Peer-to-Peer (P2P) Architecture:
Definition, Taxonomies, Examples, and Applicability

Abstract

   In this document, we provide a survey of P2P (Peer-to-Peer) systems.
   The survey includes a definition and several taxonomies of P2P
   systems. This survey also includes a description of which types of
   applications can be built with P2P technologies and examples of P2P
   applications that are currently in use on the Internet. Finally, we
   discuss architectural trade-offs and provide guidelines for deciding
   whether or not a P2P architecture would be suitable to meet the
   requirements of a given application.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   P2P (Peer-to-peer) systems have received a great deal of attention in
   the last few years. A large number of scientific publications
   investigate different aspects of P2P systems, several scientific
   conferences explicitly focus on P2P networking, and there is an
   Internet Research Task Force (IRTF) Research Group (RG) on P2P
   systems (the Peer-to-Peer RG). There are also several commercial and
   non-commercial applications that use P2P principles running on the
   Internet. Some of these P2P applications are among the most widely
   used applications on the Internet at present.

   However, despite all the above, engineers designing systems or
   developing protocol specifications do not have a common understanding
   of P2P systems. More alarming is the fact that many people in the
   telecom and datacom industries believe that P2P is synonymous with
   illegal activity, such as the illegal exchange of content over the
   Internet or P2P botnets.

   The goal of this document is to discuss the trade-offs involved in
   deciding whether a particular application can be best designed and
   implemented using a P2P paradigm or a different model (e.g., a
   client-server paradigm). The document also aims to provide
   architectural guidelines to assist in making such decisions. This
   document provides engineers with a high-level understanding of what
   defines a P2P system, what types of P2P systems exist, the
   characteristics that can be expected from such systems, and what
   types of applications can be implemented using P2P technologies.
   Such understanding is essential in order to appreciate the trade-offs
   referred to above. In addition, we stress the importance of the fact
   that P2P systems can be used to implement perfectly legitimate
   applications and business models by providing several examples
   throughout the document.

2. Definition of a P2P System

   In order to discuss P2P systems, we first need a working definition
   of a P2P system. In this section, we provide such a definition. All
   discussions in this document apply to systems that comply with that
   definition. In addition to providing examples of P2P systems, we
   provide a few examples of systems that comply only partially with the
   definition and, thus, cannot be strictly considered P2P systems.
   Since these systems are not fully P2P compliant, some of the
   discussions in this document may apply to them while others may not.
   We have chosen to include those examples anyway to stress the fact
   that P2P and centralized architectures are not completely disjoint

alternatives. There are many examples of systems that fall, for
instance, somewhere in between a pure P2P system and a centralized
one.

P2P is a term used in many contexts, sometimes with slightly
different meanings. It is possible to find several alternative
definitions, which are not all fully equivalent, in the existing
scientific literature. If we include other material (e.g., marketing
material) in our search for a definition on P2P, the diversity of
definitions is even higher.

The issue is that there is no clear border between a P2P paradigm and
other supposedly opposite paradigms such as client-server
[Milojicic2002]. In the extremes, some architectures are clearly P2P
while others are clearly client-server. However, there are
architectures that can be considered to be either or both, depending
on the definition for P2P being considered. Consequently, it is
important to understand what is common to all definitions of P2P and
what are the non-common traits some authors include in their own
definitions.

We consider a system to be P2P if the elements that form the system
share their resources in order to provide the service the system has
been designed to provide. The elements in the system both provide
services to other elements and request services from other elements.

In principle, all the elements in the system should meet the previous
criteria for the system to be considered P2P. However, in practice,
a system can have a few exceptions (i.e., a few nodes that do not
meet the criteria) and still be considered P2P. For example, a P2P
system can still be considered P2P even if it has a centralized
enrollment server. On the other hand, some systems divide endpoints
between peers and clients. Peers both request and provide services
while clients generally only request services. A system where most
endpoints behaved as clients could not strictly be considered P2P.

Although most definitions do not state it explicitly, many implicitly
assume that for a system to be P2P, its nodes need to be involved in
transactions that are related to services that do not directly
benefit the nodes.

Some authors add that the elements that form the P2P system, which
unsurprisingly are called peers, should be able to communicate
directly between themselves without passing intermediaries
[Schollmeier2001]. Other authors add that the system should be self
organizing and have decentralized control [Roussopoulus2004].

Note that the previous definitions are given within the context of a
single individual service. A complex service can be made up of
several individual services. Some of these individual services can
consist of P2P services and some of them can consist of client-server
services. For example, a file sharing client may include a P2P
client to perform the actual file sharing and a web browser to access
additional information on a centralized web server. Additionally,
there are architectures where a client-server system can serve as a
fallback for a service normally provided by a P2P system, or vice
versa.

Providing a service typically involves processing or storing data.
According to our definition, in a P2P system, peers share their
processing and storage capacity (i.e., their hardware and software
resources) so that the system can provide a service. For example, if
the service to be provided is a file distribution service, different
peers within the system will store different files. When a given
peer wants to get a particular file, the peer will first discover
which peer or peers have that file and then obtain the file from
those peers.

The definition for P2P provides us with a criterion to decide whether
or not a system is P2P. As examples, in the following sections we
apply the definition to the DNS, SIP, P2PSIP, and BitTorrent and
discuss which of these systems are P2P.

2.1. Applying the P2P Definition to the DNS

The DNS is a hierarchical distributed system that has sometimes been
classified as a hierarchical client-server system and sometimes as a
P2P system [Milojicic2002]. According to our definition, the DNS is
not a P2P system because DNS resolvers are service requesters but not
service providers. The elements in a system need to be both service
requesters and service providers for the system to be considered P2P.

2.2. Applying the P2P Definition to SIP

SIP [RFC3261] is a rendezvous protocol that allows a user to locate a
remote user and establish a communication session with that remote
user. Once the remote user is located, sessions are established in a
similar way in all SIP systems: directly between the nodes involved
in the session. However, the rendezvous function can be implemented
in different ways: the traditional SIP way and the P2P way. This
section discusses the former. Section 2.3 discusses the latter.

In traditional SIP, a central server is typically responsible for a
DNS domain. User agents in the domain register with the server.
This way, when a user agent wants to communicate with a remote user

agent in the same domain, the user agent consults the server, which
returns the contact information of the remote user agent. Session
establishment occurs directly between the user agents, without the
involvement of the server.

Inter-domain communications in SIP are implemented using server
federations. The servers responsible for each domain form a
federation in which they can communicate with each other. This way,
when a user agent wants to communicate with a remote user agent in a
different domain, the user agent consults its local server, which in
turn consults the server responsible for the remote user agent's
domain.

SIP user agents act as both clients and servers. A given user agent
can act as a client in a particular transaction and as a server in a
subsequent transaction. However, traditional SIP cannot be
considered a P2P system because user agents only share their
resources for their own benefit. That is, a given user agent is only
involved in transactions related to a service that benefits (somehow)
the user agent itself. For example, any given user agent is only
involved in SIP INVITE transactions intended to establish sessions
that involve the user agent. For a system to be P2P, its nodes need
to be involved in transactions that benefit others, that is,
transactions that are related to services that do not benefit the
nodes directly.

2.3. Applying the P2P Definition to P2PSIP

   In addition to the traditional way of using SIP, SIP can also be used
   in a way that is generally referred to as P2PSIP (P2PSIP is the name
   of the IETF working group developing the technology). In P2PSIP,
   user agents do not register their contact information with a central
   server. Instead, they register it with an overlay formed by the user
   agents in the system. This way, when a user agent wants to
   communicate with a remote user agent, the user agent consults the
   overlay, which returns the contact information of the remote user
   agent. Session establishment occurs, as usual, directly between the
   user agents. P2PSIP is a P2P system because nodes share their
   resources by storing data that is not related to them (i.e., contact
   information of different user agents) and are involved in
   transactions that are related to services that do not revert directly
   to the nodes themselves (e.g., the rendezvous of two remote user
   agents).

2.4. Applying the P2P Definition to BitTorrent

   BitTorrent [BitTorrent] is a protocol used to distribute files. The
   group of endpoints involved in the distribution of a particular file
   is called a swarm. The file is divided into several pieces. An
   endpoint interested in the file needs to download all the pieces of
   the file from other endpoints in the swarm. Endpoints downloading
   pieces of the file also upload pieces they already have to other
   endpoints in the swarm. An endpoint that both downloads (because it
   does not have the complete file yet) and uploads pieces is called a
   leecher (note that this definition is counterintuitive because, in
   other contexts, a leecher normally means someone that takes but does
   not give). When an endpoint has the whole file (i.e., it has all the
   pieces of the file), it does not need to download any pieces any
   longer. Therefore, it only uploads pieces to other endpoints. Such
   an endpoint is called a seeder.

   BitTorrent systems are P2P systems because endpoints request services
   from other endpoints (i.e., download pieces from other endpoints) and
   provide services to other endpoints (i.e., upload pieces to other
   endpoints). Note, however, that a particular swarm where most
   endpoints were infrastructure nodes that had the complete file from
   the beginning and, thus, acted all the time as seeders could not be
   strictly considered a P2P system because most endpoints would only be
   providing services, not requesting them.

3. Functions in a P2P System

   P2P systems include several functions. The following functions are
   independent of the service provided by the P2P system. They handle
   how peers connect to the system.

   o Enrollment function: nodes joining a P2P system need to obtain
      valid credentials to join the system. The enrollment function
      handles node authentication and authorization.

   o Peer discovery function: in order to join a P2P system (i.e., to
      become a peer), a node needs to establish a connection with one or
      more peers that are already part of the system. The peer
      discovery function allows nodes to discover peers in the system in
      order to connect to them.

   The functions above are provided in a centralized way in some P2P
   systems (e.g., through a central enrollment server and a central peer
   discovery server, which is sometimes called a bootstrap server).
   Taxonomies for P2P systems, which will be discussed in Section 4, do

not consider these functions when classifying P2P systems. Instead,
they classify P2P systems based on how the following set of functions
are implemented.

The following functions depend on the service provided by the P2P
system. That is, not all P2P systems implement all functions. For
example, a P2P system used only for storing data may not implement
the computing function. In another example, a P2P system used only
for computing may not implement the data storage function. Also,
some of these functions are implemented in a centralized way in some
P2P systems.

o Data indexing function: it deals with indexing the data stored in
   the system.

o Data storage function: it deals with storing and retrieving data
   from the system.

o Computation function: it deals with the computing performed by the
   system. Such computing can be related to, among other things,
   data processing or real-time media processing.

o Message transport function: it deals with message exchanges
   between peers. Depending on how this function is implemented,
   peers can exchange protocol messages through a central server,
   directly between themselves, or through peers that provide overlay
   routing.

Depending on the service being provided, some of the functions above
may not be needed. Section 5 discusses different types of P2P
applications, which implement different services.

4. Taxonomies for P2P Systems

   Taxonomies classify elements into groups so that they can be studied
   more easily. People studying similar elements can focus on common
   problem sets. Taxonomies also provide common terminology that is
   useful when discussing issues related to individual elements and
   groups of elements within a given taxonomy. In this section, we
   provide a few taxonomies for P2P systems in order to facilitate their
   study and to present such a common terminology.

   Given that different authors cannot seem to agree on a single common
   definition for P2P, the fact that there are also many different
   taxonomies of P2P systems should not come as a surprise. While
   classifying P2P systems according to different traits is something

normal, the fact that different authors use the same term to indicate
different things (e.g., first and second generation P2P systems mean
different things for different authors) sometimes confuses readers.

Arguably, the most useful classification of P2P systems has to do
with the way data is indexed. That is, how the data indexing
function is implemented. A P2P index can be centralized, local, or
distributed [RFC4981]. With a centralized index, a central server
keeps references to the data in all peers. With a local index, each
peer only keeps references to its own data. With a distributed
index, references to data reside at several nodes. Napster, early
versions of Gnutella (up to version 0.4), and Distributed Hash Table
(DHT)-based systems are examples of centralized, local, and
distributed indexes, respectively.

Indexes can also be classified into semantic and semantic-free. A
semantic index can capture relationships between documents and their
metadata whereas a semantic-free index cannot [RFC4981]. While
semantic indexes allow for richer searches, they sometimes (depending
on their implementation) fail to find the data even if it is actually
in the system.

Some authors classify P2P systems by their level of decentralization.
Hybrid P2P systems need a central entity to provide their services
while pure P2P systems can continue to provide their services even if
any single peer is removed from the system [Schollmeier2001].
According to this definition, P2P systems with a centralized index
are hybrid P2P systems while systems with local and distributed
indexes are pure P2P systems.

Still, some authors classify pure P2P systems by the level of
structure they show [Alima2005]. In unstructured systems, peers join
the system by connecting themselves to any other existing peers. In
structured systems, peers join the system by connecting themselves to
well-defined peers based on their logical identifiers. The
distinction between early unstructured systems (e.g., early versions
of Gnutella), which used local indexes and had no structure at all,
and structured systems (e.g., the DHT-based systems), which used
distributed indexes and had a well-defined structure, was fairly
clear. However, unstructured systems have evolved and now show a
certain level of structure (e.g., some systems have special nodes
with more functionality) and use distributed indexes. Therefore, the
border between unstructured and structured is somewhat blurry.

Some authors refer to different generations of P2P systems. For
some, the first, second, and third generations consist of P2P systems
using centralized indexes, flooding-based searches (i.e., using local
indexes), and DHTs (i.e., DHT-based distributed indexes),

respectively [Foster2003]. Other authors consider that second
generation systems can also have non-DHT-based distributed indexes
[Zhang2006]. Yet for other authors, the first and second generations
consist of P2P systems using unstructured (typically using flooding-
based searched) and structured (e.g., DHT-based) routing,
respectively [RFC4981]. Talking about generations of P2P systems in
a technical context is not useful (as stated previously, it is more
useful to classify systems based on how they index data) because
different generations are defined in different ways depending on the
author and because talking about generations gives the impression
that later generations are better than earlier ones. Depending on
the application to be implemented, a P2P system of an earlier
generation may meet the application's requirements in a better way
than a system of a later generation.

As discussed in Section 3, the previous taxonomies do not consider
the enrollment and the peer discovery functions. For example, a pure
P2P system would still be considered pure even if it had centralized
enrollment and peer discovery servers.

5. P2P Applications

   P2P applications developed so far can be classified into the
   following domains [Pourebrahimi2005] [Milojicic2002]: content
   distribution, distributed computing, collaboration, and platforms.

5.1. Content Distribution

   When most people think of P2P, they think of file sharing. Moreover,
   they think of illegal file sharing where users exchange material
   (e.g., songs, movies, and software in digital format) they are not
   legally authorized to distribute. However, despite people's
   perception, P2P file sharing systems are not intrinsically illegal.

   P2P file sharing applications provide one out of many means to store
   and distribute content on the Internet. HTTP [RFC2616] and FTP
   [RFC0959] servers are examples of other content distribution
   mechanisms. People would not claim that HTTP is an illegal mechanism
   just because a number of users upload material that cannot be legally
   distributed to an HTTP server where other users can download it. The
   same way, it is misleading to claim that P2P is illegal just because
   some users use it for illegal purposes.

   P2P content distribution systems are used to implement legitimate
   applications and business models that take advantage of the
   characteristics of these P2P systems. Examples of legitimate uses of
   these systems include the distribution of pre-recorded TV programs

[Rodriguez2005], Linux distributions [Rodriguez2005], game updates
[WoW], and live TV [Peltotalo2008] [Octoshape] by parties legally
authorized to distribute that content (e.g., the content owner).

The main advantage of P2P content distribution systems is their
scalability. In general, the more popular the content handled, the
more scalable the P2P system is. The peer that has the original
content (i.e., the owner of a file or the source of an audio or video
stream) distributes it to a fraction of the peers interested in the
content, and these peers in turn distribute it to other peers also
interested in the content. Note that, in general, there is no
requirement for peers distributing content to be able to access it
(e.g., the content may be encrypted so that peers without the
decryption key are content distributors but not content consumers).
Peers can distribute content to other peers in different ways. For
example, they can distribute the whole content, pieces of the content
(i.e., swarming), or linear combinations of pieces of content
[Gkantsidis2005]. In any case, the end result is that the peer with
the original content does not need to distribute the whole content to
all the peers interested in it, as it would be the case when using a
centralized server. Therefore, the capacity of the system is not
limited by the processing capacity and the bandwidth of the peer with
the original content and, thus, the quality of the whole service
increases.

An important area that determines the characteristics of a P2P
distribution system is its peer selection process. Interestingly,
the different parties involved in the distribution have different
views on how peers should be selected. Users are interested in
connecting to peers that have the content they want and also have
high bandwidth and processing capacity, and low latency so that
transfers are faster. The Content Delivery Network (CDN) operator
wants peers to connect first to the peers who have the rarest pieces
of the content being distributed in order to improve the reliability
of the system (in case those peers with the rare pieces of content
leave the system). Network operators prefer peers to perform local
transfers within their network so that their peering and transit
agreements are not negatively affected (i.e., by downloading content
from a remote network despite of the content being available
locally). Sometimes, all these requirements can be met at the same
time (e.g., a peer with a rare piece of content has high bandwidth
and processing capacity and is in the local network). However, other
times the system can just try and reach acceptable trade-offs when
selecting peers. These issues were the subject of the IETF P2P
Infrastructure (P2PI) workshop held in 2008.

Network operators also find that, depending on the dimensioning of
their networks (e.g., where the bottlenecks are), the different
traffic patterns generated by P2P or centralized CDNs can be more or
less easily accommodated by the network [Huang2007].

An example of a sensor network based on P2P content distribution and
Delay-tolerant Networking (DTL) is ZebraNet [Juang2002]. ZebraNet is
a network used to track zebras in the wild. Each zebra carries a
tracking collar that gathers data about the zebra (e.g., its
position) at different times. Mobile stations communicate wirelessly
with the collars in order to gather and consolidate data from
different zebras. Since not all the zebras get close enough to a
mobile station for their collars to be able to communicate with the
station, the collars communicate among them exchanging the data they
have gathered. In this way, a given collar provides the mobile
station with data from different zebras, some of which may never get
close enough to the mobile station. P2P networks are especially
useful in situations where it is impossible to deploy a communication
infrastructure (e.g., due to national park regulations or potential
vandalism) such as in the previous example or when tracking reindeers
in Lapland [SNC] (this project has focused on DTNs more than on P2P
so far, but some of its main constraints are similar to the ones in
ZebraNet). Note however that sensor networks such as ZebraNet cannot
be strictly considered P2P because the only node issuing service
requests (i.e., the only node interested in receiving data) is a
central node (i.e., the mobile station).

5.2. Distributed Computing

In P2P distributed computing, each task is divided into independent
subtasks that can be completed in parallel (i.e., no inter-task
communication) and delivered to a peer. The peer completes the
subtask using its resources and returns the result. When all the
subtasks are completed, their results are combined to obtain the
result of the original task.

Peers in P2P distributed computing systems are typically distributed
geographically and are connected among them through wide-area
networks. Conversely, in cluster computing, nodes in a cluster are
typically physically close to each other (often in the same room) and
have excellent communication capabilities among themselves.
Consequently, computer clusters can divide tasks into subtasks that
are not completely independent from one another and that cannot be
completed in parallel. The excellent communication capabilities
among the nodes in the cluster make it possible to synchronize the
completion of such tasks. Since computers in a cluster are so
tightly integrated, cluster computing techniques are not typically
considered P2P networking.

The main advantage of P2P distributed computing systems is that a
number of regular computers can deliver the performance of a much
more powerful (and typically expensive) computer. Nevertheless, at
present, P2P distributed computing can only be applied to tasks that
can be divided into independent subtasks that can be completed in
parallel. Tasks that do not show this characteristic are better
performed by a single powerful computer.

Note that even though distributed computing, in general, can be
considered P2P (which is why we have included it in this section as
an example of a P2P application), most current systems whose main
focus is distributed computing do not fully comply with the
definition for P2P provided in Section 2. The reason is that, in
those systems, service requests are typically generated only by a
central node. That is, most nodes do not generate service requests
(i.e., create tasks). This is why Grid computing [Foster1999] cannot
be strictly considered P2P [Lua2005]. Another well-known example
that cannot strictly be considered P2P either is SETI@home (Search
for Extra-Terrestrial Intelligence) [Seti], where the resources of
many computers are used to analyze radio telescope data. MapReduce
[Dean2004], a programming model for processing large data sets,
cannot strictly be considered P2P either, for the same reason. On
the other hand, a number of collaboration applications implement
distributed computing functions in a P2P way (see Section 5.3).

Another form of distributed computing that cannot be strictly
considered P2P (despite its name) are P2P botnets [Grizzard2007]. In
P2P botnets, service requests, which usually consist of generating
spam or launching Distributed Denial-of-Service (DDoS) attacks, are
typically generated by a central node (or a few central nodes); that
is why they cannot be strictly considered P2P. An example of this
type of P2P botnet that propagates using a DHT-based overlay is the
Storm botnet [Kanich2008]. In addition to their distributed
propagation techniques, some P2P botnets also use a distributed
command and control channel, which makes it more difficult to combat
them than traditional botnets using centralized channels [Cooke2005].
DHT-based overlays can also be used to support the configuration of
different types of radio access networks [Oechsner2006].

5.3. Collaboration

   P2P collaboration applications include communication applications
   such as Voice over IP (VoIP) and Instant Messaging (IM) applications.
   Section 2.3 included discussions on P2PSIP systems, which are an
   example of a standard-based P2P collaboration application. There are
   also proprietary P2P collaboration applications on the Internet
   [Skype]. Collaboration applications typically provide rendezvous,
   Network Address Translators (NAT) traversal, and a set of media-

related functions (e.g., media mixing or media transcoding). Note
that some of these functions (e.g., media transcoding) are,
effectively, a form of distributed computing.

P2P rendezvous systems are especially useful in situations where
there is no infrastructure. A few people with no Internet
connectivity setting up an ad hoc system to exchange documents or the
members of a recovery team communicating among themselves in a
disaster area are examples of such situations. P2PSIP is sometimes
referred to as infrastructureless SIP to distinguish it from
traditional SIP, which relies on a rendezvous server infrastructure.

5.4. Platforms

P2P platforms can be used to build applications on top of them. They
provide functionality the applications on top of them can use. An
example of such a platform is JXTA [Gong2001]. JXTA provides peer
discovery, grouping of peers, and communication between peers. The
goal with these types of P2P platforms is that they become the
preferred environment for application developers. They take
advantage of the good scalability properties of P2P systems.

6. Architectural Trade-Offs and Guidance

In this document, we have provided a brief overview of P2P
technologies. In order to dispel the notion that P2P technologies
can only be used for illegal purposes, we have discussed a number of
perfectly legitimate applications that have been implemented using
P2P. Examples of these applications include video conferencing
applications [Skype], the distribution of pre-recorded TV programs
[Rodriguez2005], Linux distributions [Rodriguez2005], game updates
[WoW], and live TV [Peltotalo2008] [Octoshape] by parties legally
authorized to distribute that content.

When deciding whether or not to use a P2P architecture to implement a
given application, it is important to consider the general
characteristics of P2P systems and evaluate them against the
application's requirements. It is not possible to provide any
definitive rule to decide whether or not a particular application
would be implemented best using P2P. Instead, we discuss a set of
trade-offs to be considered when making architectural decisions and
provide guidance on which types of requirements are better met by a
P2P architecture (security-related aspects are discussed in
Section 7). Ultimately, applications' operational requirements need
to be analyzed on a case-by-case basis in order to decide the most
suitable architecture.

   P2P systems are a good option when there is no existing
   infrastructure and deploying it is difficult for some reason. Ad hoc
   systems are usually good candidates to use P2P architectures.
   Disaster areas where existing infrastructures have been destroyed or
   rendered unusable can also benefit from P2P systems.

   One of the main features of P2P systems is their scalability. Since
   the system can leverage the processing and storage capacity of all
   the peers in the system, increases in the system's load are tackled
   by having the peers use more of their processing or storage capacity.
   Adding new peers generally increases the system's load but also
   increases the system's processing and storage capacity. That is,
   there is no typical need to update any central servers to be able to
   deal with more users or more load [Leibniz2007]. Adaptive P2P
   systems tune themselves in order to operate in the best possible mode
   when conditions such as number of peers or churn rate change
   [Mahajan2003]. In any case, at present, maintaining a running DHT
   requires nontrivial operational efforts [Rhea2005].

   Robustness and reliability are important features in many systems.
   For many applications to be useful, it is essential that they are
   dependable [RFC4981]. While there are many techniques to make
   centralized servers highly available, peers in a P2P system are not
   generally expected to be highly available (of course, it is also
   possible to build a more expensive P2P system with only highly
   available peers). P2P systems are designed to cope with peers
   leaving the system ungracefully (e.g., by crashing). P2P systems use
   techniques such as data replication and redundant routing table
   entries to improve the system's reliability. This way, if a peer
   crashes, the data it stored is not lost and can still be found in the
   system.

   The performance of a P2P system when compared to a server-based
   system depends on many factors (e.g., the dimensioning of the server-
   based system). One of the most important factors is the type of task
   to be performed. As we discussed in Section 5.2, if the task that
   needs to be computed can be divided into independent subtasks that
   can be completed in parallel, a P2P distributed computing system made
   up of regular computers may be able to perform better than even a
   super computer. If the task at hand consists of completing database
   queries, a well-dimensioned centralized database may be faster than a
   DHT.

   The performance of a P2P system can be negatively affected by a lack
   of cooperation between the peers in the system. It is important to
   have incentives in place in order to minimize the number of free
   riders in the system. Incentive systems generally aim to take the
   P2P system to optimal levels of cooperation [Feldman2004].

There are trade-offs between the scalability, robustness, and
performance of a particular P2P system that can be influenced through
the configuration of the system. For example, a P2P database system
where each peer stored all the information in the system would be
robust and have a high performance (i.e., queries would be completed
quickly) but would not be efficient or scalable. If the system
needed to grow, it could be configured so that each node stored only
a part of the information of the whole system in order to increase
its efficiency and scalability at the expense of its robustness and
performance.

Energy consumption is another important property of a system. Even
though the overall consumption of a client-server system is generally
lower than that of a P2P system providing the same service, P2P
systems avoid central servers (e.g., server farms) that can
potentially concentrate the consumption of high amounts of energy in
a single geographical location. When the nodes in a system need to
be up and running all the time anyway, it is possible to use those
nodes to perform tasks in a P2P way. However, using battery-powered
devices as peers in a P2P system presents some challenges because a
peer typically consumes more energy than a client in a client-server
architecture where they can go into sleep mode more often
[Kelenyi2008]. Energy-aware P2P protocols may be the solution to
these challenges [Gurun2006].

This section has discussed a set of important system properties and
compared P2P and centralized systems with respect to those
properties. However, the most important factor to take into
consideration is often cost. Both capital and operating costs need
to be taken into account when evaluating the scalability,
reliability, and performance of a system. If updating a server so
that it can tackle more load is inexpensive, a server-based
architecture may be the best option. If a highly available server is
expensive, a P2P system may be the best choice. With respect to
operating costs, as previously stated, at present, maintaining a
running DHT requires nontrivial operational efforts [Rhea2005].

In short, even though understanding the general properties of P2P and
server-based systems is important, deciding which architecture best
fits a particular application involves obtaining detailed information
about the application and its context. In most scenarios, there are
no easy rules that tell us when to use which architecture.

7. Security Considerations

Security is an important issue that needs to be considered when
choosing an architecture to design a system. The first issue that
needs to be considered is to which extent the nodes in the system can

be trusted. If all the nodes in the system are fully trusted (e.g.,
all the nodes are under the full control of the operator of the
system and will never act in a malicious or otherwise incorrect way),
a P2P architecture can achieve a high level of security. However, if
nodes are not fully trusted and can be expected to behave in
malicious ways (e.g., launching active attacks), providing an
acceptable level of security in a P2P environment becomes
significantly more challenging than in a non-P2P environment because
of its distributed ownership and lack of centralized control and
global knowledge [Mondal2006]. Ultimately, the level of security
provided by a P2P system largely depends on the proportion of its
nodes that behave maliciously. Providing an acceptable level of
security in a P2P system with a large number of malicious nodes can
easily become impossible.

P2P systems can be used by attackers to harvest IP addresses in use.
Attackers can passively obtain valid IP addresses of potential
victims without performing active scans because a given peer is
typically connected to multiple peers. In addition to being passive,
this attack is much more efficient than performing scans when the
address space to be scanned is large and sparsely populated (e.g.,
the current IPv6 address space). Additionally, in many cases there
is a high correlation between a particular application and a
particular operating system. In this way, an attacker can harvest IP
addresses suitable to launch attacks that exploit vulnerabilities
that are specific to a given operating system.

Central elements in centralized architectures become an obvious
target for attacks. P2P systems minimize the amount of central
elements and, thus, are more resilient against attacks targeted only
at a few elements.

When designing a P2P system, it is important to consider a number of
threats that are specific to P2P systems. Additionally, more general
threats that apply to other architectures as well are sometimes
bigger in a P2P environment. P2P-specific threats mainly focus on
the data storage functions and the routing of P2P systems.

In a P2P system, messages (e.g., service requests) between two given
peers generally traverse a set of intermediate peers that help route
messages between the two peers. Those intermediate peers can attempt
to launch on-path attacks they would not be able to launch if they
were not on the path between the two given peers. An attacker can
attempt to choose a logical location in the P2P overlay that allows
it to launch on-path attacks against a particular victim or a set of
victims. The Sybil [Douceur2002] attack is an example of such an
attack. The attacker chooses its overlay identifier so that it

allows the attacker to launch future attacks. This type of attack
can be mitigated by controlling how peers obtain their identifiers
(e.g., by having a central authority).

A trivial passive attack by peers routing messages consists of trying
to access the contents of those messages. Encrypting message parts
that are not required for routing is an obvious defense against this
type of attack.

An attacker can create a message and claim that it was actually
created by another peer. The attacker can even take a legitimate
message as a base and modify it to launch the attack. Peer and
message authentication techniques can be used to avoid this type of
attack.

Attackers can attempt to launch a set of attacks against the storage
function of the P2P system. The following are generic (i.e., non-
P2P-specific) attacks. Even if they are generic attacks, the way to
avoid or mitigate them in a P2P system can be more challenging than
in other architectures.

An attacker can attempt to store too much data in the system. A
quota system that can be enforced can be used to mitigate this
attack.

Unauthorized peers can attempt to perform operations on data objects.
Peer authorization in conjunction with peer authentication avoids
unauthorized operations.

A peer can return forged data objects claiming they are legitimate.
Data object authentication prevents this attack. However, a peer can
return a previous version of a data object and claim it is the
current version. The use of lifetimes can mitigate this type of
attack.

The following are P2P-specific attacks against the data storage
function of a P2P system. An attacker can refuse to store a
particular data object. An attacker can also claim a particular data
object does not exist even if another peer created it and stored it
on the attacker. These DoS (Denial-of-Service) attacks can be
mitigated by using data replication techniques and performing
multiple, typically parallel, searches.

Attackers can attempt to launch a set of attacks against the routing
of the P2P system. An attacker can attempt to modify the routing of
the system in order to be able to launch on-path attacks. Attackers
can use forged routing maintenance messages for this purpose. The
Eclipse attack [Singh2006] is an example of such an attack.

Enforcing structural constraints or enforcing node degree bounds can mitigate this type of attack.

It is possible to launch DoS attacks by modifying or dropping routing maintenance messages or by creating forged ones. Having nodes get routing tables from multiple peers can help mitigate this type of attack.

Attackers can launch a DoS attack by creating churn. By leaving and joining a P2P overlay rapidly many times, a set of attackers can create large amounts of maintenance traffic and make the routing structure of the overlay unstable. Limiting the amount of churn per node is a possible defense against this attack.

8. Acknowledgements

Jouni Maenpaa and Jani Hautakorpi helped with the literature review. Henning Schulzrinne provided useful ideas on how to define P2P systems. Bruce Lowekamp, Dan Wing, Dan York, Enrico Marocco, Cullen Jennings, and Frank Uwe Andersen provided useful comments on this document. Loa Andersson, Aaron Falk, Barry Leiba, Kurtis Lindqvist, Dow Street, and Lixia Zhang participated in the IAB discussions on this document.

9. IAB Members at the Time of This Writing

Marcelo Bagnulo
Gonzalo Camarillo
Stuart Cheshire
Vijay Gill
Russ Housley
John Klensin
Olaf Kolkman
Gregory Lebovitz
Andrew Malis
Danny McPherson
David Oran
Jon Peterson
Dave Thaler

10. Informative References

[Alima2005]   Alima, L., Ghodsi, A., and S. Haridi, "A
                     Framework for Structured Peer-to-peer Overlay
                     Networks", Global Computing, vol. 3267, Lecture
                     Notes in Computer Science: Springer Berlin /
                     Heidelberg, pp. 223-249, 2005.

   [BitTorrent]   Cohen, B., "The BitTorrent Protocol Specification
                       Version 11031", February 2008.

   [Cooke2005]   Cooke, E., Jahanian, F., and D. McPherson, "The
                       Zombie roundup: understanding, detecting, and
                       disrupting botnets", Proceedings of the Steps to
                       Reducing Unwanted Traffic on the Internet
                       Workshop, 2005.

   [Dean2004]   Dean, J. and S. Ghemawat, "MapReduce: Simplified
                       Data Processing on Large Clusters", Sixth
                       Symposium on Operating System Design and
                       Implementation (OSDI '04), December 2004.

   [Douceur2002]   Douceur, J., "The Sybil Attack", IPTPS 02,
                       March 2002.

   [Farber1972]   Farber, D. and K. Larson, "The Structure of a
                       Distributed Computer System - The Communications
                       System", Proceedings Symposium on Computer-
                       Communications Networks and Teletraffic,
                       Microwave Research Institute of Polytechnic
                       Institute of Brooklyn pp. 21-27, 1972.

   [Feldman2004]   Feldman, M., Lai, K., Stoica, I., and J. Chuang,
                       "Robust Incentive Techniques for Peer-to-peer
                       Networks", Proceedings of the 5th ACM Conference
                       on Electronic Commerce, 2004.

   [Foster1999]   Foster, I., "Computational Grids", Chapter 2 of
                       The Grid: Blueprint for a New
                       Computing Infrastructure, 1999.

   [Foster2003]   Foster, I. and A. Iamnitchi, "On Death, Taxes,
                       and the Convergence of Peer-to-Peer and Grid
                       Computing", 2nd International Workshop in Peer-
                       to-Peer Systems IPTPS '02, 2003.

   [Gkantsidis2005]   Gkantsidis, C. and P. Rodriguez, "Network Coding
                       for Large Scale Content Distribution", IEEE
                       INFOCOM 2005, vol. 4, March 2005.

   [Gong2001]   Gong, L., "JXTA: A Network Programming
                       Environment", IEEE Internet Computing, vol. 5,
                       no. 3, pp. 88-95, 2001.

   [Gray1983]   Gray, J. and S. Metz, "Solving the Problems of
                           Distributed Databases", Data Communications, pp.
                           183-192, 1983.

   [Gray1986A]   Gray, J., "An Approach to Decentralized Computer
                           Systems", IEEE Transactions on Software
                           Engineering, V 12.6, pp. 684-689, 1986.

   [Gray1986B]   Gray, J. and M. Anderton, "Distributed Systems:
                           Four Case Studies", IEEE Transactions on
                           Computers and Tandem Technical Report 85.5, 1986.

   [Grizzard2007]   Grizzard, J., Sharma, V., Nunnery, C., Kang, B.,
                           and D. Dragon, "Peer-to-peer botnets: overview
                           and case study", Proceedings of Hot Topics in
                           Understanding Botnets (HotBots '07), 2007.

   [Gurun2006]   Gurun, S., Nagpurkar, P., and B. Zhao, "Energy
                           Consumption and Conservation in Mobile Peer-to-
                           Peer Systems", First International Workshop on
                           Decentralized Resource Sharing in Mobile
                           Computing and Networking (MobiShare 2006), 2006.

   [Huang2007]   Huang, Y., Rabinovich, M., and Z. Xiao,
                           "Challenges of P2P Streaming Technologies for
                           IPTV Services", IPTC Workshop International World
                           Wide Web Conference, Edinburgh, Scotland, United
                           Kingdom, May 2006.

   [Juang2002]   Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh,
                           L., and D. Rubenstein, "Energy-efficient
                           computing for wildlife tracking: design tradeoffs
                           and early experiences with ZebraNet", Proceedings
                           of Conference on Computer and Communications
                           Security (CCS), ACM, 2002.

   [Kanich2008]   Kanich, C., Levchenko, K., Enright, B., Voelker,
                           G., Paxson, V., and S. Savage, "Spamalytics: An
                           Empirical Analysis of Spam Marketing Conversion",
                           Proceedings of Conference on Computer and
                           Communications Security (CCS) (ACM),
                           October 2008.

   [Kelenyi2008]   Kelenyi, I. and J. Nurminen, "Energy Aspects of
                       Peer Cooperation - Measurements with a Mobile DHT
                       System", in Proc. of Cognitive and Cooperative
                       Wireless Networks Workshop in the IEEE
                       International Conference on Communications 2008,
                       Beijing, China, pp. 164-168, 2008.

   [Leibniz2007]   Leibniz, K., Hobfeld, T., Wakamiya, N., and M.
                       Murata, "Peer-to-Peer vs. Client/Server:
                       Reliability and Efficiency of a Content
                       Distribution Service", Lecture Notes in Computer
                       Science, LNCS 4516, pp. 1161-1172, 2007.

   [Lua2005]   Keong Lua, E., Crowcroft, J., Pias, M., Sharma,
                       R., and S. Lim, "A Survey and Comparison of Peer-
                       to-peer Overlay Network Schemes", IEEE
                       Communications Surveys & Tutorials, vol. 7, no.
                       2, Second Quarter 2005, pp. 72-93, 2005.

   [MMUSIC-ICE]   Rosenberg, J., "Interactive Connectivity
                       Establishment (ICE): A Protocol for Network
                       Address Translator (NAT) Traversal for Offer/
                       Answer Protocols", Work in Progress,
                       October 2007.

   [Mahajan2003]   Mahajan, R., Castro, M., and A. Rowstron,
                       "Controlling the Cost of Reliability in Peer-to-
                       Peer Overlays", Proceedings of the 2nd
                       International Workshop on Peer-to-Peer
                       Systems (IPTPS '03), 2003.

   [Milojicic2002]   Milojicic, D., Kalogeraki, V., Lukose, R.,
                       Nagaraja, K., Pruyne, J., Richard, B., Rollins,
                       S., and Z. Xu, "Peer-to-Peer Computing",
                       Technical Report HP, March 2002.

   [Mondal2006]   Mondal, A. and M. Kitsuregawa, "Privacy, Security
                       and Trust in P2P environments: A Perspective",
                       17th International Conference on Database and
                       Expert Systems Applications 2006 (DEXA '06),
                       September 2006.

   [Octoshape]   "Octoshape - Large Scale Live Streaming
                       Solutions", <http://www.octoshape.com>.

   [Oechsner2006]   Oechsner, S., Hobfeld, T., Tutschku, K.,
                         Andersen, F., and L. Caviglione, "Using Kademlia
                         for the Configuration of B3G Radio Access Nodes",
                         Proceedings of the Fourth Annual IEEE
                         International Conference on Pervasive Computing
                         and Communications Workshops (PERCOMW '06), 2006.

   [Peltotalo2008]   Peltotalo, J., Harju, J., Jantunen, A., Saukko,
                         M., and L. Vaatamoinen, "Peer-to-Peer Streaming
                         Technology Survey", Seventh International
                         Conference on Networking, Cancun, Mexico, pp.
                         342-350, April 2008.

   [Pourebrahimi2005]   Pourebrahimi, B., Bertels, K., and S.
                         Vassiliadis, "A Survey of Peer-to-Peer Networks",
                         Proceedings of the 16th Annual Workshop on
                         Circuits, Systems, and Signal Processing, ProRisc
                         2005, November 2005.

   [RFC0959]   Postel, J. and J. Reynolds, "File Transfer
                         Protocol", STD 9, RFC 959, October 1985.

   [RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
                         Masinter, L., Leach, P., and T. Berners-Lee,
                         "Hypertext Transfer Protocol -- HTTP/1.1",
                         RFC 2616, June 1999.

   [RFC3261]   Rosenberg, J., Schulzrinne, H., Camarillo, G.,
                         Johnston, A., Peterson, J., Sparks, R., Handley,
                         M., and E. Schooler, "SIP: Session Initiation
                         Protocol", RFC 3261, June 2002.

   [RFC4981]   Risson, J. and T. Moors, "Survey of Research
                         towards Robust Peer-to-Peer Networks: Search
                         Methods", RFC 4981, September 2007.

   [RFC5128]   Srisuresh, P., Ford, B., and D. Kegel, "State of
                         Peer-to-Peer (P2P) Communication across Network
                         Address Translators (NATs)", RFC 5128,
                         March 2008.

   [Rhea2005]   Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J.,
                         Ratnasamy, S., Shenker, S., Stoica, I., and H.
                         Yu, "Open DHT: A Public DHT Service and Its
                         Uses", ACM/SIGCOMM CCR'05, vol. 35, Issue 4,
                         October 2005.

   [Rodriguez2005]   Rodriguez, P., Tan, S., and C. Gkantsidis, "On
                        the Feasibility of Commercial Legal P2P Content
                        Distribution", ACM/SIGCOMM CCR'06, January 2006.

   [Roussopoulus2004]   Roussopoulus, M., Baker, M., Rosenthal, D.,
                        Guili, T., Maniatis, P., and J. Mogul, "2 P2P or
                        Not 2 P2P", Workshop on Peer-to-Peer Systems,
                        February 2004.

   [SNC]   "http://www.snc.sapmi.net".

   [Schollmeier2001]   Schollmeier, R., "A Definition of Peer-to-Peer
                        Networking for the Classification of Peer-to-Peer
                        Architectures and Applications", In Proceedings
                        of the First International Conference on Peer-to-
                        Peer Computing P2P '01, 2001.

   [Seti]   "SETI@home", <http://setiathome.berkeley.edu>.

   [Singh2006]   Singh, A., Ngan, T., Druschel, T., and D.
                        Wallach, "Eclipse Attacks on Overlay Networks:
                        Threats and Defences", INFOCOM 2006, April 2006.

   [Skype]   "Skype", <http://www.skype.com>.

   [Tanenbaum1981]   Tanenbaum, A. and S. Mullender, "An Overview of
                        the Amoeba Distributed Operating System", ACM
                        SIGOPS Operating Systems Review, 1981.

   [WoW]   "World of Warcraft Community Site",
                        <http://www.worldofwarcraft.com>.

   [Zhang2006]   Zhang, Y., Chen, C., and X. Wang, "Recent
                        Advances in Research on P2P Networks", In
                        Proceedings of the Seventh International
                        Conference on Parallel and Distributed Computing,
                        Applications, and Technologies PDCAT '06, 2006.

Appendix A. Historical Background on Distributed Architectures

   In this appendix, we briefly provide historical background on
   distributed architectures. Distributed architectures are relevant to
   P2P because P2P architectures are a type of distributed architecture.
   That is, a distributed architecture is considered P2P if it meets a
   set of requirements, which are discussed in Section 2.

   In centralized architectures (e.g., client-server architectures), a
   central server (or very few central servers) undertakes most of the
   system's processing and storage. Conversely, decentralized
   architectures contain no (or very few) centralized elements.

   The increasing spread of packet-switched network technologies in the
   1970s made it possible to develop operational distributed computer
   systems [Farber1972]. Distributed computer systems received a lot of
   attention within the research community. Research focused on
   distributing the different parts of a computer system, such as its
   operating system [Tanenbaum1981] or its databases [Gray1983]. The
   idea was to hide from the user the fact that the system was
   distributed. That is, the user did not have to worry or even be
   aware of the fact that his or her files were stored in different
   computers or the fact that his or her tasks were processed also in a
   distributed way. Actions such as file transfers and task allocations
   were taken care of by the system in an automated fashion and were
   transparent to the user.

   In the middle of the 1980s, building distributed computer systems
   using general-purpose off-the-shelf hardware and software was
   believed to be not much harder than building large centralized
   applications [Gray1986A]. It was understood that distributed systems
   had both advantages and disadvantages when compared to centralized
   systems. Choosing which type of system to use for a particular
   application was a trade-off that depended on the characteristics and
   requirements of the application [Gray1986B].

   The client-server paradigm, where a client makes a request to a
   server that processes the request and returns the result to the
   client, was and is used by many Internet applications. In fact,
   client-server architectures were so ubiquitous on the Internet that,
   unfortunately, the Internet itself evolved as if the majority of the
   endpoints on the Internet were only interested in applications
   following the client-server model. With the appearance of Network
   Address Translators (NATs) and stateful firewalls, most Internet
   endpoints lost the ability to receive connections from remote
   endpoints unless they first initiated a connection towards those
   nodes. While NATs were designed not to disrupt client-server
   applications, distributed applications that relied on nodes receiving

connections were disrupted. In a network full of NATs, these types
of distributed applications could only be run among nodes with public
IP addresses. Of course, most users did not like applications that
only worked some of the time (i.e., when their endpoint happened to
have a public IP address). Therefore, the loss of global
connectivity caused by NATs was one of the reasons why applications
that did not follow the client-server paradigm (e.g., P2P
applications) took a relatively long time to be widely deployed on
the public Internet.

The design of NAT traversal mechanisms has made it possible to deploy
all types of distributed applications over a network without global
connectivity. While the first NAT traversal mechanisms used by P2P
applications were proprietary [RFC5128], nowadays there are standard
NAT traversal mechanisms such as Interactive Connectivity
Establishment (ICE) [MMUSIC-ICE]. ICE makes it possible for
endpoints to establish connections among themselves in the presence
of NATs. The recovery of global connectivity among Internet
endpoints has made it possible to deploy many P2P applications on the
public Internet (unfortunately, the fact that global connectivity is
not supported natively at the network layer makes it necessary for
applications to deal with NATs, which can result in highly complex
systems). Some of these P2P applications have been very successful
and are currently used by a large number of users.

Another factor that made it possible to deploy distributed
applications was the continuous significant advances in terms of
processing power and storage capacity of personal computers and
networked devices. Eventually, most endpoints on the Internet had
capabilities that previously were exclusively within the reach of
high-end servers. The natural next step was to design distributed
applications that took advantage of all that distributed available
capacity.

Authors' Addresses

    Gonzalo Camarillo (editor)
    Ericsson
    Hirsalantie 11
    Jorvas 02420
    Finland

    EMail: Gonzalo.Camarillo@ericsson.com

    Internet Architecture Board

    EMail: iab@iab.org