Peer-To-Peer Requirements On The Open Grid Services Architecture Framework

Status of This Memo

This memo provides information to the Grid community regarding the use of the OGSA framework for peer-to-peer applications. Distribution is unlimited.

Abstract

As the next generation of grid computing protocols centered around web services and OGSA are developed, the peer-to-peer community must determine how these protocols can be used for building peer-to-peer applications. Such applicability is not immediately obvious since peer-to-peer systems have significantly different properties than traditional server-based grid systems. For example, peer-to-peer systems exhibit different models of security and trust (machine user is typically also the administrator), have different connectivity characteristics (machine IP addresses may change due to mobility or firewalls/nats), and different usage models (instant messaging, file sharing, and collaboration). Because of these differences, peer-to-peer systems are likely to require different base services than server-based grid systems. Yet despite these differences, the shear numbers of desktop systems available today make the potential advantages of interoperability between desktops and servers into a single grid system quite compelling. The focus of this document is on developing a set of requirements for OGSA to be used to build peer-to-peer applications

Contents

1. Introduction: The need for a new global infrastructure

It is expected that in future ubiquitous networks that connection and interaction between devices, systems, services, people, and organizations will be the rule rather than the exception. The realization of the full potential of all conceivable patterns of interaction and collaboration will require a sophisticated global infrastructure on top of which service providers can develop their applications.

However, today's infrastructure is rudimentary, making the development of new services both difficult and expensive. A new global infrastructure is needed to handle the sheer complexity of new and varied models of interaction and collaboration. This is generally recognized by the scientific and industrial communities.

A systematic attempt to develop the framework for this global infrastructure is in progress within the GGF (global grid forum) based on the OGSA (open grid services architecture). OGSA is a refactoring of the existing Globus Toolkit Version 2.0, also known as GT2 that has achieved significant adoption within the academic and high-performance computing communities. The focus of the Globus Toolkit is to enable the creation of Virtual Communities; a set of organizations with shared interests and motivated to share a portion of their resources (CPU, disk, data, etc). GT2 defined a set of services and protocols that enabled organizations to share these resources in a secure manageable way.

<why wasn't GT2 sufficient>

Based on this refactoring, the next release of the Globus Toolkit, named GT3, will leverage open protocol standards such as XML, SOAP, and WSDL. It will be based on a services framework similar to web services and in fact uses many of the same technologies being developed within that context (WS-Security, XML-encryption, etc).

<overview of why p2p computing is appealing>

1.1 Peer-to-peer computing

The peer-to-peer computing community has and is addressing some of the important issues for the new global infrastructure.

In the peer-to-peer community much attention has been focused on collaboration. The applications cater for dynamic and fluctuating usage and connection. Groups may vary in size, and in particular may become very large, as is the case with peer-to-peer file-sharing systems (e.g. Kazaa, Gnutella) and cycle-stealing computation sharing (e.g. SETI). Groups may be long-lived, composed of friends, colleagues, and organizations or everybody. Groups may also be very short-lived and spontaneous when they are groups of people in the same place at the same time.

The peer-to-peer community has also addressed computation sharing but there the focus has been on cycle-stealing and sharing on end-user PCs. Finally the peer-to-peer community in the area of security is considering security mechanisms that are not based upon authorization. Instead of a static division between authorized and unauthorized users, the system monitors, collects, and correlates the behaviors of users and ensures that properly behaving users gradually acquire more privileges and misbehaving users lose theirs.

One important result in peer-to-peer is in the area of self-organizing virtual networks that connect groups for all types of collaboration. A key issue is naming services; in some of the most interesting P2P work reliable naming services are decentralized for scalability and to avoid third

party dependencies. The naming service needs to be able cope with the ever-changing virtual network and in particular with the mobility of users and devices

1.2 This document

The purpose of this document and the OGSA-P2P research group is to provide input to the OGSA working and research groups. The input will be based on the results and experience of the peer-to-peer computing community. The ultimate goals are to

1. ensure that the OGSA architecture is general enough to provide support for peer-to-peer type application building
2. ensure that relevant results of the peer-to-peer computing are incorporated in the OGSA standards, best practices, etc.
3. provide guidelines to OGSA implementers based on the peer-to-peer experience
4. make the OGSA community aware of fundamental tradeoffs and remaining open research issues in peer-to-peer computing that will also impact OGSA

2. Areas of work

2.1 Scalability

It's clear that one of the strengths of the peer-to-peer systems is their focus on scale: scale in terms of the number of resources, the number of users, and the number of administrative domains covered. When dealing with hundreds of thousands to millions of entities, different design decisions will be made. Architectures will favor decentralization, automatic bootstrapping, and best effort as means to provide continued service in a highly dynamic environment.

2.2 Connectivity

The "peers" in peer-to-peer systems are typically home computers running in a complex network environment. Internet service providers may block certain types of network traffic, network address translators may hide resources behind a common name, even the network address of a peer may change quite frequently. The connectivity requirements explore these issues in greater detail.

2.3 Security

Security on a traditional server-based grid typically assumes strong user identities, secure machines locked away in machine rooms, and trusted and knowledgeable administrators. Peer-to-peer systems, in contrast, have operated in environments where these assumptions are false and have had to develop alternative mechanisms for achieving security, including community-based trust (user ratings) and replication and verification. The differences are not just as functions of the environments, even the goals of the systems are different and in some ways opposite: server-based grids focus on accountability and auditability while some peer-to-peer systems focus on anonymity and user privacy. Reconciling these seemingly opposite goals seems challenging.

2.4 Failure

The requirements with respect to failures are to some degree a function of the scale and security models of peer-to-peer systems. However, we felt that despite the overlap, there are some fundamental architectural differences in dealing with failures that should be explored in its own right. For example, the use of service replicas and the ability to failover from one replica to another; consideration of alternative failure models over the simple failstop model; and <other issues?> We explore this issues in the failure requirements section below.

2.5 Location Awareness

2.6 Group Support

Peer-to-peer collaborative systems allow for the dynamic creation and management of ad-hoc groups. For example, file sharing systems allows a group of families and friends to share photos, music and even calendars. The groups are created and managed easily without requiring permission or authority from any system administrators. Access can be given and revoked purely as a local decision. Such an ability goes to the core of collaboration aspects of peer-to-peer. We explore these requirements below.

3. Application Use Cases

Peer-to-peer applications need to be categorized and analyzed to determine how well the kind of support that these applications need can be met by the OGSA architecture.

3.1 PC Grid Computing

3.2 File Sharing

3.3 P2P Content Delivery

4. Scale

One important difference between traditional grid computing and peer-to-peer computing is in the scale of the resources: both the sheer number of resources and the geographic distribution of those resources. Peer-to-peer applications are scaling to hundreds of thousands if not millions of users (SETI@HOME, instant messaging, etc) spread across the globe. To deal with this scale, P2P systems have successfully employed decentralized architectures (eg gnutella) that are capable of surviving in a very dynamic environment. In this section, we examine what “dynamic environment” means and develop some requirements for OGSA.

5. Connectivity

[Administrative Note: this is the first section that will be updated during the two weeks, March 17-28. Please add content as you see fit, but keep change tracking on and post your changes to the mailing list].

A driving premise behind peer-to-peer computing is the access of computational resources, data, and services at network edges in a transparent manner to applications and users while lowering the total cost of ownership and participation. To permit decentralized sharing of computing resources, collaborative workspaces, information and services, it is necessary for the peers at the edge of the network to communicate with one another and with the services at the heart of the networ. Therefore ensuring communication is possible in a bidirectional and transparent end-to-end manner is critical to the success of peer-to-peer computing over and between public and private networks.

Over the past few years, security, resource availability and cost issues have required the widespread deployment of mechanisms that impede communication between peers on the internet – that is, reduce end-to-end network transparency. Today it is arguable that nearly every user's access to public Internet web services and information must pass through one or more of these devices. These mechanisms, known as *Network Address Translators (NATs)* and *firewalls*, often accommodate client-initiated client-server usage models, but they prevent transparent, bi-directional peer-to-peer communication.

Furthermore, the edge of the network may have fundamentally different characteristics than traditional server-based environments. For example, data communication over wireless phone networks may have a significantly different cost model that should be considered before sending arbitrary data. Caching and replication policies may also have to be adjusted to take into account different available bandwidth capability.

In this section, we examine the issue of connectivity in greater depth and derive some requirements that need to be met for grid services to be used as foundation for peer-to-peer applications.

5.1 Network Address Translators (NATs)

A NAT typically resides at network domain boundaries and translates network addresses from one network domain to another. NATs are most often used to share limited IP public network addresses and/or to translate addresses between two networks with incompatible address domains. Two such domains are the *public IP address domain* and the *private IP address domain*. Addresses in the public IP address domain must be assigned through an Internet Assigned Numbers Authority (IANA) and are routable and guaranteed unique from any connected component of the Internet. IANA has also reserved three blocks of IP addresses for private networks. Addresses within these blocks are routable and unique only within the scope of the private network in which it belongs.

Hence, to route packets from one domain to another requires the use of a NAT that has two interfaces, one in each of the address domains that it maps between. Figure 1 shows a common scenario where NATs are used. The ISP allocates one public IP address for use (132.235.95.202). A NAT runs at this address and in addition has an address in a private network (192.168.1.1). When a client in the private network (say 192.168.1.102) initiates a connection to a server in the public network (www.globus.org), those packets are routed through the NAT. The NAT rewrites the source address in the packet headers from 192.168.1.102 to 132.235.95.202 and stores the mapping in memory. When the server receives the connection request packet, it replies to the source address, now 132.235.95.202. The reply packet is received by the NAT that provides a reverse mapping using the in-memory table and rewrites the

destination address of the reply packet as 192.168.1.102 that is routed in the private network to the correct client.

Note that because the NAT modifies the packets in route to the destination, security mechanisms used to detect whether packets have been tampered with and that include the header addresses will in general fail. IPsec is one such protocol. It carries a checksum computed over the entire header of the packet. Hence, modifications of the source address will cause the packet to fail the security check.

1.a	No use security protocols that create checksums that include the header.

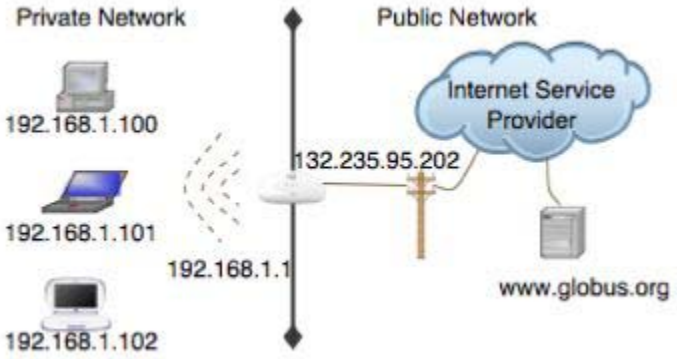

Figure 1: A common setup using a NAT to bridge a private network with a public (ISP) network.

There are many different types of NATs, including a traditional or outbound NAT (described above), network address and port translators (NAPTs), Bi-directional or two-way NATs, and twice NATs. Except for Bi-directional NATs, the basic problem with NATs is that they do not allow connection establishment from the external network into the private network, and hence block bi-directional communication capability.

The use of a NAT imposes several difficulties for Grid Services in particular. In the example in Figure 1, assume that two grid services are running, Grid Service **A** on 192.168.1.102 in the private network and Grid Service **B** running on www.globus.org in the public network. When **A** starts and registers itself in the registry (not shown in the picture but located in the public network) it registers its WSDL and grid service handle. While the NAT has rewritten the TCP packets involved in the registration process, the NAT only modifies the headers while the grid service handle is in the payload of the packet and is not modified. Therefore, when registering with a registry outside of the private domain, grid service **A** may register itself with the incorrect (or inaccessible) handle.

In general, whenever a WSDL document traverses a network domain boundary, its content must be rewritten so that the endpoints remain valid in the external domain.

1.b	A WSDL document that traverses a network domain boundary must be rewritten so that the endpoints remain valid in the external domain: ALG-GS.

This is a common problem for various protocols and there are some existing techniques that can accomplish this. A NAT can be configured with various optional *Application Level Gateways* (ALGs). Common ALGs include ALG-DNS for domain name information and ALG-FTP for dealing with the FTP protocol.

Even after registration, the external grid client will not be able to initiate communication with the grid service. The NAT requires the communication to be initiated by the host inside the private network. So therefore, grid service **B** acting as a client will not be able to communicate with grid service **A**. Furthermore, grid service **A** will be restricted even as a client: third party transfers (such as with gridftp) and delivery of notifications will not be possible. Consider an example in Figure 2. Grid Service **A** requests a file from grid service **B** that in turn uses a third-party grid service **C** to perform the transfer. In such a case, a traditional outbound NAT considers the communication from C to A as an initial connection request and will not forward the packets to grid service **A**.

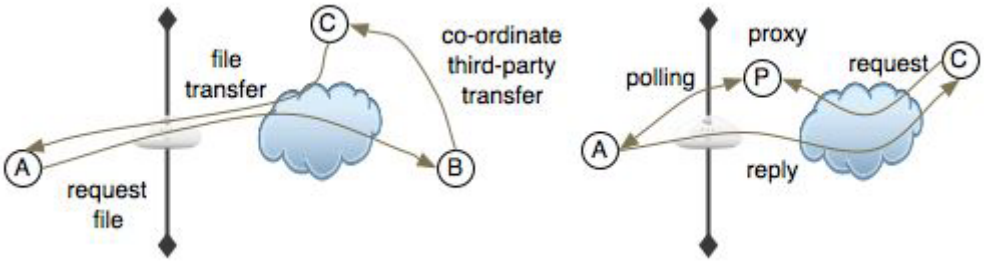

Figure 2: Example of a third-party transfer.

One typical solution to this asymmetry is to use rendezvous servers or simple relay mechanisms and have the host in the private network poll for messages to these intermediary services. The grid services infrastructure should support these solutions as well. Figure 2 shows an example of such a solution. A general grid service proxy runs in the public address domain and acts as a collection point for SOAP messages sent to grid service **A**. Grid service **A** periodically polls the proxy and retrieves the messages and responds as appropriate.

With this solution, the following capabilities are necessary:

1.c	Support for proxy forwarding.
1.c.1	Automatic tooling of grid service A to poll for messages at its proxies
1.c.2	An endpoint proxy should be constructed and store SOAP messages until the grid service can collect the messages.
1.c.3	Grid service clients should be able to tell if the service it is using is behind a NAT.

The use of these intermediaries, however, is neither ideal nor complete. If both the grid service and the grid client are behind NATs, then the store and forward solutions can become complex and unscalable. In addition, multiple levels of NATs are becoming common as ISPs are beginning to use NATs for their domains as well. This is not a problem specific to grid services. This is a general network connectivity problem present for many application frameworks. There exist many developing standards that try to address this issue, from the Universal Plug and Play (UPnP) to the IETF. As solutions and protocols are developed, the OGSA framework should support industry solutions.

5.1.1 Universal Plug and Play (UPnP)

In October 1999, the UPnP forum created the Internet Gateway Devices working group to address this problem for the home/SOHO environments. Briefly, the protocols developed will allow NATs to be detected by clients within the private network, and allow port forwarding to be programmatically configured by applications. Thus UPnP will require application modifications to be effective.

1.d	Support for UPnP detection and configuration.

5.1.2 IETF Middlebox

<should we look more carefully at these protocols?>

1.e	Support for middlebox (IETF) solutions to the NAT problem.

5.2 Firewalls

Firewalls are security devices that selectively filter (pass or drop packets) inbound and outbound network traffic based on site rules and policies. Firewalls employ three basic techniques: packet filtering, proxy services and stateful-inspection. A *packet-filtering* firewall examines network traffic passing through it and selectively passes or drops the packets. *Proxy service* firewalls also examine and filter packets but also act as a relay between hosts behind the firewall and the external network domain. Therefore, a proxy service firewall must perform address translation functions similar to that of a NAT. Finally, *stateful-inspection* firewalls monitor communication protocol and application-level state and develop a detailed context that is then used to more carefully inspect the communication protocol stack, enforce security policies, and anticipate protocol communication actions and requirements.

Typical firewall configurations allow http-based packets, that is, packets destined to or from port 80, but may drop packets destined to other ports. *Http-tunneling* refers to the use of http ports for non-http-based applications. Because the firewalls are typically configured to allow such packets through, the applications simply encode the application-specific protocols as http requests and replies.

1.f	Support for HTTP Tunneling.

<other issues in dealing with firewalls?>

5.3 DHCP

Another issue common in peer-to-peer environments is lack of fixed IP address assignment. In many environments IP addresses are assigned by DHCP (Dynamic Host Configuration Protocol) servers or may be provided by the NAT directly. When a machine boots up or is connected to the network, the machine broadcasts a IP address request to the DHCP server that assigns a IP address and possibly a resolvable hostname to the machine for a given reservation period. After the reservation period has expired, the machine will renew its request. The renewed address is not guaranteed to be the same as before.

In peer-to-peer environments there is evidence to suggest that this is not a trivial or limited problem. In [1] the authors examination of IP address aliasing finds that

- 40% of hosts use more than 1 IP address in the span of a single day,

- 32% of hosts use 5 or more IP addresses (over the course of the 15 days that they measured).

If addresses are changing on a daily basis, it's not enough to use soft-state mechanism (such as time-out and re-registration). Some mechanisms need to be provided to reroute messages as IP addresses change.

1.g	Support for IP address changes.

5.4 IP Address Mobility

IP addresses may change also due to physical movement of the machine from one network to another. This is the case when a laptop user moves from one place to another. In many corporate environments, the use of laptops far outpaces the use of personal desktop machines and therefore must be supported.

The requirement to support mobile systems is the same as in the previous section: no new requirements are needed. It should be noted however, that IP address mobility is not the same as MobileIP which is a particular solution to the problem of mobile users but operates at the IP layer.

5.5 Network characteristics and peer profiles

Peers will be running in all sorts of different computers/devices and it is not known ahead of time what kind of characteristics they would have. They could certainly have smaller resources available or if not smaller, maybe the cost of using them is higher. In the same area, network characteristics could be very different from one peer to another peer. Latency, bandwidth, reliability, cost, are some examples of possible factors that need to be considered when connecting two peers together and before they agree to start exchanging information. For example:

- Two peers that want to communicate have very distinct bandwidth to access the Internet. (Cell phone vs. PC using DSL)
- Some peers will have limited resources (i.e. no storage or very limited, like a PDA)
- Some peers may not even have a hard disk
- For some peers, reception cost might be higher (cell phone receiving a big file)
- Some networks are less reliable
- Some peers need to transfer data faster

These factors should be considered when a peer participates in grid, and it will very likely be related to QoS, SLA and policies.

1.h	Should be possible to determine the characteristics of a peer and its environment, and use it as a basis to establish the communication attributes and quality of the service.

6. Security

[Administrative Note: will be updated during the two weeks, March 31-April 13. Please add content as you see fit, but keep change tracking on and post your changes to the mailing list].

Peer to peer systems also bring a set of unique security requirements that stem for the fact that peer-to-peer systems must deal with looser notions of trust than server-based grid environments. In the server environments, there is an assumption of trust in the actions of the administrators and between the administrative domains. It is assumed that the host certificates are valid and secure; that the users have been validated before user certificates are issued; and that the software functions as advertised. The challenge for building peer-to-peer systems is that none of these assumptions hold.

6.1 Trust in identities

In server-based environments, users are assigned identities that are tied to their employment at the organization. The validity of the accounts is (usually) determined by direct interaction and an effort is made to ensure that exactly one account is created for exactly one user. Trust in this setting is given to users because accountability is provided by the system. Malicious users can be traced to their person and can affect their access, their employment, or in some cases even their liberty.

When the system scales beyond the capability to personally verify a user, such as a web-based community, alternative mechanisms for allocating trust have been developed. As an example, consider Ebay. A single user may have multiple accounts and the existence of an account does *not* immediately convey any trust. Instead, *community-based trust* has developed that allows every other user in the system to rate the trustworthiness of every other individual in the system. While a single malicious user may impugn a trustworthy person, the law of large numbers ensures that the averages will bear out the trust. Using this mechanism, trust is earned based on the number of transactions and the quality of the results of the transactions. There is some accountability in these systems (credit card verification), however, in many other communities, the only punishment for malicious users is expulsion from the community. Even this is not guaranteed since the user may just create a new identity.

In peer-to-peer environments it is necessary that community based trust does not depend on a centralized server to manage peers' reputation. It is also important that the absence of a reputation system does not prevent peers from interacting with each other, if trust is not required. However, when trust is required, and it is based on peer's reputation, it is necessary that the reputation system be available all the time, thus making a explicit requirement for reputation being part of the peer-to-peer infrastructure.

A reputation system consists mainly of three parts (i) the peer that performs an action and gets rated (2.a.1), (ii) the peers that use some other peer's rating information to decide if they would like to interact with such a peer or not (2.a.2), and (iii) the infrastructure or network used to distribute peer's reputation.

The reputation model depends heavily the infrastructure that is used to distribute the reputation information through the network. If the reputation model is embedded in the communication protocols used to connect peers, then reputation will be part of the infrastructure and easier to include and use in peer-to-peer networks (2.a.3).

It is important to distinguish the type of peers that is being rated because, we could be rating devices, applications, or users. For example, we need to distinguish for a particular peer which starts flooding the network, if it is because the device has gone wrong, the application is intended to do that or the user is doing an attack.

<I don't think peer classification is covered in the taxonomy paper – we might need more here>

If a reputation model for trust is used, it is necessary to have a classification of peers/users and distinguish for what type of behavior the reputation is being built.

Behavior for which peer trust can be built:
- Peer's network usage
- Peer's bandwidth
- Peer's responsiveness
- Peer's quality of responses
- Peer's overall presence
- Peer's reputation as perceived from other peers
- Peer's reputation based on existence
- more...

2.a	Community-based trust
2.a.1	Allow users to rate other users' trust as a function of actions.
2.a.2	Allow users to query the aggregate level of trust as a function of actions.
2.a.3	Support for reputation networks or other type of non-centralized reputation infrastructure.
2.a.4	A peer can be bound for specific behaviors to be tracked and rated, depending on its nature and classification.

Establishment of identities is another area where there may be some unique requirements, although some of these issues are coming to a head even in existing server-based grids. The basic issue is centralized versus decentralized creation of user identities. Centralized mechanisms don't scale as the number of organizations increases and requires the user to manage multiple identities (one for every grid). For example, some grid system includes a centralized certificate authority that provides user certificates to each user of the grid. So in addition to the local organization username and password, the user must maintain the grid username and password. The more grid systems the user is associated with, the more identities he or she must manage. A decentralized approach requires each individual organization that participates in the grid to issue its own user credentials (for example, by running its own certificate authority). The grid resources then are configured to accept certificates signed by each of the participating certificate authorities. This decentralized approach has the advantage of allowing each organization to independently assign user identities as they see fit and is simpler for the end-user. As systems scale in the number of organizations, decentralized identity establishment will become increasingly important.

2.b	Decentralized identity establishment

A third issue related to user identities is anonymity. While access to grid resources requires user verification and authorization, there are some applications where it is important that the users are anonymous. As an example, consider a local government grid and the application of voting. It is key that each user only is able to vote once and only if they are authorized to do so. But it is equally important that the application cannot in any way determine which users issued which votes.

2.c	User identity anonymity

2.c.1	A user should be able to shield his/her identity from the service, if desired, when using the service.
2.c.2	An anonymous peer should be able to use specific services by being member of an identified and trusted peer-group (including “no-group”).

6.2 Trust in resources

The basic notions that trust is a binary property – either the entitiy is trusted or it is not – must be examined carefully in the grid context. As grid systems scale in the number of organizations, it becomes harder to validate the level of trust that users should ascribe to resources. With a few organizations, trust is relatively simple: an agreement is made between the career system administrators at the different organizations as to the security policies that each entity will abide by. The policies are specified in writing and the agreements are legally binding contracts. Each new organization that joins the grid must agree to and abide by these security policies.

In a peer-to-peer system where each peer could potentially be in its own administrative domain, such an approach to acquiring trust is a high hurdle to pass. The number of organizations could well be in the thousands or tens of thousands. Furthermore, detecting breaches of the contract (such as malicious users) and enforcing the terms of the contract is unrealistic.

In addition to community-based trust (discussed above), the system should be able to assign different levels of trust to different resources based on the characteristics of the user and resource. Using a distributed computing example, the system may have a high degree of trust in professionally maintained server-class machines running in the same organization as the user; a medium level of trust in machines running at a different organization; and very-little trust in machines running on Internet-based grids. In each case, the system may provide different services: extensive use of virtual machine-based technology, encryption and redundancy for the Internet-based machines; data encryption and validation for machines at different organizations; and no special considerations for server machines running in the same organization.

2.d	Trust based on resource characteristics
2.d.1	Ability to run application in protected virtual environment (maintain resource integrity)
2.d.2	Ability to provide data integrity and confidentiality (encryption and data validation)
2.d.3	Ability to provide code integrity and confidentiality (using encryption, virtual environments and redundancy)

6.3 Trust in Data

An issue that is particularly important for peer-to-peer file systems is the issue of copyright identification: the data that is shared between the peers is in some cases copyrighted information and must be detected and copyright policies enforced. Digital rights management (DRM) is one mechanism to control the rights to data, but in some cases, active monitoring and identification of copyrighted material is needed. For example, corporations may have a policy to scan and limit the documents attached to email sent outside of the corporation or internet-based peer-to-peer systems may decide not to allow sharing of commercial copyrighted music or videos.

2.e	Support for DRM systems to protect data

2.f	Support for active monitoring and identification of copyrighted material

7. Failure

[Administrative Note: will be updated during the two weeks, April 14-April 27. Please add content as you see fit, but keep change tracking on and post your changes to the mailing list].

8. Location Awareness

9. Group Support

10. Security Considerations

This is a REQUIRED section.

Author Information

Contact information for authors.

Glossary

Recommended by not required.

Intellectual Property Statement

Full Copyright Notice

developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

References

1. Bhagwan, R., S. Savage, and G. Voelker. *Understanding Availability*. in *2nd International Workshop on Peer-to-Peer Systems*. 2003. Berkeley, CA.