pairs are computed at the beginning of the simulation. Upon a burst transmission request the source/ingress node combines the utilization profiles of the links on the shortest path and plans the transmission of the burst so as to avoid contention at subsequent nodes. The transmission starting time is the earliest time that the links of the shortest path are available.

In the case of the distributed architecture upon a transmission request the source node executes the algorithm (that takes into account whether a one- or a two-way reservation scheme is used) and plans the transmission of the burst accordingly. Upon completion of the algorithm, a SETUP packet is forwarded on the chosen path to actually perform the advance reservation. When the SETUP phase is dropped or blocked at an intermediate node (another reservation has already been performed) a REJ packet is sent backwards to notify the source about the rejection. The source node waits for a small period of time (back-off) and handles the blocked request as a new transmission request (re-computes a path and plans the transmission of the burst accordingly). In order to reserve the appropriate resources and set up the light path we have experimented with two variations, one from each family of reservation protocols. More specifically, we have used a variation of the JET protocol (that employs REJs for burst retransmissions) and a two-way with delayed reservations which we call Wait-For-Reservation (WFR) protocol. Note that both protocols employ delayed reservations.

In the case of the centralized architecture the source forwards the transmission request to the central BB, which, contrary to the distributed approach, has a complete knowledge of the utilization of the links. Since contentions do not occur, there is no reason for employing a two-way scheme and thus to obtain a low average end-to-end delay we have used a one-way JET protocol.

To sum up, we have experimented with two routing algorithms (AW and D/CA) when these are implemented either as centralized or as distributed. In the distributed approach we have also examined the performance of AW and D/CA algorithms when a JET or a WFR protocol is used.

6.2.3 Performance results

In order to examine the efficiency of a distributed or a centralized architecture for the advance reservation in an Optical Burst Switched network we have assessed the performance of the algorithms presented in section 6.2.2 in a centralized and a distributed Grid environment. We have extended the ns-2 platform and conducted full network simulation experiments. More specifically, we have designed appropriate link utilization profiles; we have developed ns-objects for the users, burst routers and bandwidth brokers and also defined a new ns-2 packet format for the link utilization update messages.

We obtained simulation results for a 5x5 mesh with wraparounds topology. All links were assumed to be bi-directional with propagation delays proportional to their lengths (5μsec/Km). The BCH packet processing delay was set to 0.02 μsec and the link bandwidths *C* was set equal to 1 Gb/s. The neighboring nodes were placed at a distance of 50 km, 100 km and 200 km from each other.

At each edge node, bursts transmission requests arrive according to a Poisson process with rate λ requests per second and their destinations are uniformly distributed over all remaining nodes of the network. The burst sizes were assumed to follow the exponential distribution with mean value $\overline{I}$ bits.

For the experiments in this section, unless explicitly stated, the adjacent node distance was 100km, the mean burst size $\overline{I}$ was set equal to 300kB, while the arrival rate λ of the bursts varied between 5 bursts/sec and 125 bursts/sec per source node.

We have used the following performance metrics:

- Average end-to-end delay. The end-to-end delay of a burst is defined as the time between the burst generation and the time that it has been completely transferred to its destination,

- Average number of retrials. This metric can be calculated only in distributed architectures and corresponds to the average number of retrials for a successful reservation. A retrial occurs when the burst transmission request is blocked (JET) or dropped (WFR) in the core.

- Average number of exchanged messages, which measures the communication overhead of the employed distributed algorithms. In the centralized version the average number of exchanged messages is always 2 messages per transmission request (question to/reply from the central BB). On the other hand, in a distributed scheduling architecture there is a considerable control plane overhead to "synchronize" the distributed BB (exchange utilization update messages as presented in section 6.2.2).

6.2.3.1 *Effect of arrival rate*

Figure 6-7 (a) illustrates the average-end-to-end delay experienced by the bursts. We observe that the multi-cost heuristic algorithm (AW) outperforms the D/CA algorithms in all examined cases. Regarding the architecture and the signaling protocol, the distributed JET case exhibits the best performance, followed by the

centralized JET case, while the worst performance was observed for the distributed WFR architecture. By comparing the results presented in Figure 6-7 (a) to those presented in Figure 6-6(a) and Figure 6-4 (a) we can observe the difference between the scales of the Burst routing and task scheduling problems. Specifically, in the task scheduling problem the execution time dominates the total task delay, which takes values in the order of sec, while in the burst routing problem the transmission and propagation delays are comparable and thus the network length plays a more significant role. It is worth noting that in the burst routing problem that we have examined the average end-to-end delay takes values in the order of msec.

Figure 6-7 (b) shows the average number of SETUP retrials only for the distributed architecture (since there are no contentions in the centralized versions of the algorithms). We observe that the WFR protocol results in slightly fewer retrials, while the chosen routing algorithm (AW or D/CA) affects slightly this metric. Finally, Figure 6-7 (c) shows the average number of messages per transmission request required for the reservation updates in the network (again only for the distributed scenario). We observe that the use of the WFR protocol results in the exchange of more messages, since the WFR protocol requests the reservation of a link a RTT (round trip time) after the setup packet has reached a node, enabling the communication of this reservation information to more distant nodes (according to Eq. 2).

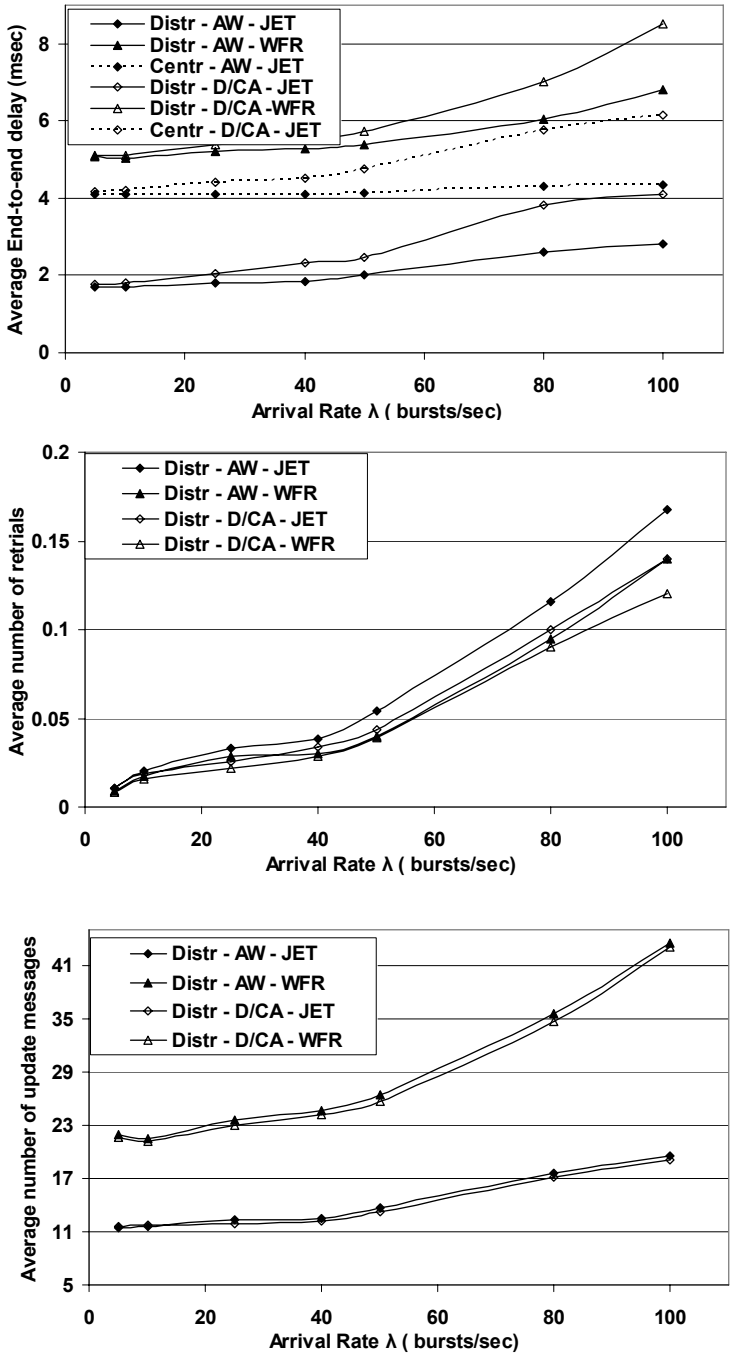

Figure 6-7: Results for the 5x5 wraparound mesh topology (a) average end-to-end delay per burst, (b) average number of retrials for a successful transmission (c) average number of exchanged messages.

6.2.3.2 *Effect of propagation delay*

The propagation delays of the network play a significant role on the link state information exchange mechanism (employed in the distributed architecture). Cleary, information regarding a reservation at a link cannot be used by a distributed BB if it reaches that BB after the time that it is useful (Eq. 2). On the other hand, the increase in the propagation delay has also negative effect in the centralized architecture, since it increases the delay to question and obtain an answer from the central BB. Thus we expect the increase in the network dimension to have a negative effect in both centralized and distributed architectures.

Figure 6-8 shows the performance of the better performing AW algorithm, in its centralized and distributed JET versions, when the adjacent node distances were set to 50, 100 or 200 km.

As expected, the average end-to-end performance of the algorithms deteriorate as the network propagation delays increase, in both centralized and distributed versions (Figure 6-8 (a)). The distributed version of the AW algorithm remains always better, from the corresponding centralized version in the same network. Figure 6-8 (b) shows the average number of update messages per request. Here we have the opposite performance and observe that as the network dimension increases fewer update messages are exchanged (something that can be explained by Eq. 2).

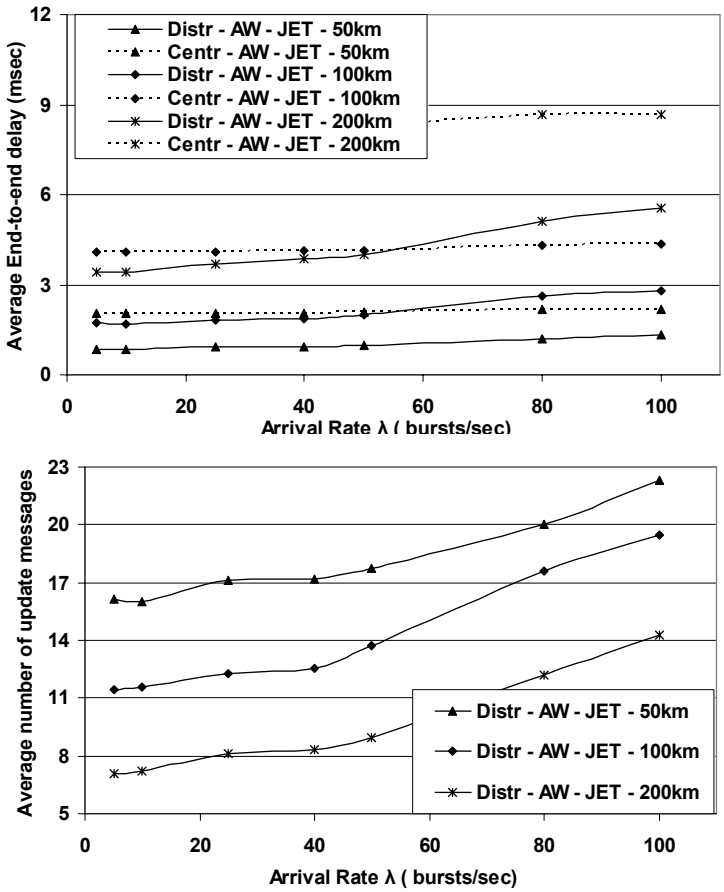

Figure 6-8: Effect of propagation delay: (a) average end-to-end delay per burst, (b) average number of exchanged messages (only in distributed algorithm) when the distance of adjacent nodes is 50, 100 and 200 km.

6.3 Malleable Advance Reservations

In the following section the reservation type MAR (Malleable Advance Reservations) is evaluated. After introducing MARs, the solution space and two strategies are presented. The strategies are evaluated each other by means of simulation. The section concludes with a note on requirements and technical details. The contents of this section are based on a paper presented at the IEEE LCN 2007 conference [61].

6.3.1 Definition and Complexity of Malleable Advance Reservations

When focusing on a file transfer service, the set of constraints *C* is reduced to a single parameter describing the file size, i.e. requests are given by r = (*s*, *d*, $t_{arrival}$, $t_{release}$, $t_{deadline}$, *v*), where *v* specifies the amount of data to be transferred; the actual time for the file transfer may be one or more sub-intervals of [$t_{release}$, $t_{deadline}$[with a corresponding transfer rate, which may change from interval to interval. Figure 6-9 shows the life cycle of a data transfer following the notion of section 3.3 with a duration of $\Delta t = t_{end} - t_{start}$.

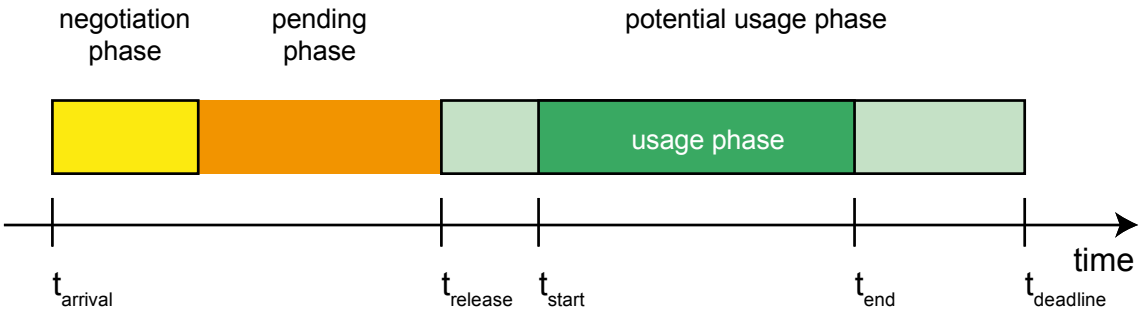

Figure 6-9: Life Cycle of an Advance Reservation

In general, the release time and the deadline need not be tight, e.g. there might be more than one possible start time. This leaves room for the reservation system to efficiently plan the data transmission in different time slots with varying data rates. In this case the reservation system may require a feedback mechanism with the application transmitting the data. Alternatively, the user or the application may define additional requirements, e.g. first-fit, best-fit, multi-path, single-path, maximum bandwidth.

6.3.2 Relaxation of Constraints

Using the concept of timeslot-based resource allocation management, the timeline $\mathfrak{T}$ is divided into a sequence of timeslots $T_i = [t_i, t_{i+1}[, 0 \leq i \leq n-1$, where $[t_0, t_n[$ represents the timeline and $T_i \in \mathfrak{T}$ In doing this, the reservation system manages utilization information individually for each timeslot, i.e. only accumulated values for every link are stored. Modelling the network topology as a capacitated, directed graph $G = (V, E, c_{max}), c_{max}: E \to R^+$, the residual capacities of a link in a given timeslot are represented by$c_{res}: T \times$

$E \rightarrow R_0^+$. When a new request is accepted by the reservation system, the residual capacities are updated. Here, we assume timeslots of dynamic length with a fixed granularity as presented in section 5.2.

6.3.3 Online Admission Control

The online admission control of the reservation system has to check whether reservation requests *r* can be realized by the residual capacities in the network given by c_{res}. Starting form dynamic timeslots with a fixed slotted granularity, a set of procedures is needed to handle requests. If a new request is processed, the reservation system has to determine all timeslots, which are overlapping with the request. Subsequently, the residual capacities can be inspected on all links within these timeslots and the admission control is able to decide on the request. The result of the admission control is a set of reservation entities R is subset of $\mathcal{T} \times P_{s,d} \times R^+$, i.e. each reservation entity is determined by a timeslot $\in \mathcal{T}$, a path from the source to the destination ($\in P_{s,d}$), and a capacity value ($\in R^+$). Note that there may be more than one reservation entity per request. The reservation entities must be stored in the reservation system to allocate and release the resources at the specified points in time. Regarding the set of reservation entities the processing of file transfer requests span three interdependent domains:

- **Spatial Domain:** Which path(s) should be used to accept the request?
- **Temporal Domain:** Which time interval(s) should be used to accept the request?
- **Capacity Domain:** How much capacity should be used?

For a file transfer service supporting deadlines, the admission control has to find a set of reservation entities within these domains such that the sum of all capacity-duration products of the corresponding reservation entities equal the data amount specified by the request. Taking a look at the spatial domains, two possible strategies to process a request can be identified:

(i) Single-Path: Only one path from the source to the sink is used to transfer data within a timeslot.

(ii) Multi-Path: A set of paths from the source to the sink is allowed to transfer data within a timeslot.

Adding the temporal domain to the single-path and multipath category leads to the question whether a spatial solution should be applied to all timeslots. Furthermore, adding the capacity domain the most contrary forms are the following:

(i) Constant Capacity Single-Path all Slots: The request uses the same path for the whole reservation time, i.e. the selected path is used for each timeslot. The reserved capacity between the source-destination pair is constant.

(ii) Variable Capacity Multi-Path per Slot: The request can use a different set of paths for each timeslot. In addition, the reserved capacity between the source-destination pair is variable per timeslot.

We take these most contrary forms as a basis for an evaluation, the following two questions arise: Which algorithms can be used to perform an online processing of file transfer requests of the mentioned strategies? Is the mapping of a request on multiple paths favourable with respect to the user and provider satisfaction? Both questions are handled in the following sections.

6.3.4 Algorithms for Malleable Reservations

After setting the scene by outlining the system architecture, fundamental data structures, and strategies to process file transfer requests, the basis for a comparison of the algorithms described in sections 6.3.4.1 and 6.3.4.2 is given. Basically, a set of data transfer requests *R* is handed to the reservation system in an online fashion, i.e. the system decides on a request without knowledge of upcoming requests. Intuitively, two basic metrics can be used to characterize the performance of the algorithms which correspond to two different optimization goals:

- User Satisfaction: Assuming that the user satisfaction is independent of the file size, accept as many requests as possible.

- Network Provider Satisfaction: Maximize the profit, i.e. in a simple setting the file size describes the profit for the provider and the sum of the sizes of accepted files should be maximized.

The corresponding metrics are entitled request blocking ratio (RBR) and bandwidth blocking ratio (BBR). In the presented context of requests spanning a time period with a certain capacity, volume blocking ratio (VBR) would also be appropriate in the latter case.

The RBR is defined as $|\bar{R}|/|R|$, where $|\bar{R}|$denotes the cardinality of the set of rejected and $|R|$ the cardinality of the set of all requests presented to the reservation system. The bandwidth or volume blocking ratio can be defined in a similar fashion [56]. Obviously, the values of RBR can range from 0 to 1, where a smaller value represents a better performance.

In this evaluation a basic job model is used in order to focus on the differences of path selection algorithms and differences of the single- and multi-path strategies. The requests generated in the simulation framework are restricted to a single file size, i.e. except of a constant factor the RBR is equivalent to the BBR. As a consequence, the performance of the algorithms is solely measured by the RBR. The generated requests are uniformly distributed between all source-destination pairs. We assume independent requests and model the inter-arrival time of the requests as being exponentially distributed.

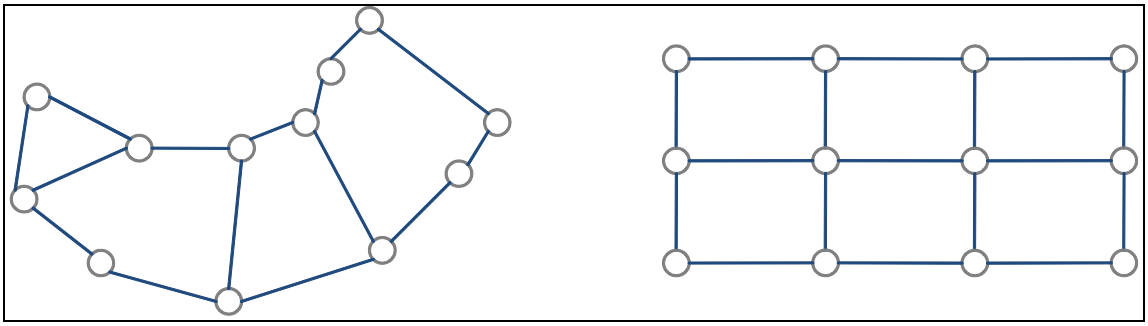

Figure 6-10: Topology of the Abilene Backbone (left) and Grid graph (3, 4) (right)

In addition to the job model, topologies are an important parameter. In Figure 6-10 two of the three topologies used in the simulative studies are shown. The graph on the left side represents the Abilene core topology, which is a cross country backbone in the USA. On the right a Grid graph (or lattice) with 12 vertices is shown. Furthermore, a complete graph (or fully connected graph) with 6 vertices is used for the simulative evaluation of our algorithms. The multi-path strategy has advantages in a richer connected topology as different paths can be used for the data transfer at the same time. In order to examine this property, topologies with an increasing nodal degree are chosen (2–3 for Abilene, 2–4 for the Grid graph, and 6 for the complete graph).

6.3.4.1 *Algorithms for the Single-Path Strategy*

In this section we concentrate on a single-path strategy, i.e. only one request is processed at a time and only one path is used in the network at a time to convey data. Therefore, the first set of algorithms maps a request r to a single path. After presenting the algorithm, an evaluation with different path selections is presented.

A Heuristic for Constant Capacity Single-Path for All Slots

This approach is derived from an algorithm presented in [8] and tries to map a file transfer request to an advance reservation given by a single path and a constant capacity. It is a heuristic and avoids the super-polynomial runtime of the variable capacity approach (cf. advance cumulative reservations [15]). Although Burchard describes a similar algorithm in [8] that maps requests according to the mentioned heuristic, details of the algorithm are different (e.g. we are using timeslots of dynamic length). Therefore, our algorithm is described in detail. The overall approach is to start the search for configurations in the temporal domain. The algorithm maps a request to a contiguous capacity block in the network resulting in a single reservation entity. It includes the following phases:

1) Determine potential time intervals for reservation entities (potential configurations),

2) compute a path for each potential configuration, and

3) select a configuration that meets the demand otherwise reject the request.

In the first phase potential configurations are determined by timeslots of already accepted reservations, which cover the release time $t_{release}$ and the deadline $t_{deadline}$ of the processed request. Given k reservations, this leads

to $\sum_{2}^{2k+2} i \in O(k^2)$ configurations at most. Furthermore, the maximum capacity given by the widest path between the source *s* and destination *d* according to c_{max} is computed to restrict the set of potential configurations. Taking a look at Figure 6-11, the potential configurations starting at time $t_r = t_1$ are presented at the right side. The corresponding timeslots are taken from a set of example reservations presented on the left side of the figure. Two default configurations are added based on the maximum capacity with respect to the widest path (cf. $[t_1, t_{min}[$ in Figure 6-11) and the minimum capacity regarding the release time and deadline (cf. $[t_1, t_d[$).

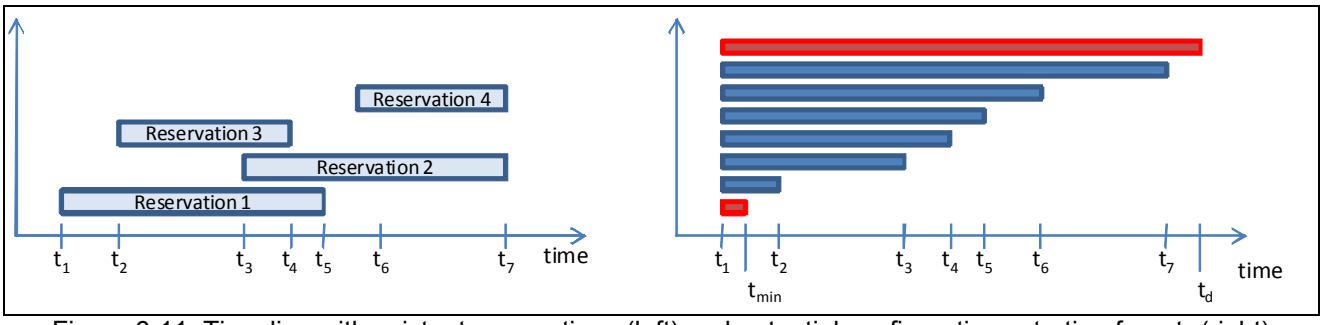

Figure 6-11: Time line with existent reservations (left) and potential configurations starting from t_1 (right)

In the second phase, these configurations are analyzed by a path selection algorithm on edges that meet the given capacity constraint on all timeslots from the start to the end. As the capacity constraint is a link constraint in the sense of QoS routing a pre-processing time of $O(k|E|)$ is needed, where $|E|$ is the number of edges in the graph and k the number of accepted requests. Subsequently, a path can be computed on the pre-processed graph. Here, a basic Dijkstra algorithm with a running time of $O(|V|^2)$ is used to find a path. Different path selection strategies, which can be implemented by the Dijkstra algorithm, are discussed in the next section.

In the third phase, three tie-breaking rules are used to select a configuration. The following rules are processed with decreasing priority: (i) prefer the configuration according to the path selection metric (e.g. the configuration with the overall shortest path is selected), (ii) prefer the longest duration, and (iii) select a configuration at random. In other words, the presented algorithm processes a single request by aligning it to the already accepted requests if possible.

Experiments on the Single-Path for All Slots Strategy

The second phase of the previously described heuristic can utilize a variety of path selection algorithms. These algorithms determine the network resources for the potential configurations. In order to identify a favourable path selection algorithm, we take a look on the performance of the following algorithms:

1) Select the shortest path (measured in hop count).

2) Select the widest path with respect to the residual capacities.

3) Select the shortest distance given by the sum of the reciprocal of the residual capacities.

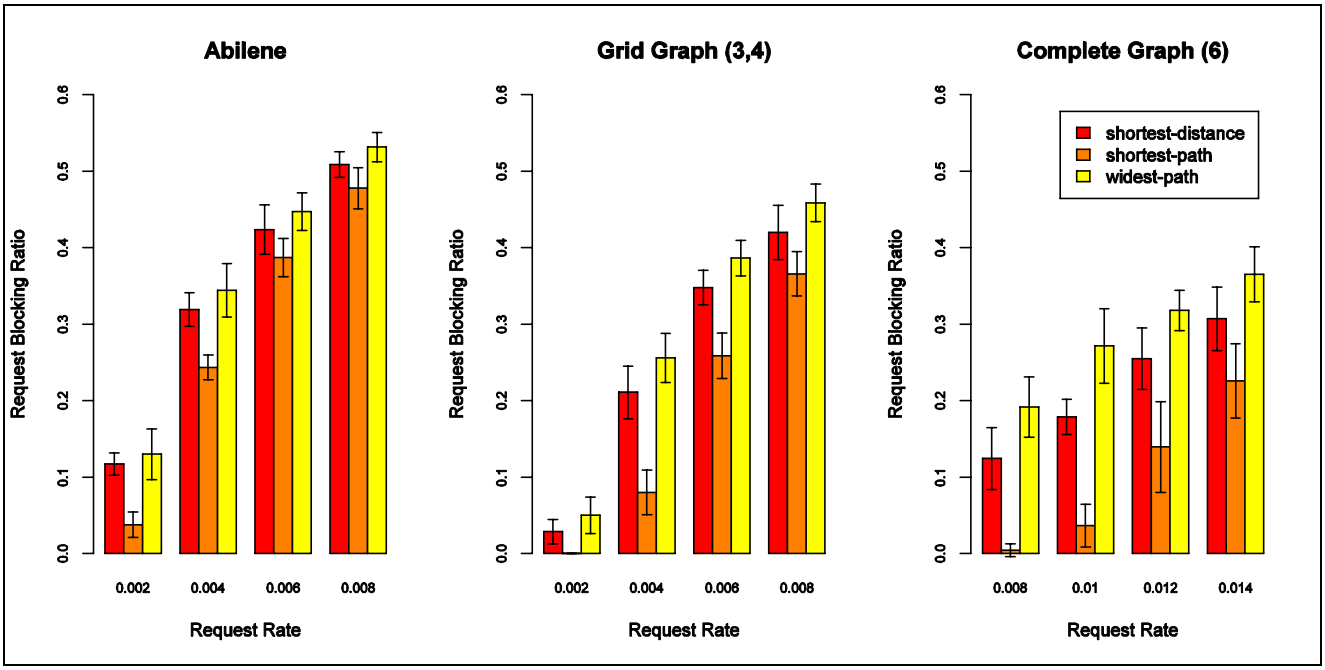

Figure 6-12: Simulation Results for the Single-Path for All Slots Algorithm

Figure 6-12 shows the RBR (with a 0.95 confidence interval) as a function of the request rate. The reciprocal value of the request rate specifies the mean inter-arrival time. The initial link capacity is identical on all links of the Abilene topology, the grid graph, and the complete graph. The generated requests are uniformly distributed between all source-destination pairs and a single value for the data size is used and a duration of approximately 4 times the minimal transmission time. The minimal transmission time is given by the widest path between the source and the destination. As a result, 4 requests share the capacity of a link in a timeslot at most. Overall, the RBR is decreasing from the Abilene topology to the complete graph as the capacity of the graphs $\left(\sum_{e \in E} c_{max}(e)\right)$ is increasing and the structures provide more paths between arbitrary pairs of vertices.

The aggregated simulation runs presented in Figure 6-12 show that the shortest path algorithm performs best. The widest path and shortest distance algorithm tend to allocate longer paths and therefore use more resources. The usage of longer paths blocks links needed for subsequent requests. Increasing the network load results in lower performance differences between the path selection algorithms. The shortest distance algorithm performs slightly better than the widest path algorithm in our scenarios. As the usage of the shortest path algorithm is the most suitable in the presented simulation scenarios, this path selection algorithm is preferred for comparing the single-path algorithm against the multi-path algorithm.

Further Single-Path Strategies

Another single-path strategy is to allow for reservation entities changing the path on a per timeslot basis. One approach for this strategy is described in [8]. The advantage of a single-path per slot strategy is depicted in Figure 6-13. In the subsequent timeslot denoted by $[t_0, t_1[$ two commodities are present. The next timeslot $[t_1, t_2[$ has three commodities. If the path for the source-destination pair (s_1, d_1) can be switched between the timeslots – for example by using path switching capabilities of MPLS nodes – additional alternatives can be regarded in order to lower the RBR.

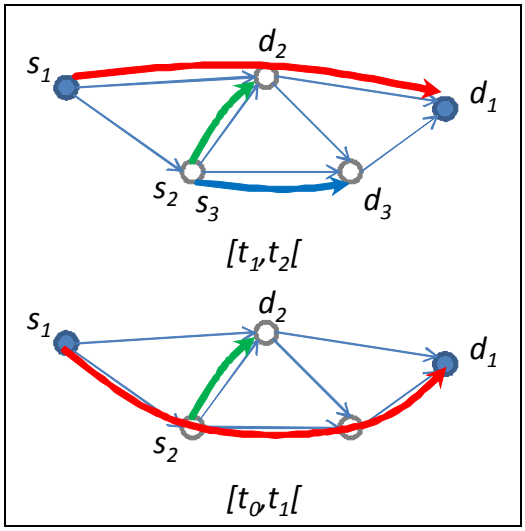

Figure 6-13: Switching a single path between timeslots

6.3.4.2 *Algorithms for the Multi-Path Strategy*

In this section the processing of a file transfer request with a multi-path strategy is presented. The algorithm maps a request to a network flow and chooses paths from this flow. The section concludes with an evaluation of different path selection strategies.

Maximum Flow Approach

The main idea of the multi-path approach is to construct a time expanded graph representing the scheduled load in the network. This graph covers the timeslots of the new request. The possible reservation entities are limited by the maximum flow from the requested source to the destination in this graph. Starting from this idea, the admission decision of a multipath request is split into three phases:

1) Create a time expanded graph representing the available capacity during the requested time period,

2) compute the maximum flow in the time expanded graph, and

3) select a set of paths from the network flow meeting the demand otherwise reject the request.

In the first phase, a time expanded graph is constructed as basis for a flow computation. Briefly, the time expanded graph contains a copy of the basic graph for each timeslot considered. The edges of each copy are annotated with the residual capacity of the corresponding timeslot. Note that each copy represents the network connectivity in terms of residual capacities in a specific timeslot. A path traversing one of the copies can be interpreted as an amount of data that can be transferred in this time interval. An example of a time expanded graph for a request $r = (s, d, t_{arrival}, t_{release}, t_{deadline}, v)$ is shown in Figure 6-14. Let $[t_0, t_1[$, $[t_1, t_2[$, and $[t_2, t_3[$ with $t_0 = t_{release}$ and $t_3 = t_{deadline}$ be the timeslots, which are candidates for data transmission of r. A virtual source s' and a virtual destination d' are defined, which connect the source and destination nodes in the different timeslots s_i, d_i

respectively. The capacity of the edges from the two virtual nodes to each copy has to be selected in accordance to the duration of the timeslot to configure the maximum amount of data that should be transmitted in the timeslot. The size of the time expanded graph depends on the number of timeslots covered by the request, i.e. up to *2k + 1* copies of the graph can be present. In the following we will refer to the indexed set of copies as layers.

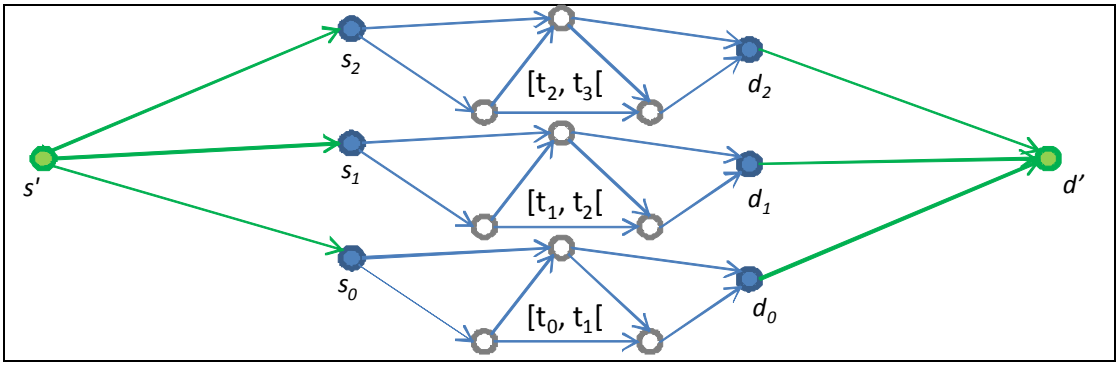

Figure 6-14: Time-Expanded Graph to solve the Maximum Flow Problem

In the second phase the maximum flow from the virtual source *s'* to the virtual destination *d'* is computed. This can be done by a maximum flow algorithm [59]. The Edmonds-Karp algorithm, which is based on the Ford-Fulkerson method, has been used. If the flow value – which relates to a data amount – is larger or equals the requested amount of data, a potential solution to the reservation request has been found. If the flow value is less than the data amount to be transferred, the request is rejected. In the case of equality a decomposition of the flow into paths is a valid solution. If the maximum flow is larger, a sub-flow with a flow value equal to the requested data amount has to be determined.

In the third phase, the flow is decomposed into a set of paths. In order to decompose the flow, different approaches can be used. Here, paths are repeatedly selected from the maximum flow and the flow graph is updated by subtracting the data amount from the corresponding edges. We use the Dijkstra algorithm to perform the decomposition into a set of paths. If the flow value is larger than the demand, a heuristic chooses a subset of all available paths from the maximum flow. This subset specifies the reservation entities. Note that paths and flows in certain layers of the time-expanded graph are associated with the duration of the associated timeslot. These paths and flows can be interpreted as a certain amount of data.

Experiments on the Multi-Path per Slot Strategy

The maximum flow value computed in the second phase of the presented algorithm can identify a larger data amount than requested. In this section two families of heuristics are presented to choose a subset of the decomposed maximum flow. One family follows the shortest path approach, i.e. the heuristic always chooses the shortest path (measured in hop count) from the set of available paths, until the set of chosen paths meets the requested data amount. The other family follows the widest path approach, i.e. repeatedly choose the path which corresponds to the largest data amount transferable. This is done with the intention to minimize the total number of paths and therefore minimize the resource configuration effort.

1) **Shortest Path**: This approach always chooses the shortest path in the set of the residual available paths. If two or more paths have equal length, the one found first is taken.

2) **Largest Residual among Shortest**: Again, this approach is based on the shortest path strategy. If two or more paths with equal length are present, this strategy considers the capacity instead of the data amount. Remember that in the context of the time expanded graph the flow refers to the amount of data that can be transferred. This strategy chooses a path which has the largest residual capacity on its bottleneck link in the given timeslot.

3) **Widest Path**: This approach always chooses the path corresponding to the largest data amount. This leads to a low number of paths to meet the data amount specified by the request.

Figure 6-15 shows the results of the simulative evaluation of the different path selection strategies. Again, the RBR (with a 0.95 confidence interval) is given as a function of the request rate. The simulation parameters are used in accordance to the previous sections. Although the duration interval is approximately 4 times the minimal transmission time with respect to the widest path between the source and the destination, more than 4 requests can share the capacity of a link in a timeslot. This is due to the fact that the demand can be met by multiple paths. The RBR is decreasing from the Abilene topology to the complete graph because the capacity of the graphs increases.

In all three scenarios the strategies based on the shortest path approach generally perform better than the widest path approach. These simulation results are in accordance with the findings of [56]: QoS routing strategies using paths with fewer hops generally perform better than those neglecting the path length. In all aggregated simulation runs presented in Figure 6-15 the largest residual among shortest strategy performs best. The algorithm results in good performance, because the network load is distributed in a way that avoids the blocking of links. This increases the chance of successful admission of subsequent requests. The widest path strategy suffers from its greedy behaviour. It takes the widest available paths, which results in long paths, and therefore in unnecessary resource usage which lowers the chance of successful admission of subsequent requests. Due to the results of these simulations, the largest residual among shortest strategy is used for the comparison in section 6.3.4.3.

Further Variations on the Multi-Path Strategy

Additional strategies for multi-path reservations are under consideration. A variant is to avoid gaps in reservations by using at least one path per timeslot. This allows for transmitting data continuously and making multi-path transmissions more apt for protocols, which cannot handle transmission interruptions. An alternative approach is to use the same or at least a limited number of paths per timeslot. This might be important if the client handles multi-path transfers by using a dedicated TCP or UDP connection per path.

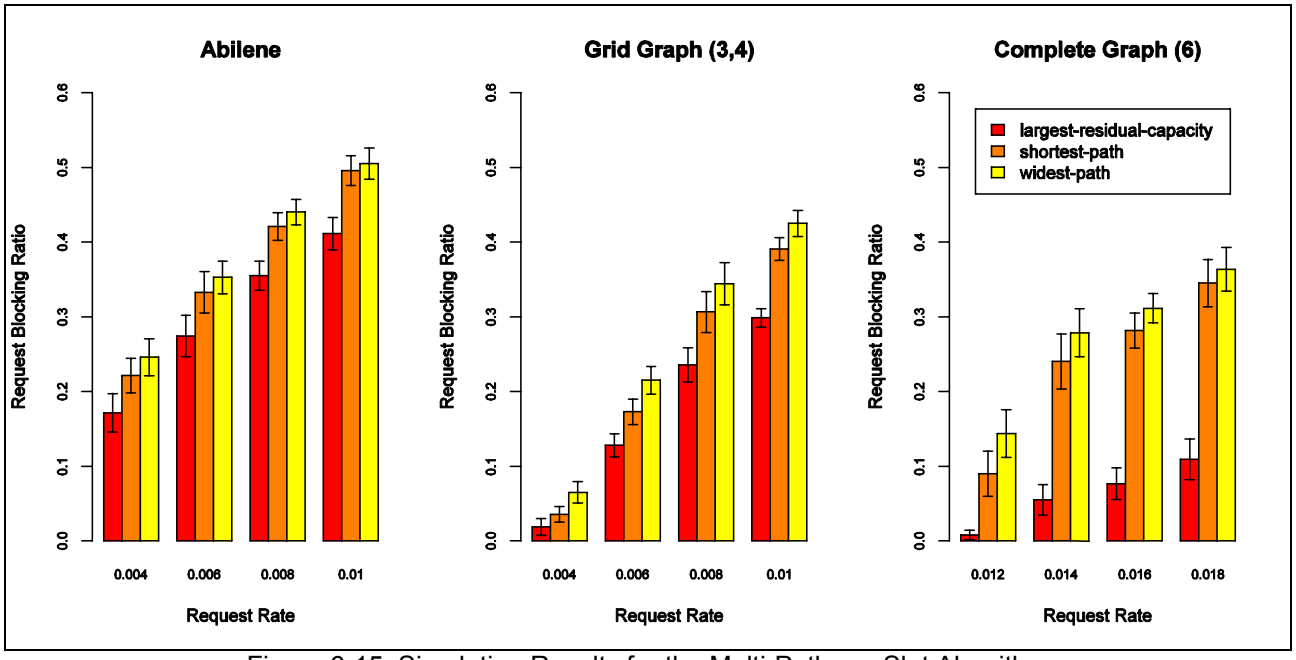

Figure 6-15: Simulation Results for the Multi-Path per Slot Algorithm

6.3.4.3 *Comparison of the Single- and Multi-Path Algorithms*

Figure 6-16 shows simulation results for the single- and multipath algorithms. Again, the RBR (with a 0.95 confidence interval) is given as a function of the request rate. The set of request rates is chosen with respect to the simulation results presented in section 6.3.4.1 and 6.3.4.2. Other simulation parameters are used in accordance to the previous evaluations.

Overall, the RBR of the multi-path algorithm is lower compared to the single-path algorithm. It is to mention that the benefit of the multi-path algorithm in Figure 6-16 increases regarding the different topologies from Abilene to the complete graph. This is due to the fact that more alternative paths exist between arbitrary source-destination pairs. The minimum link capacity that can be used by the single-path algorithm is determined by the maximum duration and the specified data amount. Links providing a lower residual capacity can be used by the multi-path strategy: The demand is distributed to several paths. Therefore, the most significant benefit of the multi-path approach is achieved in the complete graph scenarios.

Although the multi-path approach performs best, a drawback of this approach is obvious: The network, transport, and application layer have to cooperate to facilitate a transfer on multiple paths. In a circuit-switched network flows can be partitioned at the ingress and mapped onto different paths with corresponding capacities, e.g. by protocol header information. Beside the striping approach used by GridFTP, the transport layer protocol SCTP in conjunction with its multi-streaming and multi-homing feature could be used to distribute data across multiple end-to-end paths [60].

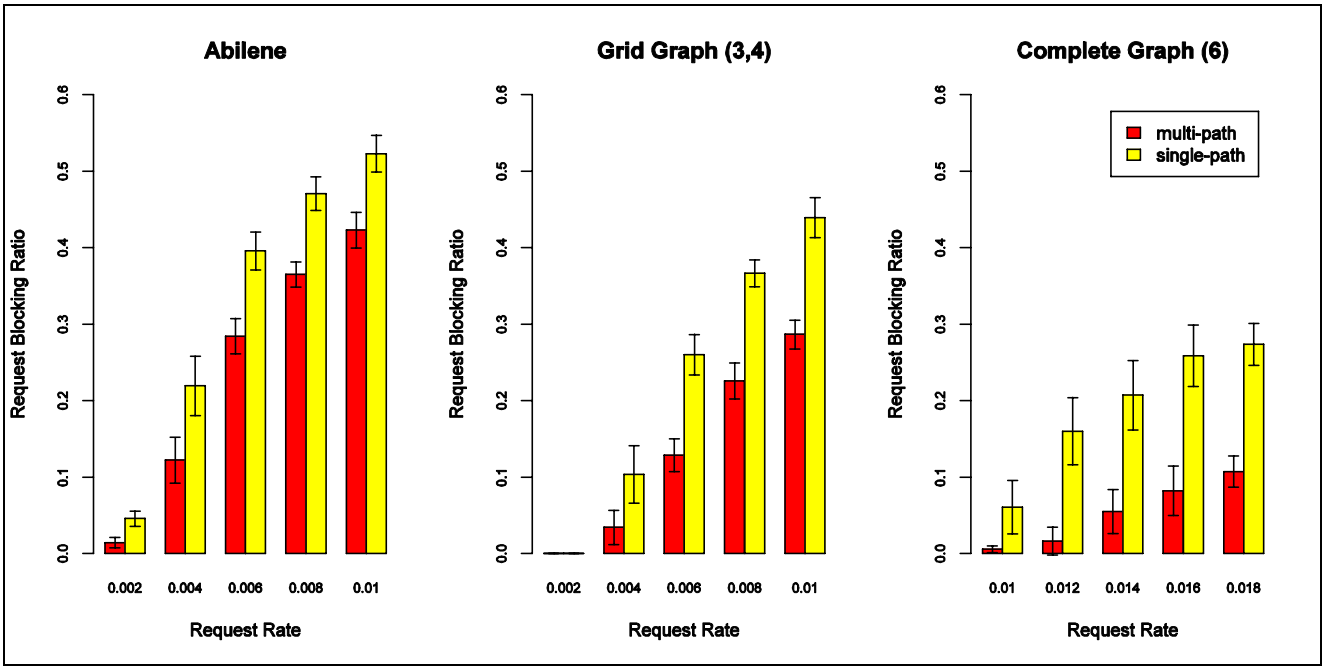

Figure 6-16: Comparison of the Single-Path for All Slots and Multi-Path per Slot Algorithms

6.3.5 Conclusion

In Sections 6.1 and 6.2 we examined, by means of simulations, the distributed and centralized advance reservations of computational and network resources. In the centralized version of the problems we assumed that the central scheduler (task-scheduler or bandwidth broker respectively) has a complete knowledge of the (computational or network) resources' utilization and thus the scheduling or routing decisions are always correct. In the distributed version we assumed that each distributed scheduler maintains a "picture" of the utilization of all the resources, which is updated by exchanging corresponding messages. We used an "exhaustive" update strategy in which reservation information is forwarded to all distributed schedulers that can use this information. In the distributed architecture there are cases that task scheduling or routing decisions are performed with outdated utilization information, and thus the advance reservations decided cannot be performed.

With respect to the computational resources, we observed that the distributed versions of the scheduling algorithms examined exhibit smaller average total delay when compared to the centralized versions. The increase in the network dimension affects the centralized architecture, since it increases the time required to communicate with the central meta-scheduler. Although the increase of the network dimension affects negatively the exchange of the utilization information, the exhaustive update strategy that we used enabled the communication of the update information to all distributed schedulers. Thus, the increase of the network dimension results in a smaller deterioration of the distributed algorithms' performance, when compared to the deterioration of the centralized algorithms. With respect to the communication resources, the conclusions are similar. We observed that the distributed version of the one-way reservation schemes performed better than the

centralized one-way and the distributed two-way reservation schemes. In the scenarios that we examined for the routing problem, the transmission and propagation delays are comparable with the reservation duration and thus the network dimension plays a more significant role than in the scheduling scenarios. The performance of the algorithms deteriorates as the network dimension increases, in both centralized and distributed versions. The distributed versions of the algorithms are always better than the corresponding centralized versions, under the exhaustive update strategy used.

Thus, the simulation results indicate that a distributed architecture can yield a better average end-to-end delay performance when compared to a centralized architecture, when an exhaustive update strategy is used. The tradeoff of the large number of update utilization messages that has to be exchanged between the distributed schedulers is not prohibitive, since modern Grid networks exchange a large number of messages for monitoring and other purposes and thus utilization information can be incorporated in them.

In Section 6.3 a flexible reservation mechanism for network resources for demanding, distributed Grid applications was introduced with a special focus on a file transfer service. This file transfer service guarantees the timely availability of data at the different Grid sites even for very large files. To cope with these requirements, an architecture for advance reservations of network resources and different strategies and algorithms for a flexible network reservation service were defined. These algorithms can be classified in two fundamentally different strategies. The first one uses a single path per file transfer request while the second class of strategies splits the data transfer for a request onto several distinct paths at the same time. For the single path strategy we described a heuristic realizing the most rigid approach, i.e. using a constant capacity during the whole transmission without interruption. The multipath approach is based on the decomposition of the maximum flow between the source and the destination into a set of paths and in general uses different data rates during the transmission. Our simulation results show that data transport for a single file via multiple paths can lead to a higher user and network provider satisfaction than single path transfers. The results were strongly dependent on the structure of the underlying network topology and suggest that with a higher nodal degree the advantage of the multi-path strategy becomes larger. The most significant gain was achieved in a fully connected topology.

7 Conclusions

An objective of the Phosphorus project is to address a set research topics to enable on-demand end-to-end network services for the Grid community, i.e. network infrastructures can be used to connect various high-end resources like computational and storage resources as well as visualization entities and scientific instruments.

The concept of advance reservations is used in the Grid context to orchestrate a set of resources and services. The orchestration allows for implementation of efficient workflows in the Grid. In order to identify scenarios and use cases – where advance reservations in the context of networking are beneficial – chapter 2 explores a set of common and Phosphorus-specific application environments. As described, inter-connections to stream or exchange information at high data rates are identified in workflows of large scale Grid application. Additionally, various scenarios describe the need to transfer data for computational tasks. The process of pre- and post-staging jobs has slightly different requirements: The data to be processed has to be delivered timely, but neither the end-to-end delay nor the used capacity is crucial.

Based on these observations, chapter 3 categorizes and specifies types of advance reservations with a focus on networking. These models of advance reservations are embedded in a broader context in chapter 4. Here, the interplay of advance reservations including computational and network resources is regarded. The chapter presents centralized and distributed approaches to integrate advance reservations of various resources.

In the following chapters 5 and 6 detailed studies on particular topics are presented. In chapter 5 the aspect of resource management for advance reservations is addressed. The main objective is a detailed study of the impact of granularity on the responsiveness and efficiency with a focus on network resource management. A dynamic timeslot model and granularity concept is introduced to decouple the number of timeslots needed from the number of accepted reservations.

The contribution of chapter 6 is twofold. On the one hand, a comparison of distributed versus centralized advance reservation architectures is presented by means of simulation. The simulation results indicate that a distributed architecture can yield a better average end-to-end delay performance when compared to a centralized architecture, when an exhaustive update strategy is used. The exhaustive update strategy comes with the trade-off of a large number of update messages to be exchanged. On the other hand, different

strategies and algorithms for malleable advance reservations (MARs) are regarded. These algorithms can be classified in two fundamentally different strategies. The first one uses a single path per transfer request, while the second class of strategies splits the data transfer for a request onto several distinct paths at the same time. Simulation results show that data transport via multiple paths leads to a higher user and network provider satisfaction than single path transfers.

An important topic is to incorporate the gained results in developments performed by other work packages in the Phosphorus project. This allows for a validation and demonstration of applications, which access new and improved services via the Grid middleware in the Phosphorus testbed. At the time of writing, developments in WP1 are considered as good candidates for utilizing the gained results described in this document. Depending on architectural details, the resource management for advance reservations presented in chapters 5 can be used as a basis for NRPSs (e.g. ARGON, DRAC, and UCLP), NRPS adapters or the NSP (Network Service Plane). Furthermore, implementing malleable advance reservations in various NRPSs and the NSP would provide a unique service for Phosphorus' applications.

8 References

[1] Grid High-Performance Networking RG (GHPN-RG), https://forge.gridforum.org/projects/ghpn-rg, last accessed 19.09.2007

[2] Open Grid Forum, http://www.ogf.org/, last accessed 19.09.2007

[3] Tiziana Ferrari, “Grid Network Services Use Cases OGF GHPN”, version 2.10, 2006

[4] EGEE - Enabling Grids for E-sciencE, http://www.eu-egee.org/, last accessed 20.9.2007

[5] Recommendation ITU-T G.114, One-Way Transmission Time, Int'l Telecommunication Union, Geneva, 1996

[6] Phosphorus Deliverable D.3.1, “Use cases, requirements and design of changes and extensions of the applications and middleware”, Thomas Eickermann, Wolfgang Ziegler (editors), 2007

[7] Joseph Y-T. Leung (editor), “Handbook of Scheduling: Algorithms, Models, and Performance Analysis”, Chapman & Hall / CRC Computer and Information Science Series, 2004

[8] L.-O. Burchard, “Advance Reservations of Bandwidth in Computer Networks”, PhD Thesis, Technische Universität Berlin, 2004

[9] Martin Karsten, Nicole Berier, Lars Wolf, Ralf Steinmetz, “A Policy-Based Service Specification for Resource Reservation in Advance”, Proceedings of the International Conference on Computer Communications (ICCC'99), 1999

[10] Alexander Schill, Sabine Kuhn, Frank Breiter; "Resource Reservation in Advance in Heterogeneous Networks with Partial ATM Infrastructures", ACM INFOCOM, 1997

[11] Phosphorus Deliverable D.2.1, “The Grid-GMPLS Control Plane architecture”, N. Ciulli (editor), 2007

[12] Jun Zheng and Hussein T. Mouftah, “Routing and Wavelength Assignment for Advance Reservation in Wavelength-Routed WDM Optical Networks”, IEEE International Conference on Communications (ICC), 2002], [L.-O. Burchard, “Advance Reservations of Bandwidth in Computer Networks”, PhD Thesis, Technische Universität Berlin, 2004

[13] Alain Roy and Volker Sander, „GARA: a uniform quality of service architecture” in „Grid resource management: state of the art and future trends”, 2004, pages 377-394, Kluwer Academic Publishers

[14] Eric He, Xi Wang, Venkat Vishwanath, Jason Leigh, "AR-PIN/PDC: Flexible Advance Reservation of Intradomain and Interdomain Lightpaths," IEEE GlobeCom 2006, July 2006

[15] Roch A. Guérin, Ariel Orda, Networks With Advance Reservations: The Routing Perspective, Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2000

[16] Domenico Ferrari and Amit Gupta and Giorgio Ventre, "Distributed Advance Reservation of Real-Time Connections", Multimedia Systems, Volume 5, Issue 3, 1997

[17] Olov Schelén and Stephen Pink, "Sharing Resources through Advance Reservation Agents", Journal of High Speed Networks, Special issue on Multimedia Networking, Vol 7, No 3-4, 1998

[18] L. C. Wolf and L. Delgrossi and R. Steinmetz and S. Schaller, "Issues of Reserving Resources in Advance", LNCS 1018, 1995

[19] I. Foster and C. Kesselman and C. Lee and R. Lindell and K. Nahrstedt and A. Roy, A distributed Resource Management Architecture that Supports Advance Reservation and Co-Allocation, In Proc. of IWQOS, UK, 1999, pp. 27-36.

[20] M. Siddiqui, A. Villazón, Thomas Fahringer, "Grid allocation and reservation - Grid capacity planning with negotiation-based advance reservation for optimized QoS", ACM/IEEE SC2006 Conference on High Performance Networking and Computing, 2006

[21] Joerg Decker, Joerg Schneider, "Heuristic Scheduling of Grid Workflows Supporting Co-Allocation and Advance Reservation," *ccgrid*, pp. 335-342, Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), 2007

[22] Anthony Sulistio, Uros Cibej, Sushil Prasad, Borut Robic and Rajkumar Buyya, "GarQ: An Efficient Data Structure for Advanced Reservations in Grid Computing", Technical Report, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2007.

[23] S. McGough, A. Afzal, J. Darlington, N. Furmento, A. Mayer, and L. Young, Making the Grid Predictable through Reservations and Performance Modelling, The Computer Journal, 48(3):358--368, 2005

[24] Maui Cluster Scheduler, http://www.supercluster.org/maui/, last accessed 21.09.2007

[25] Grid Optical Burst Switched Networks, http://www.ogf.org/Public_Comment_Docs/Documents/Jan-2007/OGF_GHPN_GOBS_final.pdf

[26] C. Qiao and M. Yoo, "Optical burst switching (OBS)–a new paradigm for an optical Internet," J. of High Speed Networks, vol. 8, no. 1, pp. 69–84, 1999.

[27] J. Turner, "Terabit burst switching," J. High Speed Networks, vol. 8, pp. 3–16, 1999.

[28] E.A. Varvarigos, V. Sharma, "The ready-to-go virtual circuit protocol: a loss-free protocol for multigigabit networks using FIFO buffers", IEEE/ACM Transactions on Networking, vol.5, (no.5): pp.705-18, Oct. 1997.

[29] E.A. Varvarigos and V. Sharma, "An efficient reservation connection control protocol for gigabit networks", Computer Networks and ISDN Systems 30, pp. 1135–1156, 1998.

[30] Yang Chen, Chunming Qiao and Xiang Yu, "Optical Burst Switching (OBS): A New Area in Optical Networking Research", IEEE Network Magazine,Vol. 18 (3), pp.16 - 23, May-June 2004.

[31] M. Dueser and P. Bayvel. “Analysis of a dynamically wavelength-routed optical burst switched network architecture”, IEEE/OSA Journal of Lightwave Technology, 20:574-585, April 2002.

[32] J. Y. Wei and R. I. McFarland. Just-in-time signaling for WDM optical burst switching networks. Journal of Lightwave Technology, 18(12):2019{2037, December 2000.

[33] I. Baldine, G. N. Rouskas, H. G. Perros, and D. Stevenson. JumpStart: A just-in-time signaling architecture for WDM burst-switched networks. IEEE Communications, 40(2):82{89, February 2002.

[34] K. Manousakis, V. Sourlas, K. Christodoulopoulos, E.A. Varvarigos, K. Vlachos, “A Bandwidth monitoring mechanism: Enhancing SNMP to record Timed Resource Reservations”, Journal of Network and Systems Management (JNSM), Vol 14 No 4, pp 583-597, December 2006.

[35] Kostas Christodoulopoulos, Kyriakos Vlachos, Kostas Yiannopoulos and Emmanuel A. Varvarigos, “Relaxing Delayed Reservations: An approach for Quality of Service differentiation in Optical Burst Switching networks”, IEEE International Conference on Broadband Communications, Networks and Systems, in proceedings of BROADNETS, vol.1, pp. 1-8, Oct. 2006, San Jose, CA, USA.

[36] M. Yoo, C. Qiao, and S. Dixit, “QoS performance of optical burst switching in IP-over-WDM networks,” IEEE J. Select. Areas Commun., vol. 18, pp. 2062–2071, Oct. 2000.

[37] E.A. Varvarigos, V. Sourlas, K. Christodoulopoulos, “Routing and Scheduling in Connections in Networks that Support Advance Reservations” submitted to Journal of Computer Networks, Elsevier.

[38] Z. Zhang, Z. Duan, and Y. T. Hou, “On scalable network resource management using bandwidth brokers,” Proc. IEEE/IFIP NOMS 2002, pp. 169-183, 15-19 April 2002.

[39] H. Wen, H. Song L. Li and S Wang “Load Balancing contention resolution in OBS networks based on GMPLS”, Int. J High Performance Computing and Networking 2005, Vol. 3 No 1 pp 25-32.

[40] A. Zapata and P. Bayvel, “Dynamic wavelength-routed optical burst-switched networks: scalability analysis and comparison with static wavelength-routed optical networks”, Proceedings OFC 2003, pp 212-213.

[41] J.Teng and G.Rouskas, “Routing Path optimization in optical burst switched networks”, Proc. of ONDM 2005, p 1-10.

[42] J. Xu, C. Qiao, J. Li, and G. Xu, “Efficient channel scheduling algorithms in optical burst switched networks,” in Proceedings of INFOCOMM, 2003, vol. 3, pp. 2268–2278.

[43] E. Varvarigos, N. Doulamis, A. Doulamis, T. Varvarigou,“Timed/Advance Reservation Schemes and Scheduling Algorithms for QoS Resource Management in Grids”, Chapter 16

[44] Lu Shen, B.Ramamurthy, “Centralized vs. Distributed Connection Management Schemes under Different Traffic Patterns in Wavelength-Convertible Optical Networks” ICC 2002, pp 2712 - 2716 vol.5

[45] James Cai Andrea fimagalli, Chi Guan, “Centralized vs. Distributed On-Demand Bandwidth Reservation Mechanisms in WDM Ring”OFC 2001, pp MH2-1- MH2-3 vol. 1

[46] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour "Evaluation of job-scheduling strategies for grid computing", In *1st International Workshop on Grid Computing*, 2000, pp 191–202.

[47] Consider the strengths and weaknesses of distributed versus centralized ITS telecommunications systems, http://www.itslessons.its.dot.gov/its/benecost.nsf/Lesson?OpenForm&3176B4EBDFF65BD78525728A00766A08%5ELLCats, last accessed 21.09.2007

[48] Phosphorus, Deliverable D.5.2, "QoS-aware resource scheduling", Kostas Christodoulopoulos (editor), 2007

[49] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). GWD-R (Proposed Recommendation), Open Grid Forum, 2007. http://www.ogf.org/documents/GFD.107.pdf

[50] Phosphorus Deliverable D.1.1, "Requirements and specifications of interfaces architecture for interoperability between NRPS, GMPLS, Middleware", Sergi Figuerola (editor), 2007

[51] Phosphorus Deliverable D.5.3, "Grid Job Routing Algorithms", Anna Tzanakaki (editor), 2007

[52] Christoph Barz, Thomas Eickermann, Markus Pilz, Oliver Wäldrich, Lidia Westphal, Wolfgang Ziegler, "Co-Allocating Compute and Network Resources - Bandwidth on Demand in the VIOLA Testbed", CoreGRID Symposium, Rennes, France, August 2007, Springer, 2007, CoreGRID Series, Towards Next Generation Grids, pp. 193 – 202

[53] VIOLA (Vertically Integrated Optical Testbed for Large Applications in DFN), http://www.viola-testbed.de/, last accessed on 19.09.2007

[54] Lars-Olof Burchard, "Analysis of Data Structures for Admission Control of Advance Reservation Requests", IEEE Transactions on Knowledge and Data Engineering, 17(3):413–424, 2005

[55] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees", IEEE International Conference on Network Protocols, Atlanta, GA, USA, 1997

[56] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees", IFIP Fifth International Workshop on Quality of Service, pages 115–126, 1997

[57] The EGEE project, "Statistics for grid-computing jobs within the EGEE project", http://ccjra2.in2p3.fr/EGEE-JRA2/QAmeasurement/index.php, last accessed 28.10.2006

[58] A. Schill, F. Breiter, and S. Kuhn, "Design and evaluation of an advance reservation protocol on top of RSVP", IFIP 4th International Conf. Broadband Communications, pages 23– 40, 1998

[59] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows", Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1993

[60] J. R. Iyengar, P. D. Amer, and R. Stewart, “Concurrent multipath transfer using sctp multihoming over independent end-to-end paths,” IEEE/ACM Transaction on Networking, vol. 14, no. 5, pp. 951–964, 2006

[61] C. Barz, P. Martini, M. Pilz, F. Purnhagen, „Experiments on Network Services for the Grid“, Proc. of 32nd IEEE Conference on Local Computer Networks (LCN), Dublin, Ireland, October 2007

9 Glossary

In this chapter definitions that are particularly relevant in the context of computational Grids and advance reservations are provided. References to documents are provided if normative bodies (IETF, ITU, OGF,…) describe the keyword more precisely or the definition is derived from the specified document.

Keyword	Source	Definition
Grid	[OGF-GFD81], [OGF-GFD-I.112]	A system that is concerned with the integration, virtualization, and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organizations) across traditional administrative and organizational domains (real organizations). The term (computational) Grid is a concept in modern distributed computing architectures. An objective of a computational Grid is to extend the concept of application, data and system virtualization. These extensions allow applications and data to be used in a flexible, shared environment provided by IT service providers.
Job	[OGF-GFD81]	A user defined task that is scheduled to be carried out by an execution subsystem. In OGSA-EMS, a job is modeled as a manageable resource, has an endpoint reference, and is managed by a job manager.
Resource		A resource is an entity in a Grid environment. The term encompasses entities that can be used as a whole or be separable, e.g. the entity provides a given capacity like disks, and networks. Entities can be pooled (e.g. hosts, software licenses, IP addresses). Furthermore, entities such as

		processes, print jobs, database query results and virtual organizations may also be represented and handled as resources.
NRM		A Network Resource Manager (NRM) is a synonym for a NRPS.
NRPS		Network Resource Provisioning System (cf. deliverable D1.1 by WP1)
Allocation	[OGF-GFD81]	The process of assigning a set of resources for use by a job.
Capability		
Choreography	[OGF-GFD81]	Choreography describes required patterns of interaction among services and templates for sequences (or more structures) of interactions.
Orchestration	[OGF-GFD81]	Orchestration describes the ways in which business processes are constructed from Web services and other business processes, and how these processes interact.
Workflow	[OGF-GFD81]	Workflow is a pattern of business process interaction, not necessarily corresponding to a fixed set of business processes. All such interactions may be between services residing within a single data center or across a range of different platforms and implementations anywhere.
EPS	[OGF-GFD81]	Execution Planning Service. In OGSA-EMS, a service that establishes relationships between jobs and resources for scheduling purposes.
Job manager	[OGF-GFD81]	In OGSA-EMS, a service that manages a set of one or more job instances, which may be structured (e.g. a workflow or dependence graph) or unstructured (e.g. an array of non-interacting jobs). The job manager encapsulates all aspects of job execution, including interacting with execution planning services, the

		provisioning system, containers, and monitoring services. It may also deal with failures and restarts, it may schedule jobs to resources, and it may collect agreements, reservations and job service data.
MPI	[OGF-GFD81]	Message Passing Interface: a standard API for implementing message passing libraries. MPI libraries are generally used to coordinate activity within parallel applications. See http://www.mpi-forum.org for more information.
Notification	[OGF-GFD81]	A message communicating the details of an event to an interested party.
Provisioning		The activity of specifying, reserving, allocating and deploying the set of resources required to accomplish a task.
Quality of service (QoS)	[OGF-GFD81]	A measure of the level of service attained, such as security, network bandwidth, average response time or service availability.
Bandwidth		See *capacity.*
Capacity		The term capacity is used in conjunction with a resource and usually describes the current or maximum quantity that can be used. Capacity is given in a resource specific unit (e.g. kilo bytes, bit per second). The (technical) properties of the resource determine the utilization, e.g. usage in specific quantities.
Delay		Duration given by the sending and the reception of a signal or request by the addressee.
Jitter		Variation of delay.
WAN		Wide Area Network

MAN		Metropolitan Area Network
LAN		Local Area Network
Release	[OGF-GFD81]	The action of returning an allocated resource to the pool of available resources.
Pre-staging		A set of actions (prearrangements) executed before the ultimate action either can be performed or is performed more efficiently.
Post-staging		A set of actions (arrangements) executed after the ultimate action, e.g. to transfer or secure results.
Reservation	[OGF-GFD81]	The process of reserving resources for future use by a planned task.
Resource allocation		See *allocation*.
Resource configuration	[OGF-GFD81]	The process of adjusting the configurations of a set of resources to meet the requirements of the task to which they have been allocated. For example, configuration may involve setting appropriate parameters and storing policies for middleware, O/S, firmware and hardware. Resource configuration may be preceded by resource deployment.
Schedule	[OGF-GFD81]	A mapping (relation) between services and resources, possibly with time constraints. A schedule can be extended with a list of alternative schedule deltas.
Schedule deltas	[OGF-GFD81]	A set of transformations that may be produced for use if some part of the current schedule becomes invalid.

		For example, if a resource becomes unavailable, it may be possible to use a schedule delta rather than reschedule the job from scratch.
Scheduling		The process of creating a schedule.
Service		Often an abbreviation for Web Service, a software system designed to support interoperable machine- or application-oriented interaction over a network. Economically, a service can be described as an activity that is performed by the service provider for a customer or user. Usually, performing a service results in a benefit for the customer.
SLA, Service Level Agreement	[OGF-GFD81]	A contract between a provider and a user that specifies the level of service that is expected during the term of the contract. They might specify availability requirements, response times for routine and ad hoc queries, and response time for problem resolution (network down, machine failure, etc.).
Use case	[OGF-GFD81]	A use case captures interactions of an agent or entity with a system and/or its constituents, and the expected behavior of the parties as a consequence, where such interactions are directed towards achieving a specific goal. Different sequences of behavior, or scenarios, can unfold, depending on the particular requests made and conditions surrounding the interactions. The use case description may include the environment and context salient to each scenario.
Network domain		A group of computers and devices on a network that are administered as a unit with common rules and procedures. Within the Internet, domains are defined by a range of IP addresses or Autonomous Systems.
Domain		See Network domain.
Admission control		In the context of Quality of Service (QoS) within networks or SLAs the admission control decides if a requested resource or

		the requested capacity of a single or compound resource can be used by a customer. The process of (network) admission control requires a specification of the intended service (e.g. network capacity in bits per seconds), information about resources available, and a strategy inspecting and judging the requested service. Admission control is useful in situations where a certain number of requested services (calls, reservations) share a resource, while either more requests cause significant service degradation for all users (congestion) or the scheduling offers a benefit to the users.
Circuit switching		A paradigm in network technologies in which a dedicated path is established usually between two endpoints through one or more intermediate nodes. Commonly, the dedicated path can be used to identify traffic handled with certain QoS parameters.