

09.06.06

Deliverable DJ3.3.2: Functional Specification of GÉANT2 Inter-domain Manager (IDM) Prototype



Deliverable DJ3.3.2

Contractual Date: 30/11/05
Actual Date: 09/06/06
Contract Number: 511082
Instrument type: Integrated Infrastructure Initiative (I3)
Activity: JRA3
Work Item: 3
Nature of Deliverable: R (Report)
Dissemination Level: PU (Public)
Lead Partner: GARR
Document Code: GN2-06-091v3

Authors: Maarten Büchli (DANTE), Mauro Campanella (GARR, Editor), Gábor Ivánszky (HUNGARNET), Radosław Krzywania (PSNC), Damir Regvar (CARNET), Michal Przybylski (PSNC), Victor Rejis (HEANET), Afrodite Sevasti (GRNET), Maciej Stroiński (PSNC), Chrysostomos Tziouvaras (GRNET), Jan Węglarz (PSNC), Dave Wilson (HEANET)

Abstract

The Bandwidth on Demand service, as defined in the context of the GN2 project, aims at providing a guaranteed capacity, connection oriented service between two end points. The overall BoD service architecture is described in Deliverable DJ3.3.1, 'GEANT2 Initial Bandwidth on Demand Framework and Architecture'. This deliverable specifies a key module of the JRA3 BoD system: the Inter-Domain Manager. Part A reports its functional specifications and Part B its implementation specifications with specific details for its first release, named as Prototype or Phase 0. The Pathfinder algorithms specifications are still in a draft stage and are not detailed. The abstract notation specification is detailed, and a subset will be implemented in the IDM prototype. A fast prototyping cycle of the implementation is currently underway in order to evaluate open issues and validate the specifications.

Table of Contents

0	Executive Summary	v
1	IDM prototype overview	1
1.1	IDM general architecture	1
1.2	The IDM prototype	2
1.2.1	Blocks implemented in the IDM prototype	3
1.2.2	Partially implemented blocks in the IDM Prototype	3
1.2.3	Unimplemented blocks in the IDM prototype	4
1.3	Prototype assumptions	4
2	IDM general issues	5
2.1	Addressing	5
2.2	Advance scheduling	8
2.2.1	Resource scheduling at provisioning time	9
2.2.2	Soft resource scheduling at request time	9
2.3	Logging and accounting	10
2.4	Signalling framework	11
3	Abstract Network Representation	13
3.1	Introduction	13
3.2	Entities which describe a BoD service	13
3.2.1	adminDomain entity	14
3.2.2	provDomain entity	14
3.2.3	node entity	15
3.2.4	port entity	15
3.2.5	link entity	16
3.2.6	path entity	17
3.2.7	linkToPath entity	19
3.3	Entity-Relationship (E-R) diagram	19
4	Pathfinder	21
4.1	Overview	21

4.2	Pathfinder operation	22
4.3	Future considerations	24
5	User Access and Main Request Handling Logic	25
5.1	User Access	27
6	Authentication and Authorisation Infrastructure (AAI)	29
7	Domain Manager	30
8	Flow Diagram	31
9	IDM prototype Use Cases	33
9.1	Use Case (UC) 1: User service request acceptance	33
9.2	Use Case 2: User status request	34
9.3	Use Case 3: User cancels service request	35
9.4	Use Case 4: Service schedule procedure	36
9.5	Use Case 5: Reservation scheduling procedure at a domain along the reservation path	38
9.6	Use Case 6: Reservation scheduling procedure at the last domain along the reservation path	40
10	Implementation Notes	42
11	References	45
12	Acronyms	46

Table of Figures

Figure 1: The Bandwidth on demand system architecture showing its modules.	1
Figure 2: Inter-domain Manager main blocks	2
Figure 3: Blocks implemented in the IDM prototype	3
Figure 4: BoD addressing example, showing the data and control plane addresses	7
Figure 5 . Sample scheduling diagram	10
Figure 6: Signalling framework	11
Figure 7: E-R diagram	20
Figure 8: Placement of the Pathfinder module within the IDM prototype.	22
Figure 9: Reference case for inter-domain topology	23
Figure 10: Main Request Handling Logic module (shaded box) interactions with other IDM blocks and the DM	26
Figure 11: User Access sub-block interactions with external requestors.	28
Figure 12: Flow diagram of a reservation request processing phase	31
Figure 13: Flow diagram of a reservation commit phase	32
Figure 14: Flow diagram for case UC1	34
Figure 15: Flow diagram for case UC2	35
Figure 16: Flow diagram for case UC3	36
Figure 17: Flow diagram for case UC4	38
Figure 18: Flow diagram for case UC5	40
Figure 19: Flow diagram for case UC6	41
Figure 20: Modules to Java packages mapping	43

0 Executive Summary

The overall BoD service architecture is described in Deliverable DJ3.3.1: 'GEANT2 Initial Bandwidth on Demand Framework and Architecture' [DJ3.1.1]. One of the key elements of the BoD service is the Inter Domain Manager module (IDM). It is the original part of the system, as interdomain control plane communications at the various layers are still under development in all the relevant standardization bodies. The IDM is, at the same time, a key element of the BoD service, as the service goal is to provide a multi-domain path through distributed, cooperating domains, without any central management. This deliverable specifies the Inter-Domain Manager: It provides a detailed view of its functional specifications and some initial information on its implementation specifications. A fast prototyping cycle of the implementation is currently underway in order to evaluate open issues and validate the specifications. In the next steps of the implementation, more advanced and feature-rich versions of the IDM module will be delivered in a phased manner.

The deliverable describes the characteristics of the prototype implementation of the IDM as well as provides a set of the functional principles of next versions of the IDM module. The Pathfinder algorithms specifications are still in an early stage and are not fully detailed. An abstract notation specification is reported, although only a subset will be implemented in the prototype.

1 IDM prototype overview

1.1 IDM general architecture

The IDM is a key module of the Bandwidth on Demand system, the architecture of which is described in Deliverable DJ3.3.1 [DJ3.1.1]. A graphical overview of the general system architecture is copied for reference in **Figure 1**, listing the modules of which the system is composed. In brief, the Inter Domain Manager is responsible for inter-domain and user communication, while the Domain Manager (to be developed in future stages of the project) is responsible for local domain management. The technology proxies act as bridges between the BoD system modules, which operate using an abstract network representation, and the real network, which may be implemented using different technologies.

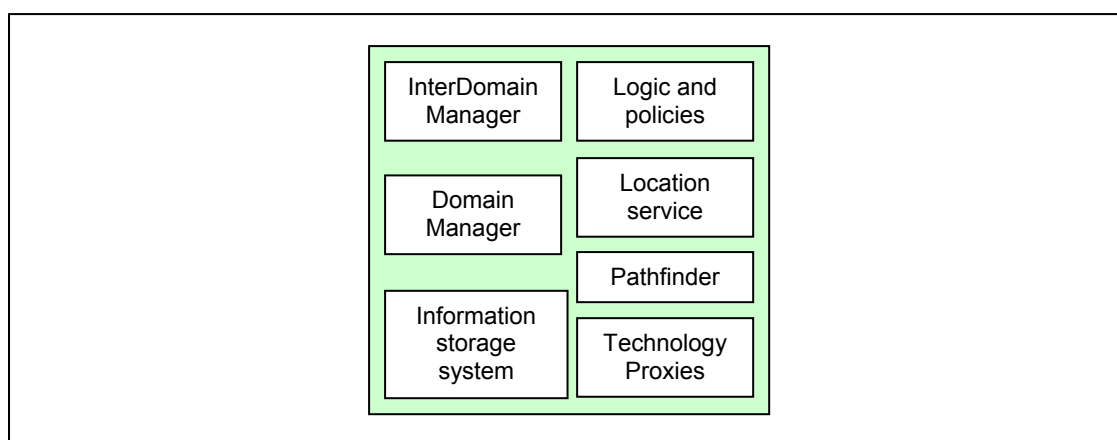


Figure 1: The Bandwidth on demand system architecture showing its modules.

A more detailed overview of the IDM module functionality is summarized below (Figure 2). The IDM module:

- is the one and only ingress point to the BoD system. It receives and processes multi- or single domain BoD reservation requests from users or from other IDMs of neighbouring domains
- selects the next domain to contact to establish an end-to-end path for serving a reservation request and may select a list of routes/paths through the BoD domains from end-to-end,
- participates in a commit process between all IDMs along the end-to-end path used to serve a BoD reservation request

- interacts with the AAI service, when available, to authenticate the identity of the BoD service requestor and his authorization privileges for the BoD service. It always applies the local domain BoD rules and policies.
- is based on the authentication of each BoD service requestor; it uses a credit management system for the controlled allocation of bandwidth resources among the BoD users of the domain.
- operates an accounting and logging sub-system that keeps, processes and presents accounting data of the BoD service usage and availability per BoD session and in general within the domain. This functionality is particularly useful for assessing the successful deployment of the service as a whole and for signalling the need for adjustments in resources set aside for the BoD service purposes in cases that these resources are under- or over-utilized
- each IDM implements its own policies, using also the information provided the DM, for allocation of BoD resources and for management of the BoD service within the domain

The IDM relies on the local Domain Manager (DM) to implement in the respective network domain the service requested in the form of a provisioned circuit. It is the DM that deals, through the technology proxies, with the physical details of the particular network domains and the different technologies used to implement the BoD circuit. The high level view of an IDM is shown in Figure 2, where its building blocks are listed. For more details refer to [DJ3.3.1].

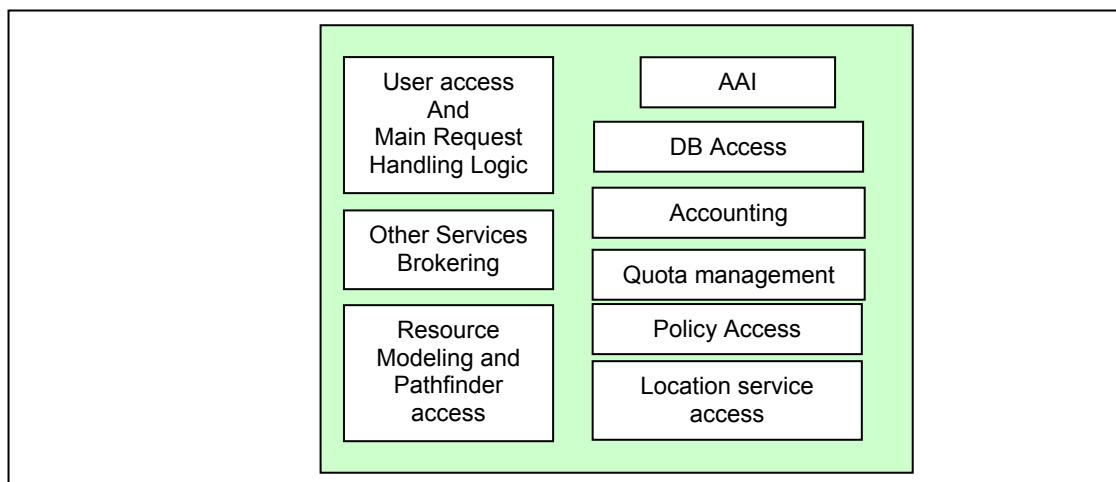


Figure 2: Inter-domain Manager main blocks

1.2 The IDM prototype

The IDM is the key element of the BoD system and has to fulfil the complex inter-domain communication task. The IDM prototype (Phase 0 of the IDM releases) will be implemented as a standalone system, without relying on external services or the implementation of other modules. The modular architecture allows this phased implementation so as to avoid the creation of complex supporting modules at early stages of the IDM

deployment. The external modules, like AAI, can be implemented as dummy modules with simple, static information.



Figure 3: Blocks implemented in the IDM prototype

The IDM prototype will not provide all the functionalities of the final system, but a key subset. **Figure 3** depicts the implementation choices for the IDM prototype.

1.2.1 Blocks implemented in the IDM prototype

The ‘User access and Main Request Handling Logic’ prototype module implementation will validate and refine the user-to-IDM and IDM-to-IDM interfaces and signalling as well as the request handling procedures and the IDM structure.

The ‘Resource Modelling and Pathfinder’ prototype modules are the key components to test and validate the network abstraction, its handling at the IDM level and the procedures to determine the end-to-end path. The prototype implementation of these modules will employ a simplified abstract representation and Pathfinder implementation, using predefined topologies; in the future it will evolve to base its operations on a routing engine based on the Open Shortest Path First protocol with Traffic Engineering extensions (OSPF-TE).

1.2.2 Partially implemented blocks in the IDM Prototype

A simple AAI module will be implemented for intra-domain security and authentication and authorization of requests, with static information.

A database will be used to store information about services and reservations. The database strategy is to avoid choosing a specific and complex database management system for the prototype. In order to allow the greatest level of flexibility:

Project:	GN2
Deliverable Number:	DJ3.3.2
Date of Issue:	09/06/06
EC Contract No.:	511082
Document Code:	GN2-06-091v3

- the database access will be provided by a simplified user access interface created, as an example, using Data Access Objects (a standard Java object to abstract and encapsulate all access to the data source).
- the topology, paths and constraints will be loaded from static XML files to emulate the path finding process and the search for local domain constraints.

The accounting and logging module is implemented in the IDM since the beginning, thus it is part of the IDM prototype. Its scope is primarily to facilitate the debugging of the IDM modules as well as data and command flow analysis, rather than to produce reports.

1.2.3 Unimplemented blocks in the IDM prototype

The remaining blocks (see **Figure 3**), 'Other services brokering', 'Quota management', 'Policy access' and 'Location service', are not relevant to the key functionalities of the IDM and will be implemented and tested in the next phases of the prototype of the IDM. The 'Location Service' basic functionalities will be automatically provided by the use of the WEB service architecture [webservices] e.g. via UDDI, which is a SOAP-based Web service for locating Web services and programmable resources on a network. The 'Policy access' module is expected, when implemented (in later phases of the IDM deployment), to introduce the definition of users and user groups within the BoD system and associate with them different levels of privileges for using the system and for applying for BoD services.

1.3 Prototype assumptions

It has been decided that the IDM is the only access point to the JRA3 BoD system, while the DM is considered able to push requests to the IDM. Each reservation request must be presented to the IDM from the source domain of the end-to-end path.

Advance reservation, that is the possibility to submit a request with starting time in the future, is considered a key feature of the BoD system and it will be immediately implemented in the IDM prototype.

The functionality of 'modifying' the properties of an on-going reservation is not scheduled for the IDM prototype.

The DM is assumed to be a dummy module in the current phase and the IDM prototype will not directly configure the network.

2 IDM general issues

The prototyping activity allows further elaborating on a number of issues related to the IDM architecture.

2.1 Addressing

A network providing a Bandwidth on Demand service must access its nodes in an out-of-band method. An addressing scheme is therefore needed for the control/management plane, which is independent from the data planes it controls.

The main requirement for the BoD addressing scheme is that the addresses assigned to the interfaces used for inter-domain traffic are unique within the set of domains that participate in offering the BoD service. In addition, it is preferable to use an addressing scheme that is already supported by existing routing protocol implementations. Indeed, it is not expected that the JRA3 activity will implement a routing protocol from scratch. It will use existing ones and, if necessary, modify them. Three well-known addressing schemes were therefore considered: IPv4, IPv6 and NSAP (ISO) addresses. Because of the abundance of available address space, it was decided to use public IPv6 addresses. Furthermore, existing routing protocol open-source implementations such as Quagga [quagga] or XORP Project [xorp] are supporting IPv6, which can contribute to the Pathfinder module implementation.

More specifically, the following approach to addressing at the data and control plane has been adopted both for the IDM prototype implementation and the future developments:

- Physical or logical interfaces at individual equipment modules within a single domain have their technology-specific identifiers. These identifiers will be stored and associated with the corresponding interfaces within the technology-specific topology database schema, held by the DM module, in later phases of the BoD system implementation
- At the IDM layer, the abstract, technology-neutral representation of the underlying domain's topology is maintained (see section 3 of this document) for the IDM prototype and subsequent phases of the IDM implementation. In this representation, there is a mapping of physical/logical interfaces, equipment modules, links etc. of the technology-specific representation of the domain to their abstract topology

equivalents. In this mapping, technology-specific identifiers at the DM layer are mapped to globally unique IPv6 addresses at the IDM layer and within the abstract representation. Thus, at the IDM layer, all addressable elements obtain their unique IPv6 addresses.

As an example, the leftmost upper router in **Figure 4** has a Gigabit Ethernet port named “gal-pe1 gig1/1.2” and a separate ID as technology specific identifier. The port has a IPv4 and IPv6 data plane address (not shown in the figure) and has an IPv6 control plane address “2001:770:1FFF:10::1”. The interface participates in the IDM operations (e.g. path-finding) with its IPv6 control plane address and its configuration can be signalled from the IDM using the router’s IPv6 control plane address (not shown in the figure).

In the IDM prototype there will be no service lookup available, limiting the possibility of searching neighbour BoD systems. Therefore in the current approach the unique identifier of a BoD domain will be defined by the web service address of its specified BoD system. The identifier will be in the form of a URL, containing the DNS name of the host running the IDM system, e.g. idm.geant2.net.

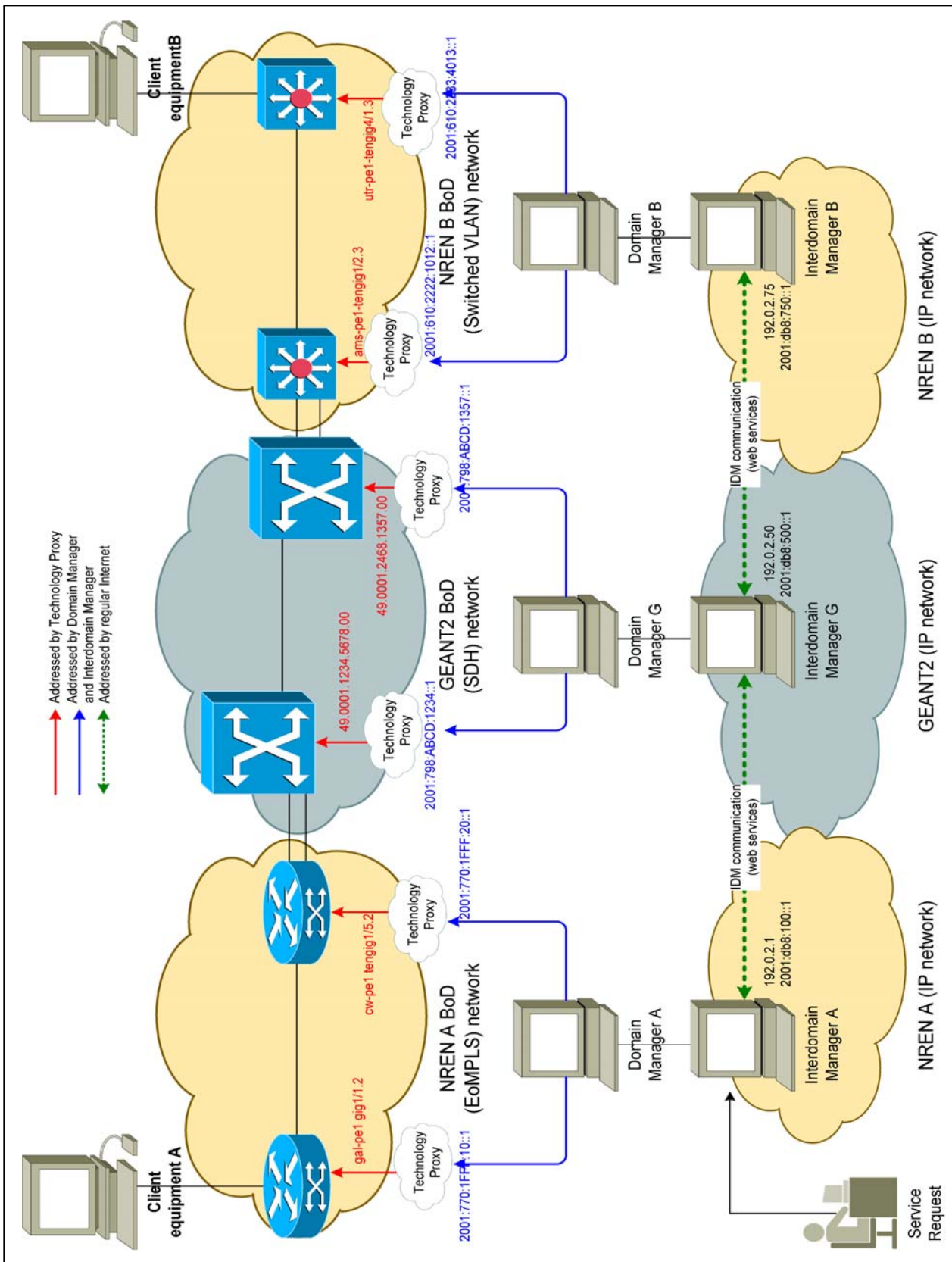


Figure 4: BoD addressing example, showing the data and control plane addresses

Project:	GN2
Deliverable Number:	DJ3.3.2
Date of Issue:	09/06/06
EC Contract No.:	511082
Document Code:	GN2-06-091v3

2.2 Advance scheduling

Advance scheduling is a function to support BoD service requests with a starting time in the future; the IDM will implement advance scheduling since its Phase 0.

For the final system two scenarios have to be considered:

1. The advance scheduling function is part of the IDM. The IDM has to maintain a detailed view of the internal topology of the network and its resource usage/availability and to check its validity with the DM before accepting a reservation.
2. The advance scheduling function is part of the DM. The IDM is basically a client to the DM just as any other user. In this case the IDM does not need to have knowledge of the complete internal topology of the network.

The option of early resource availability check in both the IDM and DM is not feasible, as it is considered redundant and prone to synchronization problems. A decision of the most suitable scenario for the advance reservation will be based on the prototype results.

Two functional components can be identified as part of advance scheduling.

The first functional component is a calendar to store the start and end time for each service request. Upon the start or end time this function triggers the provisioning or deletion process of the network service. This function will be tested in the IDM prototype.

The second component is devoted to resource availability check. This function can be implemented in two ways:

- Resource availability is checked at request time, but it is reserved only at the start time of the reservation. In this case the service may not be available at the specified start time if the resources are no longer available in the network. Network Management Systems (NMS) and control planes such as GMPLS/G.ASON typically follow this approach.
- “Soft” resource scheduling at the time of the service request, i.e. before the start time of the reservation. This approach allows the system to provide the user with an assurance that the service will be provisioned at the requested time (except the case of unexpected outages). This approach requires the module to maintain a database that stores the resource availability within the network between the present and the maximum allowable reservation time in the future, e.g. six months later and to pre-allocate resources in the network

In the following paragraphs both approaches will be elaborated on in more detail. The approach of “Soft” resource scheduling at the time of the service request will be used for the prototype of the IDM.

2.2.1 Resource scheduling at provisioning time

The approach of resource scheduling at the time of provisioning is based on the notion of a well-planned and provisioned network in order to have a very low blocking probability at any time. This approach works well for large numbers of service requests where the requested capacity is relatively small compared to the capacity of the links in the network and the duration of the service is short. In that case Erlang calculations for the estimation of traffic can be used to dimension the network such that the blocking probability is very small. This approach is typical for the telephony networks. Such an approach is much less suited to a BoD system devoted to circuit switching for data traffic at high capacity. Even if the number of requests is low, each request may require a large fraction of BoD capacity, and thus making it difficult to satisfy the advance scheduling requirements with real-time reservations system. This holds true independently that the function is implemented in the IDM or in the DM module.

2.2.2 Soft resource scheduling at request time

The approach of advanced resource scheduling at the time of the service request seems to fit well for the BoD service because the number of service requests is small and the required capacity per request is high (equal or larger than 1 Gb/s). In addition, the duration of the service is expected to vary between hours and months. If the capacity between two endpoints is not available at the specified time the user will be informed quickly after his service request.

This approach will increase the size of the resource database, as it will get a time dimension and it needs to track available capacity per link for any given time instance in the future. To simplify the implementation of the prototype, phase 0 will only implement capacity reservation; other resources, such as specific VLAN numbers, will not be booked. The resources are booked per link, in the form depicted in Figure 5.

When reserving network resources in the future, the system has to address a series of issues

- Network topology changes may occur between reservation time and start of the service.
- Scalability of the resource database in case the number of services instances increases.
- Establishment of links in coordination with the maintenance calendar.
- Proper management of resources and network status information to ensure coherence of the information at any instant, in cases of concurrent requests or when different IDM modules attempt to access and update this information.

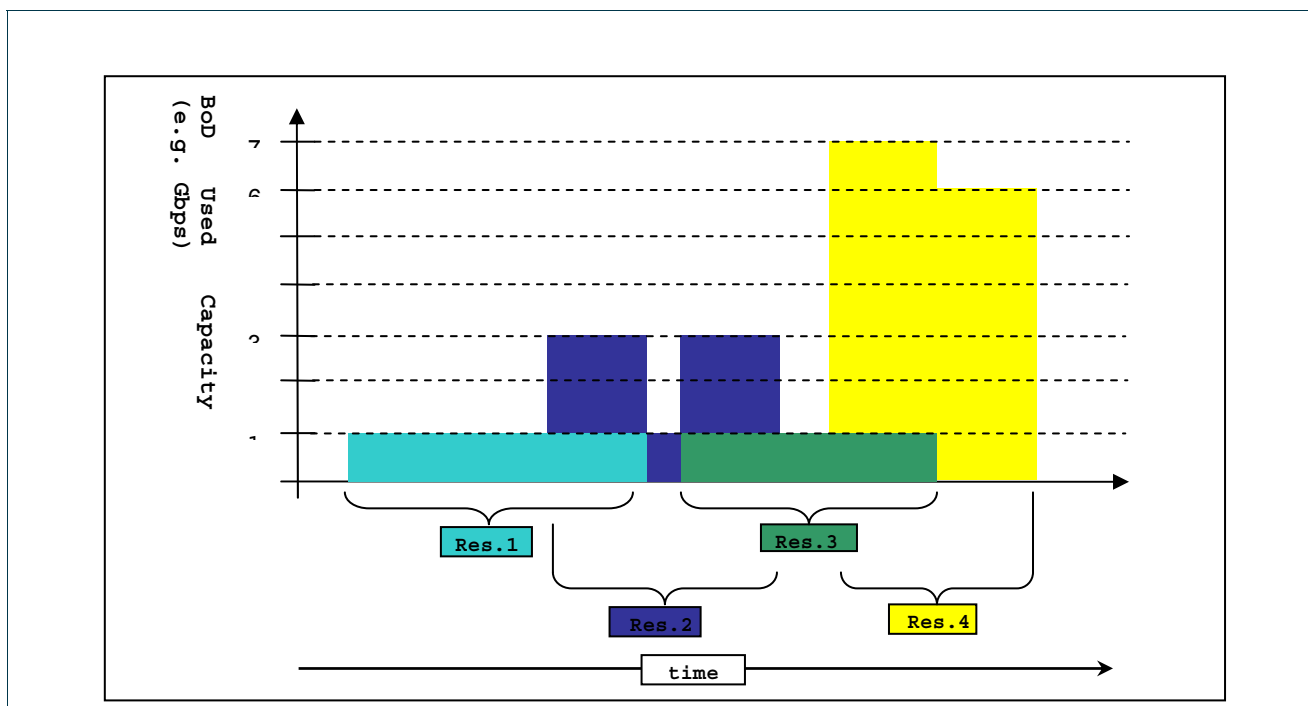


Figure 5 . Sample scheduling diagram

Figure 5 shows an example of the evolution of BoD capacity usage as a function of time. This type of information has to be kept for every BoD circuit and the reservation has to be validated according to future availability in the path. Summarizing, advance scheduling will be implemented in the IDM prototype, with functionality limited to capacity pre-reservation and network configuration at service start time. For sake of simplicity, the network is considered stable and the network reservation database will not be periodically verified against network status. Taking into account other resources, like VLAN numbers, appreciably increases the complexity of the reservation system and additional effort is needed to improve advance reservation in the next BoD system releases.

2.3 Logging and accounting

Logging and accounting are important functions of the BoD system. They allow both monitoring the behaviour of the system and the possibility to provide analysis and reports of its use.

Logging will be implemented in the IDM prototype in particular to facilitate debugging. Logs should contain various BoD service metrics, as an example the number of service requests handled, time stamping each step of a service reservation or deletion the service, number of failed attempts, number of failures, who requested them (user, other IDM, or pushed from DM). Two types of logging are identified, error logging and session processing logging, with the former one serving for locating problems in the IDM operation by recording the messages exchanged between IDMs and the second one being used for tracking the steps in each reservation processing instance.

Accounting will be implemented as a set of tools to analyze logs in the next phase of IDM deployment (Phase 1). Accounting will provide information mainly about system use, performance, average and peak system load, network utilization. The BoD manager should log each reservation in the system. For each reservation all service parameters and the event flow should be logged.

The system should have the possibility to log every transaction. An accurate time stamping of the logs is relevant for a successful debugging and accounting. A time synchronization between IDMs and their components using the NTP protocol is considered sufficient for the IDM prototype.

More details about the logging and accounting reports produced by the IDM prototype will be provided by the deliverable DJ3.4.1 'Technical documentation for the inter-domain BoD service manager (IDM)'.

2.4 Signalling framework

The interdomain BoD system will support the following functions:

- Request an interdomain circuit service
- Cancel an existing service
- Query the details of an existing service

The “modify” service function will be introduced in later stages of the IDM deployment. In order to support the above functions in an interdomain environment there exists the need for communication between the different domains. In other words, a signalling protocol is required between the IDMs, which are the only ingress and egress points of the system. Every IDM communicates only with the IDMs of the adjacent domains (i.e. directly connected domains). This results in a chain like approach for signalling as illustrated in **Figure 6**, which is very similar to the behaviour of the RSVP protocol. The request should always be initiated in the domain where the service has one of its end points (domain 1 in **Figure 6**).

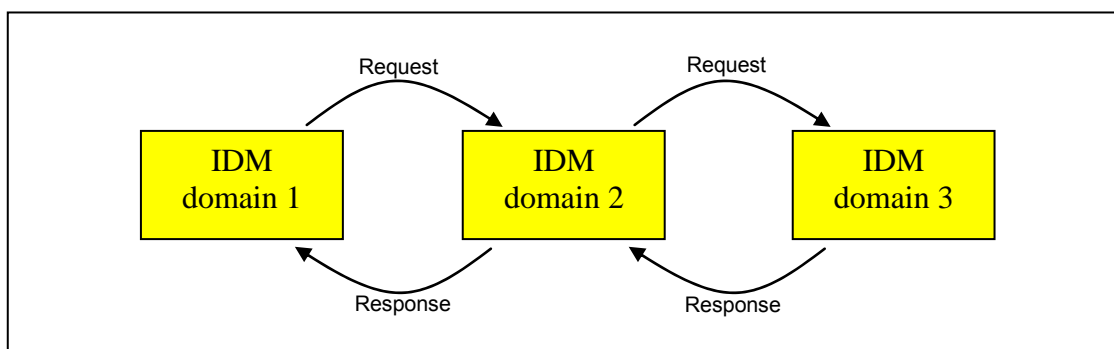


Figure 6: Signalling framework

This chained architecture alleviates the need for a centralized management of end-to-end path setup. In the NREN environment this allows for full flexibility of each participating NREN in the path establishment process, as the NREN can independently decide if and how a request is served locally based on local policies and availability and propagate the request as appropriate.

During each step of the signalling request process, the following operations are performed:

- Each domain checks in its local reservation database whether resources are available between the start and the end time specified in the request. If the resources are available, it will pre-reserve them such that they are marked as unavailable for other service requests. For the IDM prototype, the network is assumed to be stable and there is no need for constant updates on the status of available resources from the network to the IDM. In the final system this function will be implemented.
- End-to-end parameters may have to be specified and agreed, e.g. VLAN identifiers. In that case, the source domain will provide a list of possible VLAN identifiers and each domain will add its local list of VLAN identifiers that can be used. In this way, technology specific information from the individual domains is accumulated towards the destination domain as a result of the signalling process. It is then the destination domain that decides on these end-to-end parameters. Note that certain parameters may only be relevant to the source and destination domain.
- Parameters used to stitch together circuits in neighbour domains may have to be specified and agreed. These are technology specific parameters, such as timeslots or VLAN identifiers, and are agreed between two adjacent domains.

Once the IDM of the destination domain receives the request it will select the end-to-end parameter(s) and return a response back towards the IDM of the source domain. Upon this response each domain will commit the resources that were already pre-reserved and will store the end-to-end parameter(s) that are specified in the response message.

One of the outcomes of the testing of the IDM prototype will be to determine how long it takes before a response is returned to the first domain. In case it turns out that domains need e.g. hours or days to go through the steps described (for example due to human intervention) the above signalling approach may have to be revised.

In that event, an alternative approach would be to send the response back before reserving resources and sending an additional message back to the IDM of the source domain to indicate the resources were pre-reserved correctly. Once all domains have confirmed the resources were pre-reserved correctly, a signalling message should be sent to commit the resources. This behaviour allows to provide a status report to the requestor during the set-up time, if the latter is too long.

The user must be able to query the status of his or her request.

3 Abstract Network Representation

3.1 Introduction

This section aims at defining the database schema that will be used for providing a consistent view of the “BoD network” at the inter-domain level. The schema has to be refined according to the results of the prototype testing. The IDM prototype will use a subset of the schema.

A separate instance of the proposed database will be installed at every administrative domain that participates in the BoD service. Defined entities, detailed below, will be used both by the IDM Pathfinding module when searching and creating the list of paths which satisfy the request, and by the User Access and Main Request Handling Logic module to cross-check the technical feasibility of establishing a specific path.

This representation starts by considering the technology-neutral elements of the network.

The deliverable DJ3.2.2 “Initial Review of Technologies Related to the Provision of Bandwidth-on-Demand (BoD) Services” provided an overview of related work on this topic.

3.2 Entities which describe a BoD service

This section presents the entities to be used in the database. Each instance of the proposed database located at a single domain will contain information about the details of the local domain and limited information about other domains as defined at section 5 of this deliverable. Data consistency between the databases is not dealt with in this section; the responsibility is assumed to be on the module, which updates the database.

The definition of entities (tables) was made on the basis of producing a straightforward Entity Relationship (E-R) diagram, which captures the relationship among these entities in an efficient way.

The proposed database schema is targeted at the definition of a database to be used in the final phase of the BoD service. The effort for abstraction is an ongoing activity and the IDM prototype will deploy a reduced version of the proposed schema.

3.2.1 adminDomain entity

The adminDomain entity represents a domain, under the control of a single administrative authority, and refers (for example) to an NREN, GÉANT2 and potential BoD clients.

The following table lists the attributes of the adminDomain entity. The entries of this entity include all domains offering the BoD service, including the domains of potential BoD clients.

Attribute Name	Description
adminDomainID	It identifies an administrative domain.
ASID	It identifies the public Autonomous System (AS) number of the particular administrative domain. It is possible that a domain can include more than one AS number, although it is not the case for any European Research Network. This case is not examined here so as to keep the E-R diagram as simple as possible.
Name	It is used for specifying the name of the particular domain, for instance GÉANT2.

3.2.2 provDomain entity

An administrative domain can potentially include more than one “provisioning domains”, that is subsets in the administrative domain that use a common provisioning method. Each of those subsets is represented by a provDomain entity.

The following table lists the attributes of the provDomain entity. The entries of this entity include the provisioning domains that comprise the local administrative domain.

Attribute Name	Description
provDomainID	It identifies a local provisioning domain
provMethod	{lambda SDH SONET Ethernet L2 MPLS VPN QoS PIP IP MPLS QoS GMPLS UCLP }. It identifies the BoD provisioning method that is used at the particular technological domain.
adminDomainID	It identifies the administrative domain that the particular technological domain belongs to.

3.2.3 node entity

A node entity represents a network element that is part of a domain.

The following table lists the attributes of the node entity. The entries of this entity include the nodes part of the local technological domain.

Attribute Name	Description
nodeID	It identifies a local node.
Type	{PSC L2SC TDM LSC FSC ... }. It identifies the type of the particular network element, according to the GMPLS architecture [RFC 3945].
Address	The technology-specific address that is used to access the particular network device (e.g. IP or TNA address, please refer to section 2.1 for more details)
bodAddress	The control plane addresses that BoD system modules (i.e. both IDM and DM modules) will use to refer to the particular network node (please refer to section 2.1 for more details)
provDomainID	It identifies the provisioning domain that the node belongs to.
adminDomainID	It identifies the administrative domain that the node belongs to.

3.2.4 port entity

A port entity represents a physical or virtual interface on a node.

The following table lists the attributes of the node entity. The entries of this entity include both the ports residing at the local administrative domain and all edge ports residing at the neighbour administrative domains that participate at the BoD service, including BoD client edge ports.

Attribute Name	Description
portID	It identifies a port.

Address	The address that is used to access the port. The format of this address will be examined on a per domain case, since there are many options as a function of the deployed technology.
bodAddress	The address that BoD system modules will use to refer to the particular port.
technology	{Fibre WDM SDH SONET Ethernet IP}. It identifies the networking technology that the port uses.
nodeID	It identifies the node that the port belongs to.
adminDomainID	Its states the administrative domain that the port belongs to. This attribute is mainly used for identifying the administrative domain that a non-local port belongs to
bundled	{true false}. It states whether this is a “bundled” port or not, that is a port that comprises at least two physical or virtual ports. The bundling concept is introduced at [4] and targets at routing scalability enhancements to multi-lambda networks.

3.2.5 link entity

A link entity represents a circuit between two ports. The nodes connected by a link can be:

- neighbours belonging to the local administrative domain,
- neighbours belonging to different administrative domains where one of the nodes resides at the local administrative domain (interdomain links),
- non-neighbouring nodes.

In the last case two options are possible:

- The link entity represents a connection among two edge ports of a domain in order to represent the intra-domain part of an end-to-end path as a single intra-domain link. This kind of link will be called “virtual links” and will be used by the IDM path finding module to compute end-to-end paths.

- The link entity represents the union of two or more contiguous local links. This kind of link will be called “composite link”. A composite link is represented exactly as an ordinary link and is treated as such by the path finding algorithms. Composite links can only be composed by intra-domain links.

The following table lists the attributes of the link entity. The entries of this entity include: local links, interdomain links, virtual links and composite links.

Attribute Name	Description
linkID	It identifies a link.
sourcePortID	It identifies the source port of the link.
destPortID	It identifies the destination port of the link.
Direction	{unidirectional bidirectional}. It identifies the directionality of the link. In most cases, the links that will be used for provisioning the BoD service will be bidirectional.
Kind	{regular virtual composite interdomain}. It identifies whether this link is a regular link that is connecting neighbouring ports, virtual, composite or inter-domain.
manualCost	It states the administrative cost of the link (assigned manually and not computed)
Delay	It states the delay of the link in milliseconds.
monetaryCost	It states the monetary cost of the link
minResCapacity	It states the minimum amount of capacity that can be allocated at the link.
maxResCapacity	It states the maximum amount of capacity that can be allocated at the link.
Resilience	{1+1 1:1 none}. It states the resiliency properties of the link. 1+1 is a complete backup using an alternate path, 1:1 uses the backup via other non-dedicated resources.
granularity	It states the capacity granularity of the link.

3.2.6 path entity

A path entity represents a BoD path that can be:

- currently used by a BoD user,

Project:	GN2
Deliverable Number:	DJ3.3.2
Date of Issue:	09/06/06
EC Contract No.:	511082
Document Code:	GN2-06-091v3

- a reserved path that will be activated at a defined time in the future,
- a candidate path produced by the path finding module.

The following table lists the attributes of the path entity. The entries of this entity include all BoD paths for which resources from the local administrative domain are used, will be used or may be used.

Attribute Name	Description
pathID	It identifies a path.
bodUser	It states the user that requested the BoD service.
fromSourceClientPortID	It identifies the source BoD client edge port of the path.
toDestClientPortID	It identifies the destination BoD client edge port of the path.
Direction	{unidirectional bidirectional}. It identifies the directionality of the path. In most cases, the path that will be used for provisioning the BoD service will be bidirectional.
totalManualCost	It states the administrative routing cost of the path (assigned manually and not computed)
totalDelay	It states the delay of the path in milliseconds.
totalMonetaryCost	It states the monetary cost of the path.
reservedCapacity	It states the amount of bandwidth that is dedicated for the path.
Status	{active reserved candidate}. It states the current status of the path.
startTime	The point of time (including date) at which the BoD service provisioning starts. For pre-reserved paths, startTime will refer to the future. This attribute is not applicable for candidate paths.
expireTime	The point of time (including date) at which the BoD service provisioning ends. This attribute is not applicable for candidate paths.

Resilience	{1+1 1:1 none}. It states the resiliency properties of the path.
serviceID	It states the universal identity of a reservation (comprised of one or more paths), recognised by all involved domains. This attribute applies for all kinds of paths (currently used, pre-reserved and candidate).

3.2.7 linkToPath entity

A linkToPath entity is an intermediate entity for representing the links that belong to a single path. Note that a single link can belong to more than one path.

The following table lists the attributes of the linkToPath entity. The entries of this entity include the links that are used so as to implement a single BoD path.

Attribute Name	Description
pathID	It identifies a path.
linkID	It identifies a link that is used for implementing a particular path.

3.3 Entity-Relationship (E-R) diagram

Figure 7 shows the E-R diagram among the tables that were already defined.

Each local administrative domain must include at least one local technological domain. Each local technological domain must include at least one node. Each local node must include at least one local port. However, since non-local ports will be stored in the port table, some entries of the port table may not belong to a local node but to a particular administrative domain that is different from the local domain.

Each link, whether local or not, will be identified by its source and destination port. Each path must consist of at least one link and will be uniquely identified in a global fashion by its serviceID attribute. According to the proposed design, local BoD modules (both IDM and DM ones) will identify the specifics of the path as regards to the fraction that is locally implemented and for the remaining fraction of the path only the virtual and the interdomain links will be recognized.

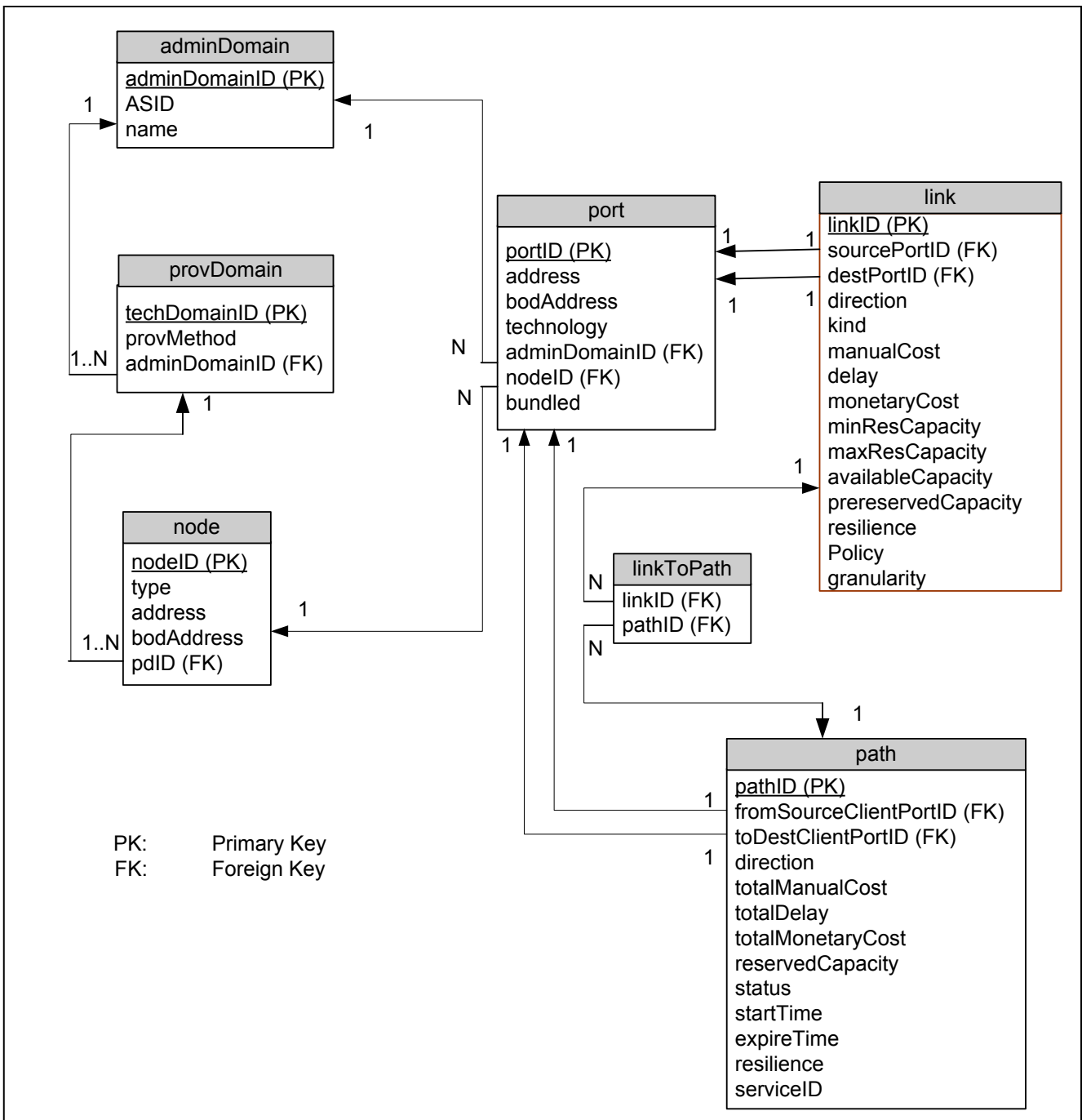


Figure 7: E-R diagram

4 Pathfinder

4.1 Overview

A set of autonomous control domains, which collaborate to provide a BoD service, need to establish an interdomain communication control channel to exchange BoD routing information. The main task of the Pathfinder module in the IDM is to provide a candidate path to fulfil BoD requests.

The pathfinder module receives as input from the Reservation module a set of parameters for each reservation request and returns to the Reservation module a set of possible inter-domain paths over which the request can potentially be implemented.

For the first phases of the IDM deployment, a decision has been made to use an OSPF-based signalling and routing protocol. This choice is preferred even though it requires that each domain receives detailed information on the whole BoD interdomain topology, because it benefits from the existing effort on Traffic Engineering extensions and the initial number of BoD domains does not yet create scaling issues.

The Pathfinder module applies a constrain-based algorithm to create a list of paths to be handed back to the Reservation module. Each path in the list represents an inter-domain route over a set of interconnected domains, and includes the ingress and egress interface in each transit domain.

For its operation and path-selection logic, the Pathfinder module requires routing and traffic engineering information, consisting of a network topology and a number of parameters for each inter-domain link.

The Traffic Engineering extensions are used for distributing the inter-domain routing information to be used by a separate module ('Routing Protocol module', see **Figure 8**) within the IDM. At the current early stages of the IDM module deployment, it is envisaged that the Routing Protocol module of the IDM will be realized using the Quagga OSPFv2 routing daemon implementation with custom defined Opaque LSAs to accommodate the parameters needed for inter-domain path finding. However, as the Quagga OSPFv2 daemon is a SPF (shortest Path First) engine and not a constraints-based SPF engine, the Pathfinder module is required to perform additional CSPF computations using the TE information disseminated by the OSPFv2 daemons of the inter-domain environment.

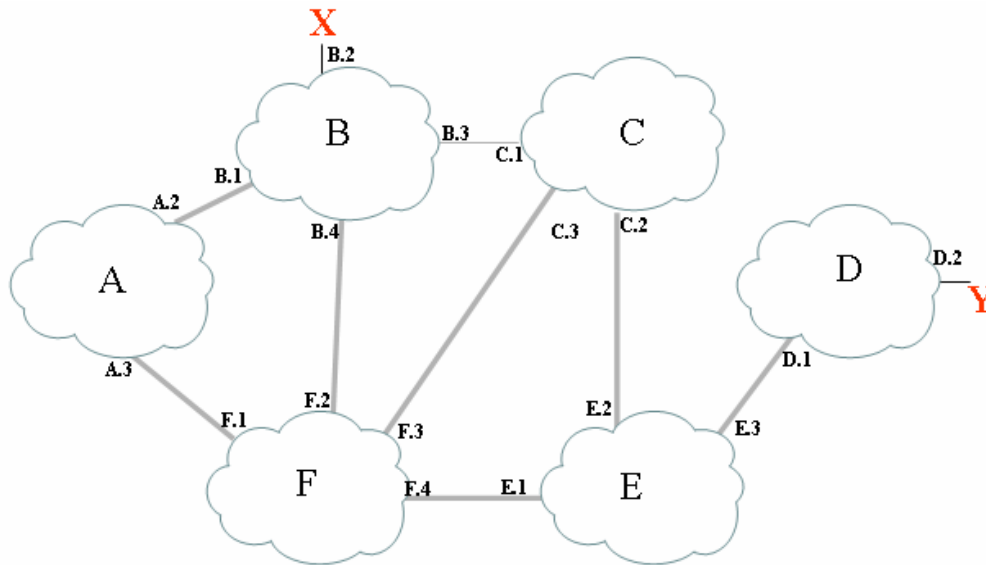


Figure 9: Reference case for inter-domain topology

The Pathfinder uses the topology information table resulting from the OSPF-TE advertisements received for its path calculations. The topology information table contains the most recent information on inter-domain path links as follows (Table 1 with some sample values):

Link	Reservable capacity			Resiliency	Delay	Policy
	min	max	granularity			
B.3 C.1	1 Gbps	1 Gbps	1 Gbps	1+1	5ms	(tbd)
A.2 B.1	1 Gbps	2.5 Gbps	VC-4	none	10ms	(tbd)
A.3 F.1	1 Gbps	10 Gbps	any	1+1	4ms	(tbd)
B.4 F.2	(tbd)
C.3 F.3	(tbd)
E.2 C.2	(tbd)
F.4 E.1	(tbd)
E.3 D.1	(tbd)

Table 1: Sample inter-domain routing table for the topology of Figure 9

The Pathfinder ‘prunes’ the topology table based on the input parameters of the reservation request and then conducts on the remaining links a full Path Finding algorithm to retrieve all paths between the requested reservation end-points. It then creates a list of paths in order of cost, optimizing on the chosen constraint (e.g. the physical Delay).

The format in which each resulting path is represented is proposed to be as follows, for the case of a path between the X & Y endpoints of Figure 9:

DomainB_ID — <B.3_IPv6_address> — <C.1_IPv6_address> — DomainC_ID — <C.2_IPv6_address> — <E.2_IPv6_address> — DomainE_ID — <E.3_IPv6_address> — <D.1_IPv6_address> — DomainD_ID

The resulting set of inter-domain feasible paths is returned from the Pathfinder module to the Reservation module, so that the reservation request can be further processed.

4.3 Future considerations

It is under study and experimental evaluation how the IDM Pathfinding module should act for the intra-domain parts of the inter-domain path. The design and implementation of the BoD service anticipates that internal topology and traffic engineering information of a domain may not be exposed to its neighbours for the purposes of the end-to-end service establishment. OSPF-TE is a valid candidate also for Intradomain path finding.

If the inter-domain path finding procedure may use also the properties of the intra-domain parts of the end-to-end path, e.g. the available capacity or resiliency between B.2-B.3, C.1-C.2, E.2-E.3 and D.1-D.2 in Figure 9, it might provide a better list of candidate path. Otherwise, if the IDM Pathfinder module uses only inter-domain links' information, the optimisation on the computation of the end-to-end path might be sub-optimal. A possible solution to this issue is that the DM announces towards the IDM and the Pathfinder module a set of virtual links (mesh) between the domain edge nodes together with the traffic engineering properties of those links, so that the links can be used by the IDM Pathfinder module in the computation of end-to-end paths. In that case, the Routing Protocol module can handle the condensed intra-domain routing information in the same way as that of inter-domain links and the resulting topology table (see Table 1) includes this information in the form of virtual links. The use of an abstraction for the intra-domain topology and routing information at the IDM level will not be delivered for the IDM prototype, but will be considered in later phases.

For the implementation and successful operation of the IDM Pathfinding module, it is important that in later phases of development the IDM-DM interface will provide a push method for the DM to update the abstract representation of the intra-domain topology in the IDM, as well as that of inter-domain links kept at the IDM, whenever a change in the physical topology of the underlying network occurs that affects the connectivity or traffic engineering attributes of a given link. Mapping and consistency between actual links, ports, devices and their representation at the abstract layer is in this case a key element to allow correct path-finding results.

5 User Access and Main Request Handling Logic

This block is responsible for controlling the requests and for managing the reservation process at inter-domain level. It can be considered as composed by two main parts: a “User Access” sub-block and a “Main Request Handling Logic” sub-block.

A request can be any of the following:

1. service request, which is submitted by a user and may contain multiple reservation requests (at least one must be present)
2. reservation request, which is submitted by a neighbouring IDM module

Users are allowed to submit service requests via the User Access sub-block. Each service request may contain multiple reservations. To have a simpler initial implementation, each request, even if composed by multiple reservations which will be realized initially as a single transaction, so that any single reservation failure causes the whole service request to be rolled back and rejected. In the next phases of the IDM the reservation handling will be more complex. The service logic includes the following steps:

1. get a single reservation from a service request,
2. find the inter-domain reservation paths list for that reservation,
3. choose a single inter-domain path (if any of the following steps fail, try another path),
4. retrieve local domain reservation constraints (e.g. VLAN numbers, capacity, etc.) and pre-reserve local resources (in the IDM prototype only capacity is considered),
5. forward the reservation request, including constraints, to next IDM on path (using a chain model, as described in 2.4),
6. await for reservation confirmation from the end domain along the path,

7. schedule in the local domain the reservation (release pre-reservation and provide resource allocation instead),
8. continue with first step until all reservations in the service request are scheduled or one fails.

In the case of a reservation request by a neighbour IDM, the domain, which started the request, has already computed the end-to-end path and the domain receiving the request can use the pre-computed path. In case of issues, the domain which received the request can perform steps 2-3 to re-compute its egress port and the path from itself to the end and use this computation to forward the reservation.

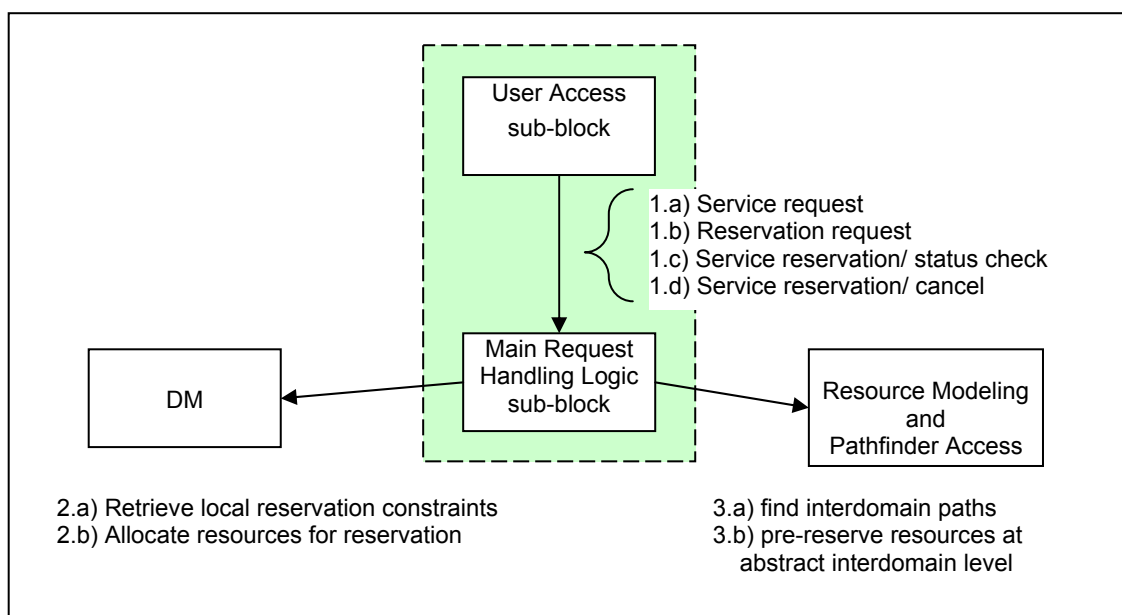


Figure 10: Main Request Handling Logic module (shaded box) interactions with other IDM blocks and the DM

A description of the possible interactions listed in Figure 10 follows:

- 1.a Service request – contains a user service request, with user information and reservations attributes
- 1.b Reservation request –reservation request from a neighbour IDM, with reservation attributes, including interdomain path
- 1.c Service reservation/ status check – request for a service or reservation status
- 1.d Service reservation/ cancel – request for cancellation of a service or reservation
- 2.a Retrieve local reservation constraints – request to Domain Manager to get local domain reservation constraints (those constraints depend on local domain technology, and may be associated with availability of VLAN numbers, SDH time-slots, delay, etc.)

- 2.b Allocate resources for reservation – request for resource allocation in order to make a scheduled reservation (this implies no reconfiguration of devices, but rather marking the resources to be used some time in the future)
- 3.a Find interdomain paths – request to find interdomain paths between two end-points of a reservation, taking into account the reservation attributes (including capacity, delay, etc.)
- 3.c Pre-reserve resources at abstract interdomain level – request for resources' pre-reservation (for the IDM prototype purpose, only capacity is considered), before the reservation request is sent to a neighbouring domain's IDM

5.1 User Access

The User Access sub-block is dedicated to the interaction between the BoD system and external entities, which may be users or neighbouring IDMs. The following requests are acceptable to the User Access sub-block, independently of which type of user accesses the BoD system:

- Service request,
- Service status request,
- Service cancellation request.

A service request must contain user information, including credentials for authorization, and at least one reservation attribute (multiple reservations are possible). For service status and cancellation requests, user information and a service identifier are sufficient. All requests are forwarded to Main Request Handling Logic sub-block, after authentication and authorization is granted by the AAI.

Toward or from a neighbouring IDM, the following requests are acceptable:

- Reservation request,
- Reservation status request,
- Reservation cancellation request,

A reservation request must contain sender information and credentials, reservation attributes and a set of reservation constraints from previously queried domains, if any. For reservation status and cancellation request, the sender information and reservation identifier are required. Reservation schedule report informs about the success or failure of a previously sent reservation request. It must contain sender information, reservation identifier, reservation result status, and a short message description in case of reservation failure. Also in this case all requests are forwarded to Main Request Handling Logic block, after authentication and authorization by the AAI block.

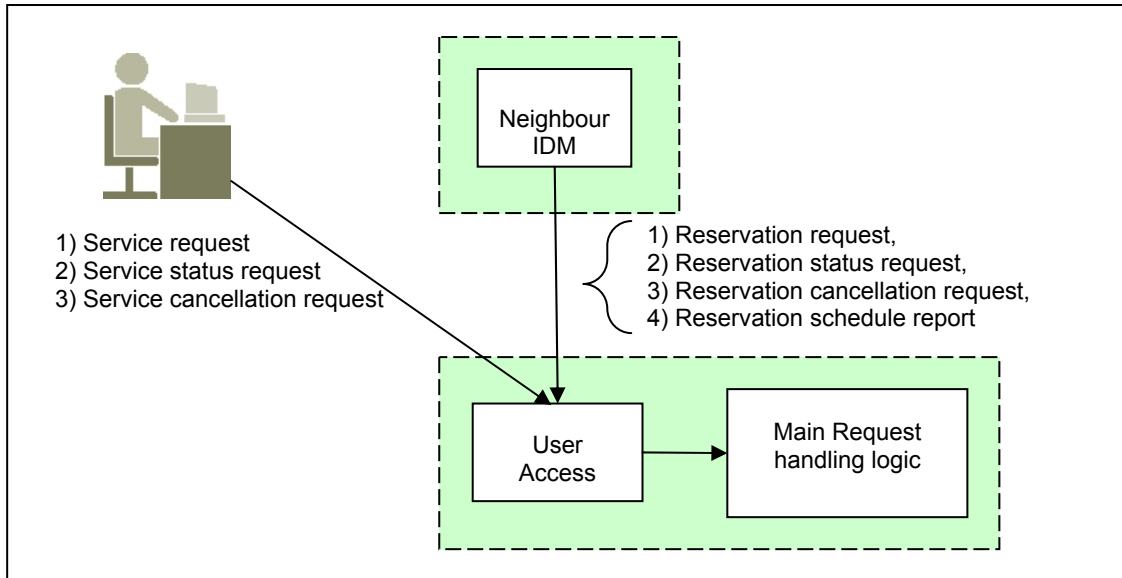


Figure 11: User Access sub-block interactions with external requestors.

6 Authentication and Authorisation Infrastructure (AAI)

The AAI module is responsible for the authentication and authorization of users and other BoD systems. For the authentication process, user or BoD system credentials will be required. Authorization will also require an action descriptor; the reservation system may use a list of action-based privileges for each user. For IDM prototype purposes, the AAI module functionality will be limited, and both functions will always result with success, independently of the user or BoD system credentials. For the IDM prototype, it is envisaged that a user or BoD system will be authenticated and authorized at request time.

No AAI check will be performed in the IDM prototype for communication between blocks and modules, including Pathfinder or DM access.

JRA3 is in the process of defining its AAI requirements at different levels of the system (user to BoD system AA, IDM-to-IDM AA, etc.). JRA3 is also working closely with JRA5, the 'Roaming and Authentication' activity of the GN2 project to define the necessary modules and interfaces so that the JRA3 BoD system can use the federated AA infrastructure designed and implemented by JRA5 (eduGAIN), for its multi-domain operations. The results of this work will be reflected in the design and implementation of the IDM Phase 1 release.

7 Domain Manager

The Domain Manager module is responsible for the management and configuration of the local domain, through the technology proxies, in order to implement in the network BoD service requests. For the operation of the IDM prototype the DM module will be a dummy module, which accepts requests and always reports a success.

As there are already proprietary domain manager implementations in the research network environment, its important to standardize the interface between the DM and the IDM to allow the BoD service system to use existing modules and services.

Prototype testing allows also analyzing the placement of some functionalities, like the service database and the network topology status, to be either in the DM or in the IDM. It may have a large impact on dataflow and system efficiency.

8 Flow Diagram

The **Figure 12**, provides the sequence of steps according to which the IDM prototype modules communicate during a reservation request processing phase

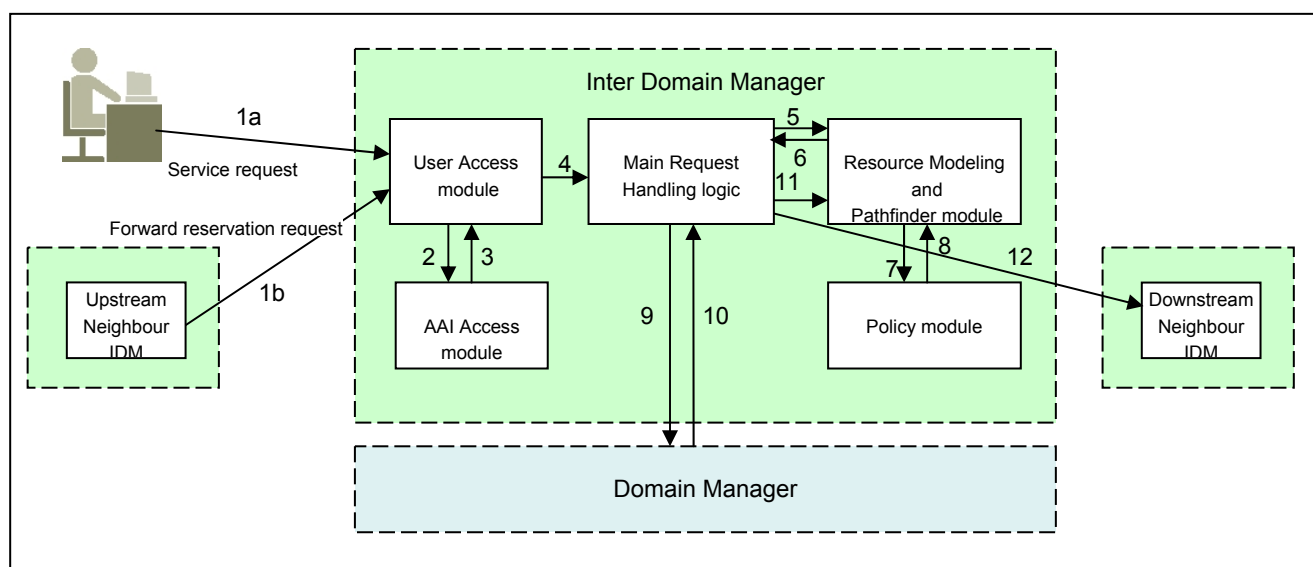


Figure 12: Flow diagram of a reservation request processing phase

The **Figure 13** provides the sequence of steps according to which the IDM prototype modules communicate during a reservation commit phase

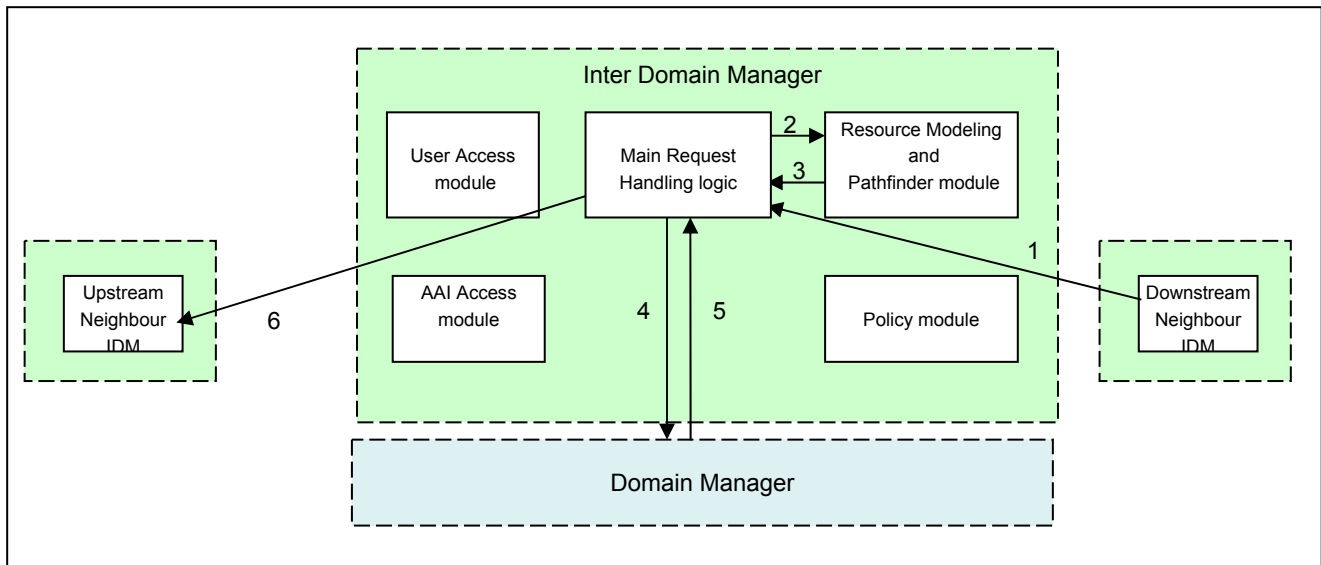


Figure 13: Flow diagram of a reservation commit phase

The detailed set of steps performed within each of the modules of the IDM prototype, are presented in the detailed Use Cases of the following section.

9 IDM prototype Use Cases

9.1 Use Case (UC) 1: User service request acceptance

Goal	Accept user request and prepare it for scheduling.
Actors	<ul style="list-style-type: none"> • User • User Access sub-block • Main Request Handling Logic sub-block • AAI access block
Pre-conditions	<ul style="list-style-type: none"> • User submits the request in an XML format (a file containing an XML description for prototype purposes, a web portal in later development stages) • User is addressing BoD system in his home domain • All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> 1. User connects to User Access and provides an XML specification of service request. 2. User Access reads user credentials from request and authenticates User with AAI module. 3. Once authentication succeeds, User Access authorizes User for service request submission. 4. Once authorization succeeds User Access assigns unique identifier to the service and forwards the service for execution to Main Request Handling Logic. 5. User Access returns service identifier to User.
Alternative flow	2a, 3a If User authentication or authorization fails, service is refused.
Post-conditions	<ul style="list-style-type: none"> • Service is accepted by system, and waits for resource analysis and scheduling.

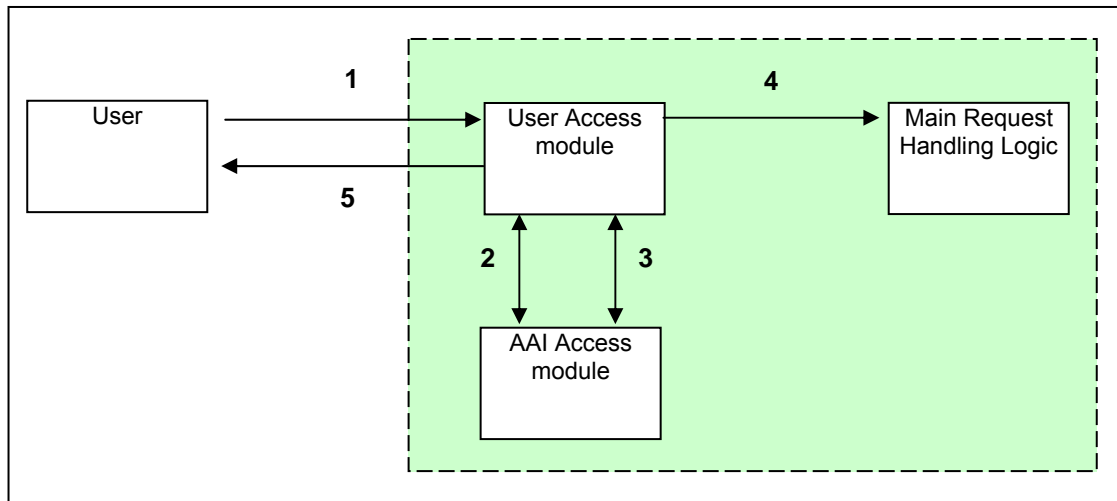


Figure 14: Flow diagram for case UC1

9.2 Use Case 2: User status request

Goal	Response to User with request status via e-mail.
Actors	<ul style="list-style-type: none"> • User • User Access sub-block • Main Request Handling Logic sub-block • AAI access block
Pre-conditions	<ul style="list-style-type: none"> • User has prepared correct XML representation of check status request (as XML file for prototype, and through web portal in later development stages) • User is addressing BoD system in his home domain • All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> 1. User connects to User Access and provides an XML specification of check status request. 2. User Access reads user credentials from request and authenticates User with AAI access. 3. Once authentication succeeds, User Access authorizes User for check service status operation. 4. Once authorization succeeds User Access contacts Main Request Handling Logic in order to retrieve service status (a check if User is service owner is required). 5. User Access sends retrieved status to User using e-mail address enclosed in request.
Alternative flow	<p>2a, 3a If User authentication or authorization fails, request is discarded.</p> <p>4a If no such service was found an error message is sent to user via e-mail</p> <p>4b If user is not a service owner an error message is sent to user via e-mail</p>

Post-conditions	<ul style="list-style-type: none"> User should get a service status.
-----------------	---

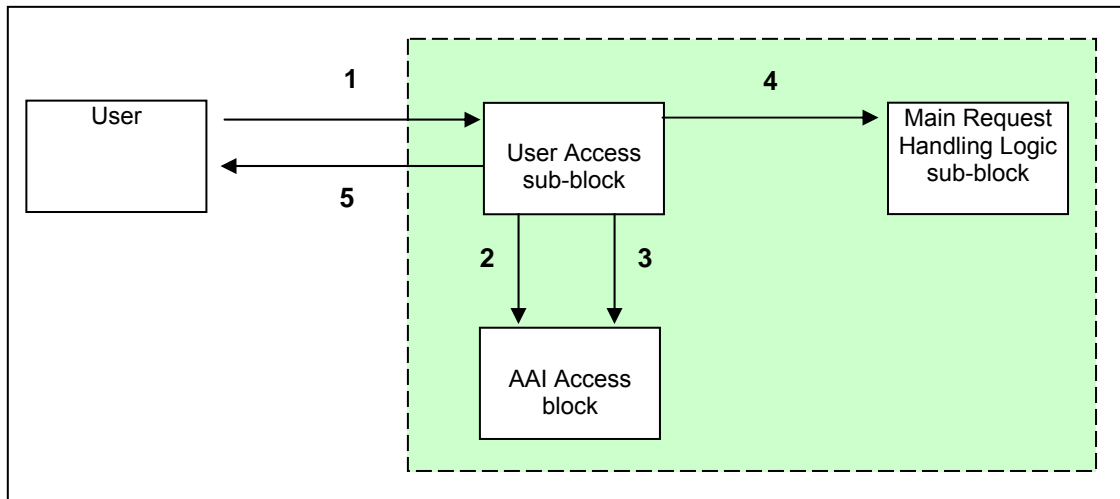


Figure 15: Flow diagram for case UC2

9.3 Use Case 3: User cancels service request

Goal	Cancel service submitted by user.
Actors	<ul style="list-style-type: none"> User User Access sub-block Main Request Handling Logic sub-block AAI access block
Pre-conditions	<ul style="list-style-type: none"> User has prepared correct XML representation of cancel request (as XML file for prototype, and through web portal in later development stages) User is addressing BoD system in his home domain All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> User connects to User Access and provides an XML specification of cancel request. User Access reads user credentials from request and authenticates User with AAI access. Once authentication succeeds, User Access authorizes User for check service status operation. Once authorization succeeds User Access contacts Main Request Handling Logic in order to cancel service (a check if User is service owner is required). Once service is cancelled, User Access sends message to User using e-mail address enclosed in request.
Alternative flow	2a, 3a If User authentication or authorization fails, request is discarded.

	<p>4a If no such service was found an error message is sent to user via e-mail</p> <p>4b If user is not a service owner an error message is sent to user via e-mail</p>
Post-conditions	<ul style="list-style-type: none"> Service is cancelled, all scheduled or pre-reserved resources are released, and all corresponding transactions are rolled out.

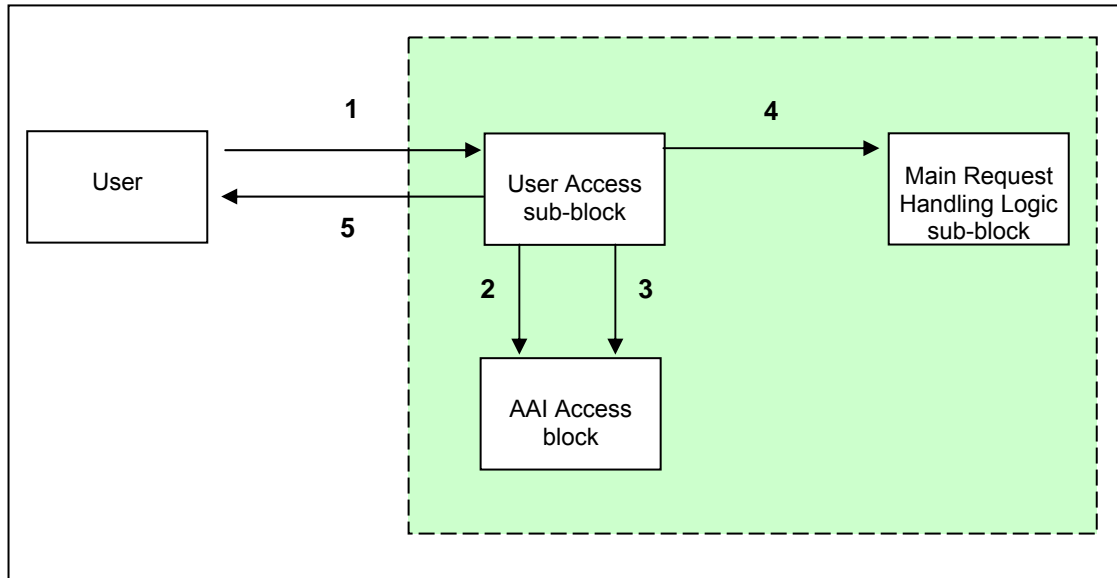


Figure 16: Flow diagram for case UC3

9.4 Use Case 4: Service schedule procedure

Goal	Schedule all reservations in service and allocate resources to be used in the future for all domains along reservation paths.
Actors	<ul style="list-style-type: none"> User Main Request Handling Logic sub-block Resource Modelling and Pathfinder Access block Domain Manager module AAI access block
Pre-conditions	<ul style="list-style-type: none"> Service is accepted and awaits schedule in request queue All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> Main Request Handling Logic sub-block starts scheduling of each reservation from service in sequential mode (end-start manner). Main Request Handling Logic sub-block searches for possible interdomain paths for reservation, with usage of Resource Modelling and Pathfinder Access block (a list is returned). Main Request Handling Logic sub-block checks first interdomain

	<p>path on the list, if there are enough interdomain resources (capacity) for reservation, reported by Resource Modelling and Pathfinder Access block (domain knows only the interdomain resources that he manages).</p> <ol style="list-style-type: none"> 4. Main Request Handling Logic sub-block pre-reserve interdomain resources (capacity) at Resource Modelling and Pathfinder Access block. 5. Main Request Handling Logic sub-block checks at Domain Manager module, local domain constraints (technology agnostic) which have to be agreed by all domains along reservation path. 6. Main Request Handling Logic sub-block forwards reservation request (including local constraints) to next domain along reservation path (messages are forwarded on chain basis). 7. Once confirmation of resource allocation (schedule) along the path arrives, Main Request Handling Logic sub-block authenticates sender (neighbour BoD system) via AAI access block. 8. Main Request Handling Logic sub-block confirms pre-reserved resources at Resource Modelling and Pathfinder Access block, and allocates technology agnostic constraints. 9. Once the Main Request Handling Logic module schedules all reservations, within the service, the User is notified via e-mail about the result.
Alternative flow	<ol style="list-style-type: none"> 2a If no paths could be found, the service fails and the User is notified about it via e-mail. 3a If no resources are available, another interdomain path from the list is chosen 3b If no resources are available, and there is no path left on the list, the service fails. Main Request Handling Logic sub-block needs to roll back all previously scheduled reservations within the current service. User is notified about it via e-mail. 5a If Domain Manager module reports that a reservation is impossible within the local domain, Main Request Handling Logic sub-block releases pre-reserved resources at Resource Modelling and Pathfinder Access block, chooses next interdomain path from the list, and follows processing from step 3 (in case there is no path left on the list, service fails, Main Request Handling Logic sub-block needs to roll back all previously scheduled reservations within current service and User is notified about it via e-mail). 7a If any domain along the path refuses to allocate resources, Main Request Handling Logic sub-block releases pre-reserved resources at Resource Modelling and Pathfinder Access block, chooses next interdomain path from the list, and follow processing from step 3 (in case there is no path left on the list, service fails; Main Request Handling Logic sub-block needs to roll back all previously scheduled reservations within current service and the User is notified about it via e-mail).
Post-conditions	<ul style="list-style-type: none"> • All reservations within the service have obtained the necessary resources in the future (in case of any failure, no reservations has allocated resources).

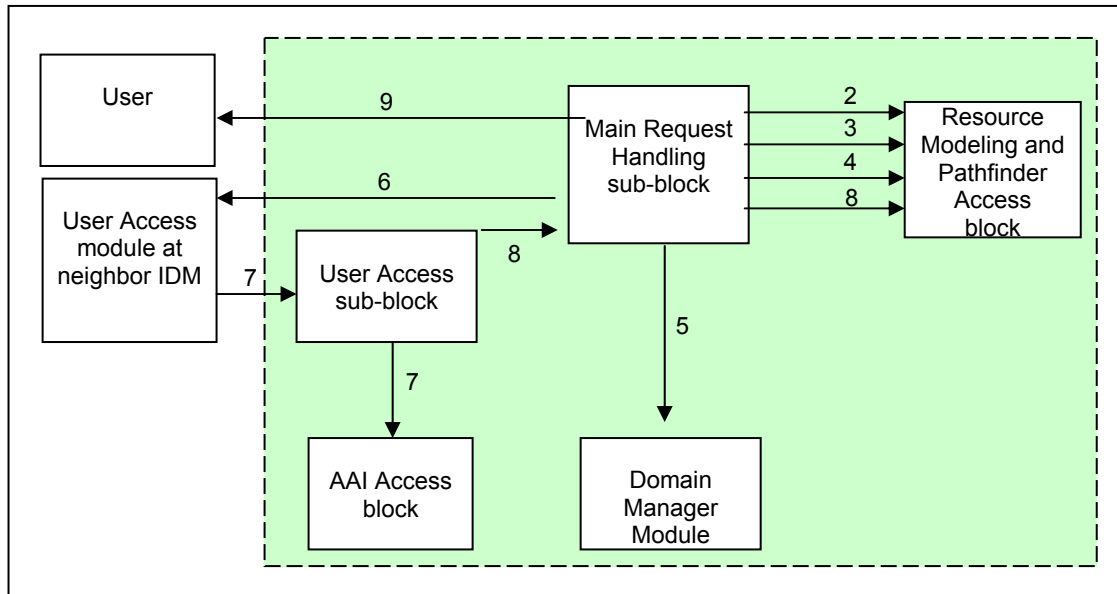


Figure 17: Flow diagram for case UC4

9.5 Use Case 5: Reservation scheduling procedure at a domain along the reservation path

Goal	Schedule reservation and allocate resources to be used in the future in local domain along the reservation path
Actors	<ul style="list-style-type: none"> • User Access sub-block • Main Request Handling Logic sub-block • Resource Modelling and Pathfinder Access block • Domain Manager module • AAI access block
Pre-conditions	<ul style="list-style-type: none"> • The reservation request sender is a neighbouring domain's BoD system • All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> 1. User Access reads neighbour BoD credentials from request and authenticates User with AAI access. 2. Once authentication succeeds, User Access authorizes User for service submission. 3. Once authorization succeeds User Access module forwards request to Main Request Handling Logic to schedule reservation. 4. Main Request Handling Logic checks, if there are enough interdomain resources (capacity) for reservation, reported by Resource Modelling and Pathfinder Access (domain knows only

	<p>the interdomain resources that he manages).</p> <ol style="list-style-type: none"> 5. Main Request Handling Logic pre-reserves interdomain resources (capacity) at Resource Modelling and Pathfinder Access. 6. Main Request Handling Logic checks at Domain Manager, local domain constraints (technology agnostic) which have to be agreed by all domains along the reservation path. 7. Main Request Handling Logic forwards reservation request (including local constraints) to next domain along reservation path (messages are forwarded on chain basis). 8. Once confirmation of resource allocation (schedule) along the path arrives, Main Request Handling Logic authenticates sender (neighbour BoD system) via AAI access. 9. Main Request Handling Logic confirms pre-reserved resources at Resource Modelling and Pathfinder Access, and allocates technology agnostic constraints. 10. Main Request Handling Logic responds to reservation request owner that reservation is scheduled at all domains from this one, to the final domain.
Alternative flow	<ol style="list-style-type: none"> 1a, 2a If authentication or authorization fails, the reservation request is discarded. 4a If no resources are available, Main Request Handling Logic responds immediately to the reservation request owner that the request failed. 6a If the Domain Manager reports that reservation is impossible within local domain, Main Request Handling Logic releases pre-reserved resources at Resource Modelling and Pathfinder Access, and responds immediately the to reservation request owner that the request failed. 8a If any domain along the path refuses to allocate resources, Main Request Handling Logic releases pre-reserved resources at Resource Modelling and Pathfinder Access, and responds immediately to the reservation request owner that request failed.
Post-conditions	<ul style="list-style-type: none"> • Reservation has allocated resources in the future for this domain and all domains from this one up to the last one along the reservation path.

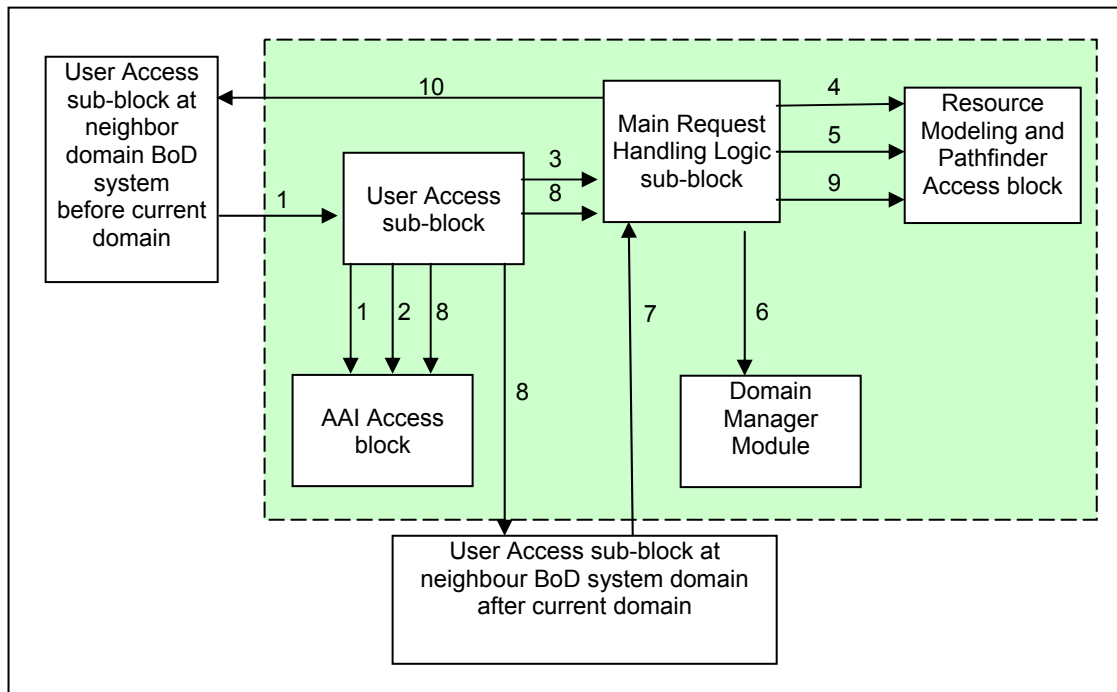


Figure 18: Flow diagram for case UC5

9.6 Use Case 6: Reservation scheduling procedure at the last domain along the reservation path

Goal	Schedule reservation and allocate resources to be used in the future in local domain along reservation path and define technology agnostic reservation constraints.
Actors	<ul style="list-style-type: none"> • User Access sub-block • Main Request Handling Logic sub-block • Resource Modelling and Pathfinder Access block • Domain Manager module • AAI access block
Pre-conditions	<ul style="list-style-type: none"> • The reservation request sender is the BoD system of a neighbour domain • Current domain is the last one on reservation path • All components of BoD system are up and ready
Basic flow	<ol style="list-style-type: none"> 1. User Access reads neighbour BoD system credentials from request and authenticates User with AAI access. 2. Once authentication succeeds, User Access authorizes User for service submission. 3. Once authorization succeeds User Access forwards request to Main Request Handling Logic to schedule reservation. 4. Main Request Handling Logic checks, if there are enough interdomain resources (capacity) for reservation, reported by the Resource Modelling and Pathfinder Access (the domain knows only the interdomain resources that he manages). 5. Main Request Handling Logic pre-reserves interdomain resources

	<p>(capacity) at Resource Modelling and Pathfinder Access.</p> <p>6. Main Request Handling Logic checks at Domain Manager, local domain constraints (technology agnostic) which have to be agreed by all domains along reservation path.</p> <p>7. Main Request Handling Logic analyzes all constraints collected along the reservation path, and selects strict constraint values to finalize reservation along domains.</p> <p>8. Main Request Handling Logic confirms pre-reserved resources at Resource Modelling and Pathfinder Access, and allocates technology agnostic constraints.</p> <p>9. Main Request Handling Logic responds to the reservation request owner that reservation is scheduled at the final domain.</p>
Alternative flow	<p>1a 2a If authentication or authorization fails, the reservation request is discarded.</p> <p>4a If no resources are available, Main Request Handling Logic responds immediately to reservation request owner that the request failed.</p> <p>6a If Domain Manager report that reservation is impossible within local domain, Main Request Handling Logic releases pre-reserved resources at Resource Modelling and Pathfinder Access, and responds immediately to reservation request owner that request failed.</p> <p>7a If the intersection of all the available parameters ranges received from each domain results in an empty set (i.e. there are no available parameters due to the constraints received), Main Request Handling Logic releases pre-reserved resources at Resource Modelling and Pathfinder Access, and responds immediately to reservation request owner that request failed.</p>
Post-conditions	<ul style="list-style-type: none"> Reservation has allocated resources in the future in the last domain along the reservation path.

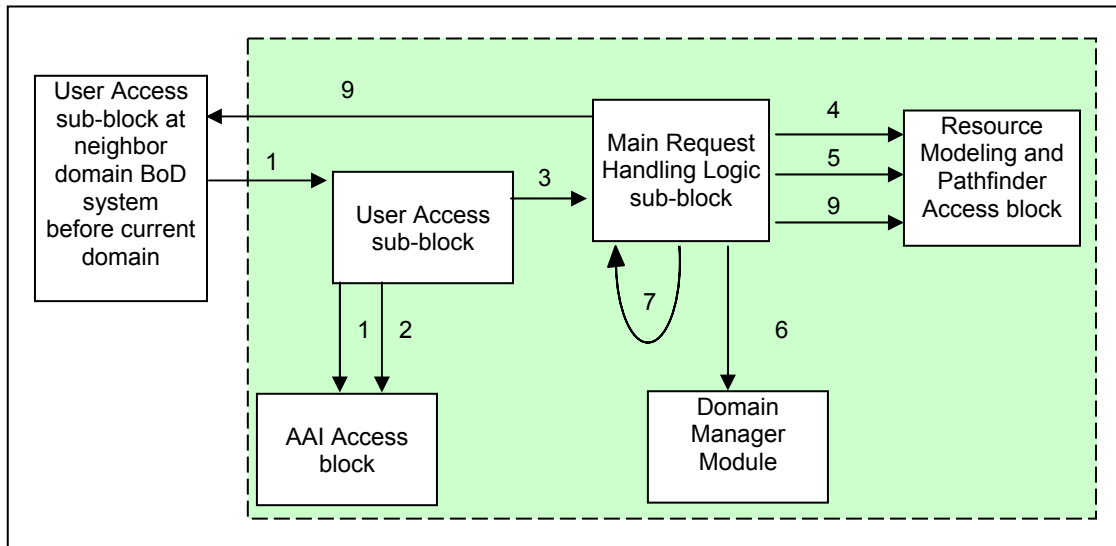


Figure 19: Flow diagram for case UC6

Project:	GN2
Deliverable Number:	DJ3.3.2
Date of Issue:	09/06/06
EC Contract No.:	511082
Document Code:	GN2-06-091v3

10 Implementation Notes

The prototype is focused on the study of the communication and processing logic of the BoD system. Therefore the “Main Request Handling Logic” and “User Access” module are developed in greater detail and with richer functionalities, while other modules are more limited, some of them being just dummy objects. To implement the prototype in such a way to allow reuse of its code in Phase1 and further, it is important to follow from the beginning the key engineering choices, which will be respected in development process:

- Java 1.5 is chosen as implementation language;
- WebServices are chosen for communication to the user and to other domains;
- the AAI system will be provided by an activity external to JRA3, the JRA5 activity (‘Roaming and Authentication’) of the GN2 project and proper interfaces will be built as a collaborative effort between JRA3 and JRA5

To better describe the implementation structure, Figure 20 shows the mapping between module names and corresponding Java packages. Each package name starts with *net.geant2.jra3*. The “User Access and Main Request Handling Logic” module is divided into three packages:

1. *communication* – involves user to IDM and IDM to IDM implementations
2. *interdomain* – involves logic required for inter-domain service processing
3. *reservation* – involves representation of a service request and a reservation

Resource Modelling and Pathfinder Access module are implemented into two packages:

1. *pathfinder.interdomain* – involves simulation of interdomain pathfinder, using predefined paths
2. *resources* – involves simulation of a database for capacity utilization on interdomain links.

The AAI module is mapped into a single *aai* package. In order to simulate final system functionality, two additional packages are planned. The *intradomain* package is devoted to the DM module emulation, using a predefined behaviour. The *network* package contains classes used for abstract inter-domain network representation.

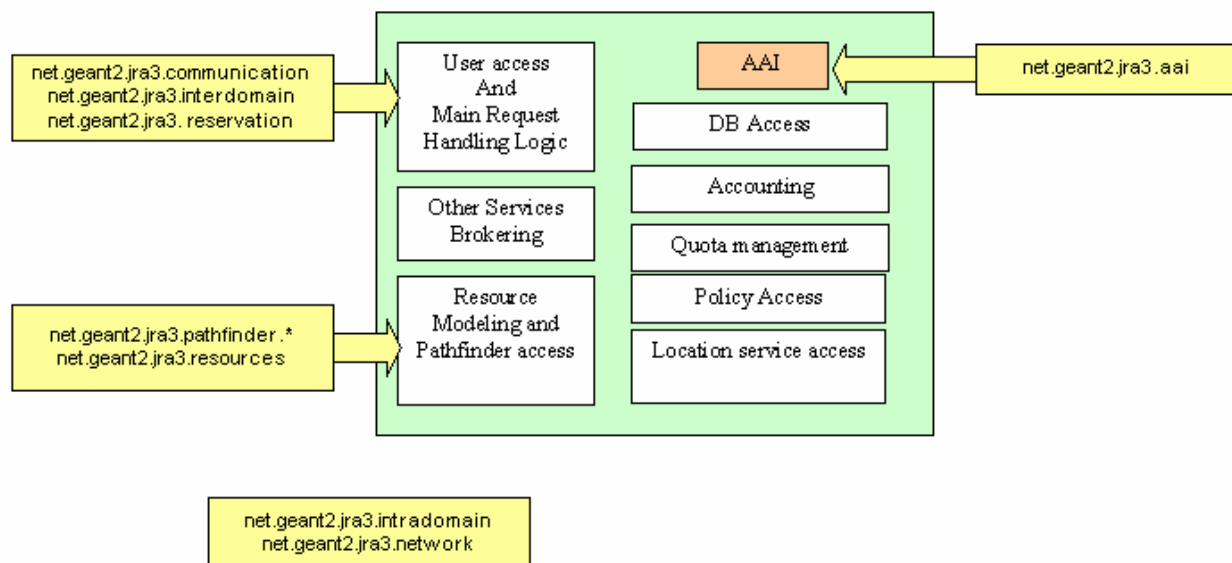


Figure 20: Modules to Java packages mapping

The following Table 2 describes the packages in details, stressing the differences between prototype and final system.

Package name	Implementation status
net.geant2.jra3.communication	Provides functionality similar to final system, regarding current status of user to system interface and IDM to IDM interface. Uses WebServices as communication protocol, secured with X.509 certificates.
net.geant2.jra3.aai	Simulates the AAI module behaviour. Its functionality is limited. Authentication and authorization never fails in the prototype.
net.geant2.jra3.interdomain	Contains the logic of inter-domain reservation, service management and interaction policy with neighbour BoD systems. Provides a functionality similar to the final system, offering a proof of concept for JRA3 BoD service assumptions. No database is provided for reservation storage.

net.geant2.jra3.reservation	Represents service request and reservation handling for internal application usage. It is implemented quite similar to the final version, but its contents are a function of the final definition of the data required for user and reservation description. Reservations are not stored into a database.
net.geant2.jra3.pathfinder.interdomain	Responsible for searching inter-domain paths, according to routing information. For prototype purposes, functionality is limited to provide predefined inter-domain paths, stored in XML files. This is a sub-package of the net.geant2.jra3.pathfinder.* package
net.geant2.jra3.resources	Responsible for controlling current resource usage for inter-domain links. Functionality is limited to tracking capacity usage only, in order to deny service if no capacity is available. The package will be re-designed after prototype tests, in order to provide the functionalities required in the next implementation phases.
net.geant2.jra3.intradomain	Contains logic of intra-domain reservations. Implemented as an emulator, with no logic. The purpose of these packages is to provide domain paths with constraints and accept local domain reservations. The functionality is implemented using static entries stored in XML files, which must be created before the prototype starts. No database is provided for reservation storage.
net.geant2.jra3.network	Contains entities used to represent the network for prototype purposes. Its implementation contains a subset of the abstract representation objects as described in section 3. The functionality is more limited in order to minimize the implementation effort and its complexity. No <i>node</i> object is present and <i>port</i> objects are directly associated with <i>adminDomain</i> objects.

Table 2: Objectives of Java packages and implementation status

11 References

- [DMTF] <http://www.dmtf.org>
- [DJ3.2.1] GEANT2 Deliverable DJ.3.2.1: BoD User and Application Survey.
- [DJ3.2.2] GEANT2 Deliverable DJ.3.2.2: Initial review of BoD related technologies.
- [DJ3.3.1] GEANT2 Deliverable DJ3.3.1: GEANT2 Bandwidth on Demand Framework and General Architecture, 20 Dec 2005, http://www.geant2.net/upload/pdf/GN2-05-208v7_DJ3-3-1_GEANT2_Initial_Bandwidth_on_Demand_Framework_and_Architecture.pdf
- [RFC 3945] “Generalized Multi-Protocol Label Switching (GMPLS) Architecture”, E. Mannie, IETF RFC 3945, October 2004.
- [quagga] <http://www.quagga.net>
- [UCLP] “User Controlled Lightpaths Architecture, Design, Progress”, R. Boutaba, <http://bcr2.uwaterloo.ca/~canarie/slides.pdf>
<http://www.uclp.ca>, User Controlled Light Path software home page for version 1 and 2
- [UCLP-API] <http://phi.badlab.crc.ca/uclp/documents/api/overview-summary.html>
- [webservices] <http://www.w3.org/2002/ws/>
- [xorp] <http://www.xorp.org/status.html>

12 Acronyms

AAA	Authorization, Authentication and Accounting
AAI	Authorization and Authentication Infrastructure
ASON	Automatically Switched Optical Network
BGP	Border Gateway Protocol, the widely used inter-domain routing protocol specified by IETF
BoD	Bandwidth on Demand
DB	Database
DM	Domain Manager
DMTF	Distributed Management Task Force
DNS	Domain Name System
e2e	End to End
FSC	Fiber Switch Capable
GMPLS	Generalized Multi-Protocol Label Switching
IDM	Inter Domain Manager
IETF	Internet Engineering Task Force - http://www.ietf.org
L2SC	Layer-2 Switch Capable
LSA	(OSPF) Link-State Advertisement
LSC	Lambda Switch Capable
MPLS	Multi-Protocol Label Switching
NMS	Network Management System
NOC	Network Operation Centre
NNI	Network to Network Interface
NTP	Network Time Protocol
NREN	National Research and Education Network
OSPF	Open Shortest Path First
PIP	Premium IP
PSC	Packet-Switch Capable
QoS	Quality of Service
RFC	Request for Comments
RSVP	ReSource ReserVation protocol
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SLS	Service Level Specification

Project:	GN2
Deliverable Number:	DJ3.3.2
Date of Issue:	09/06/06
EC Contract No.:	511082
Document Code:	GN2-06-091v3

SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Network
SPF	Shortest Path First
TDM	Time-Division Multiplexing
TE	Traffic Engineering
TL1	Transaction Language 1 – a widely used management protocol in telecommunications.
UCLP	User Controlled Light Paths
UDDI	Universal Description, Discovery and Integration protocol
UNI	User to Network Interface
URL	Uniform Resource Locator
VPN	Virtual Private Network