Implementation of Adaptive Control for P2P Overlays

Theofrastos Koulouris[1], Robert Henjes[2], Kurt Tutschku[2], and Hermann de Meer[1,3]

[1] Department of Electronic and Electrical Engineering, University College London, Torrington Place, London, WC1E 7JE, United Kingdom, t.koulouris@ee.ucl.ac.uk
[2] Institute of Computer Science, University of Würzburg, Am Hubland, 97074, Würzburg, Germany, {henjes|tutschku}@informatik.uni-wuerzburg.de
[3] Faculty of Mathematics and Informatics, University of Passau, Innstr. 33, 94032, Passau, Germany, demeer@fmi.uni-passau.de

Abstract. Peer-to-peer networking enjoys euphoric support and fierce resistance simultaneously, and for the same reasons. It presents a model where decentralization and lack of structure, hierarchy and control are promoted. Although significant research is carried out to tackle individual issues arising from that paradigm, there has been no obvious approach for evening out differences on a more general basis. In this paper we introduce a framework and provide implementation techniques for such an approach. The framework aims at integrating partial techniques that solve individual problems and has been designed for flexibility. The integrated approach we are proposing includes forming and maintaining of peer-to-peer overlays, controlling the underlying topology being formed, limiting the signaling traffic being generated and optimizing the payload traffic.

1 Introduction

Peer-to-peer (P2P) applications are close to become the dominating application of the Internet: P2P cooperation appears highly attractive to an increasing number of users due to appealing "free" resource sharing and easy use. P2P services can be loosely defined as being about *networked cooperation of equals*. Three main characteristics of P2P services can be emphasized: sharing of pooled and exchangeable resources, all nodes having similar roles, and all nodes being highly autonomous. That contrasts sharply with other distributed architectures such as Client/Server where asymmetric roles are typical. While loss of a primary component, i.e. the breakdown of a server, in an asymmetric architecture may result in a major disruption, any peer in a peer-to-peer architecture can be moved without resulting in a loss of service.

P2P systems are often referred to as being self-organizing where a coherent behavior emerges spontaneously without external coercion or control. Pure P2P architectures, such as early the Gnutella service, however turned out to be non-scalable. As a response, the need to introduce structure and limited control has been recognized, cf. [19]. In order to introduce heterogeneity into unstructured, pure P2P services, various mechanisms have been proposed. The suggestions range from "ultrapeers" and "superpeers", as in Gnutella [9] and Kazaa [18] respectively, to distributed mediation

N. Wakamiya et al. (Eds.): IWAN 2003, LNCS 2982, pp. 292–306, 2004.

servers and peer caches as in eDonkey2000 [20]. These approaches comprise only partial solutions to a more complex control problem. In particular, variability in service demand or load patterns can only be dealt with in a limited way. The demand for services may form hot spots which may shift within an overlay from one location to another one over time. P2P applications may therefore require a more flexible and dynamic method of control and management [3]. In particular, different control methods should be in place when and where needed, should be flexibly usable in combination with each other, and should be extensible in an evolutionary manner. More generally, it is our goal to introduce and to implement control and structure into peer-to-peer applications on demand.

In this paper we introduce a framework which facilitates a flexible and highly adaptive mode of P2P service management and P2P overlay management. The framework provides means for supporting self-organization, e.g. by mechanism to restructure an overlay topology. We present in detail the implementation of our scheme for an adaptive control of peer-to-peer overlays. The approach is based on the introduction of the concept of virtual nodes, called Active Virtual Peers (AVP). The proposed approach based on AVPs includes for example a *dynamic* forming and maintaining of peer-to-peer overlays or an adaptive routing of signaling and download traffic. The approach is also extensible to enforce security or ownership rights and to include mechanisms of charging. These additional features, however, are beyond the scope of this paper.

This paper is organized as follows: Section 2 discusses the objectives and requirements of control for P2P overlays. Section 3 presents the Active Virtual Peer concept. Next, in section 4, a prototype implementation of an AVP for optimizing Gnutella is described. Section 5 presents a discussion of the performance features of the AVP concept. A related work section is following in Section 6. Finally, Section 7 summarizes and concludes the paper.

2 Objectives and Requirements on Control for P2P Overlays

P2P services are effective in providing solutions in a large area of applications because of their distributed nature and focus they give to the resources found on the edges of the network. However, it has become evident over the past few years that some form of control is necessary to tackle issues such as the use of the service, the separation between the P2P overlay and the network layer, the short and unpredictable lifecycles of peer relations and the high signaling traffic generated, as we examine in [3]. We believe that there exist four areas where the enforcement of control will be beneficiary for such applications.

The first is *access control*. Participants of P2P overlays are typically granted access to all resources offered by the peers. These resources are valuable. Thus, the resource provider, either content provider or network provider, need to identify and regulate the admission to the overlay. In particular for P2P file sharing applications, access control should block off P2P applications or enable controlled content sharing.

The second area is *resource management*. The resources of individual peers have to be treated with care, e.g. low-bandwidth connected peers should not be overloaded with download requests and exploited equally. For P2P file sharing applications, for

example, content caching capabilities will improve the performance while reducing the stress imposed on the network.

A third area of interest is *overlay load control*. Overlay load control copes with traffic flows inside the overlay. Its goal is to balance the traffic and load in order to maintain sufficient throughput inside the overlay while also protecting other network services by mapping this load in an optimum way onto the underlying network infrastructure

Finally, the forth area of command is adaptive *topology control.* Overlay connections may be established or destroyed arbitrarily by the peers since they can join or leave the virtual network at any time. Topology control may enforce redundant connections, thus increasing the reliability of the service. In addition, topology control may force the structure of the virtual network to be more efficient and faster in locating resources when using broadcast protocols.

The last two areas support the aim of having adaptive and application-suited, management strategies for P2P services. The outlined control objectives might violate the populist concept of unlimited access to free resources in P2P services, but control mechanisms governed by these objectives do increase the stability of P2P services based on overlays. It is necessary to find the proper trade-off between regulation and autonomy in P2P overlays.

Having identified the objectives of control for a P2P overlay, it is important to examine how adaptive and un-supervised control mechanisms need to be implemented, without diminishing the virtues of the P2P model or introducing further complexity and overhead to the network. We believe that it is vital to preserve the autonomy of the peers inside a P2P network. Additional control loops, which adapt to the behavior of a P2P overlay, must not interfere with the autonomous nature of any P2P application. To achieve this goal, we suggest implementing control through an additional *support infrastructure*. This infrastructure will provide all the necessary tools and interfaces to implement the desired forms of control and at the same time protect a P2P application. In addition, it is able to combine different algorithms in order to obtain optimally suited control structures. Finally, the mechanisms in this infrastructure permit self-organization or constraint-based self-organization, with the ultimate aim of flexibility and adaptivity. The support infrastructure will be formed of self-organized, interworking modules that may resemble a P2P network on their own.

3 The Active Virtual Peer Concept

The main element of the support infrastructure suggested in this paper is the Active Virtual Peer (AVP). As its name implies, an AVP is a virtual entity which interacts with other peers inside a P2P network. An AVP is a representative of a community of peers. Its purpose is to enhance, control and make the P2P relation more efficient inside that community. AVPs enable flexibility and adaptivity by the use of self-organization. An AVP consists of various distributed and coordinated components that facilitate different forms of control. By combining these components based on network conditions or administrative policies, we can create AVPs of different functionality.

The AVP performs certain functions, not expected by an ordinary peer. These AVP functions are arranged in horizontal layers as well as in vertical planes, see Figure 1.

The horizontal layers correspond to the layers on which an AVP imposes control. The vertical separation describes the functional planes of AVPs. These architectural planes have been examined in detail in [3,4].

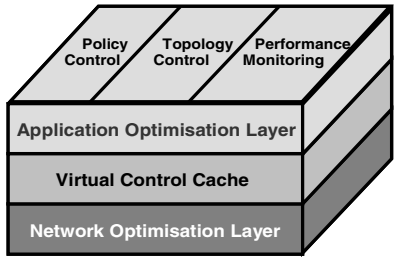

Fig. 1. The AVP architectural layers.

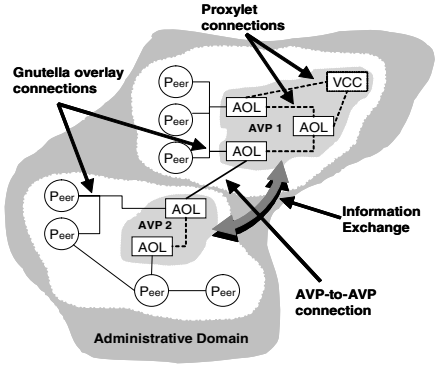

Fig. 2. The AVP realm.

The upper horizontal layer of an AVP is called the "Application Optimization Layer (AOL)". It controls and optimizes the peer-to-peer relation on the application level. The AOL may apply application-specific routing in conjunction with *access policies*. The routing performed by the AOL is based on metrics such as the state of the peers ("virtual peer state") or the state of the links between peers ("virtual overlay link state") thus changing the peer load and overlay link characteristics such as packet drop rate, throughput, or delay. In addition, the AOL allows for active overlay topology control, which is accomplished in two ways. The Active Virtual Peer may initiate, accept or terminate overlay connections based on access restriction or topology features. Topology characteristics such as the number of overlay connections or characteristic path length can be enforced or may govern the overlay structure. Furthermore, the AOL layer makes also use of the ALAN control mechanisms, examined below, for implementing its self-organization features. The AOL can instantiate modules implementing AOL functions whenever and wherever needed. These features enable the AOL to adapt the virtual overlay structure to varying demand, traffic patterns and connectivity requirements by launching new overlay connections and new virtual peers. These self-organization features make the AOL a very flexible architecture.

The middle layer of the AVP is denoted as the "Virtual Control Cache (VCC)". The VCC provides content caching on the application-level similar to conventional proxies. By maintaining often-requested content in close proximity, for instance inside an ISP's domain, large economies in resources and performance gains can be achieved. In addition, the VCC may offer control flow aggregation functions.

The lower layer of AVPs is denoted as the "Network Optimization Layer (NOL)". Its main task is the implementation of dynamic traffic engineering capabilities that map the P2P traffic onto the network layer in an optimized way. The mapping is performed with respect to the performance control capabilities of the applied transport technology. The AVP architecture may apply traffic engineering for standard IP routing protocols [7] as well as for explicit QoS enabled mechanisms like MPLS [8].

Figure 2, above, depicts a scenario where two AVPs, AVP 1 and AVP 2, are located within a single administrative domain. AVP 1 consists of three AOL modules and one VCC component, while AVP 2 comprises of two AOL modules. Multiple ordinary peers, denoted by "Peer", maintain connections to them. The two AVPs maintain overlay connections to each other. The AOL modules of the AVPs are in command of the overlay connections. This way, the AVPs can impose control on the overlay connection.

Implementation support. The current instance of the AVP technology is based on the *Application Level Active Networking* (ALAN) concept [1, 2]. The ALAN infrastructure allows a rapid deployment of network services and their on-demand provision to specified users or communities. ALAN is based on an overlay technique: Active nodes, which operate on the application level, are strategically placed within the network. These nodes, called *Execution Environments for Proxylets* (EEPs), enable the dynamic loading and execution of active code elements, denoted as *proxylets*, from designated servers. The resulting services may interfere with data transport and control. ALAN provides mechanisms for EEP discovery, application specific routing, and service creation by deploying a web of proxylets across the physical infrastructure. The Self Organizing Application-level Routing (SOAR) protocol [1], which is a key component of ALAN, enables clustering and grouping of proxylets. This way, ALAN facilitates the creation of an application-specific connectivity mesh and the dynamic forming of topology regions. Finally, ALAN provides the basic administrative mechanisms necessary for managing such an architecture.

The AVP layer modules are implemented by single or multiple interconnected proxylets. This allows the implementation of the layered AVP architecture in separate components. For instance, a proxylet may execute the AOL functions whereas an additional proxylet may materialize the Virtual Control Cache or the Network Optimization Layer. This approach facilitates better flexibility and efficiency in the constantly changing conditions of a Peer-to-Peer overlay. Different configurations of AVPs can be deployed in parts of the network that experience different characteristics, or even in the same network at different times of the day when conditions have changed. In addition, it is possible that different proxylets exist which implement the same layer functions differently. This gives further choice over the functionality of the AVP.

How an AVP imposes control. Having earlier identified the objectives for control of a P2P overlay, it is time to see how the AVP facilitates these control issues. Deployed AVPs create a realm wherein they constantly exchange information. Each AVP consists of multiple AOL and VCC proxylets which communicate and collaborate. The exchange of information allows for coordinated control of the overlay. A realm of AVPs is more suitable to evaluate the conditions inside a particular part of a P2P overlay than a single entity and this knowledge is distributed in order to achieve better results. Again, this capability promotes the flexibility and adaptivity of the AVP approach. Continuing, an AVP imposes control by providing effectors on connection level. The effectors comprise so far the *Router module* and the *Connection Manager module*. The Connection Manager enforces control by manipulating the connections peers maintain with each other. That is a significant difference compared to most P2P applications where the way peers connect to each other is random. By applying connection management, the AVP can enforce different control schemes.

The *Router module* governs the relaying of messages on application–level according to local or federated constraints, e.g. access restriction or virtual peer state information. The *Sensor module* provides state information for the distributed and collaborative control scheme. In the remainder of the paper, we discuss in detail how the suggested effectors are implemented.

4 Implementation of the AVP

The AVP concept is not based on any particular P2P application and does not require any specific P2P components in order to operate. Furthermore, the AVP does not address issues found only on P2P file-sharing applications but provides a generic performance management framework suitable for any type of P2P application that uses overlays. Nevertheless, for evaluating our prototype implementation, we use the Gnutella P2P file-sharing protocol as a vehicle and test environment. We chose Gnutella because it is a well-tested, open source, fully distributed P2P network with thousands of users; therefore ideal for realistic experiments. Furthermore, through Gnutella we are able to illustrate several realistic showcases where the AVP technology can provide solutions, many of them presented below. The showcases presented in the next section are all representative of experiments carried out at the University of Wuerzburg and University College London. We focus on the Gnutella protocol version 0.6 [5] to join the Gnutella network (GNet).

Access control. One of the core capabilities of the AVP is access control. The AOL component can create areas of control inside a P2P overlay, where all communications between the controlled domain and the global Gnutella network are managed by the AOL. Its goal is to control who can access the peers and their resources inside the domain. An AOL proxylet imposes access control by blocking and modifying Gnutella packets communicated between the controlled domain and the global Gnutella network. The result is that peers inside the controlled domain see only each other and become invisible to any peer outside that domain that is not granted access. At the same time, the AOL proxylet becomes the single point of contact between the controlled domain and the global network.

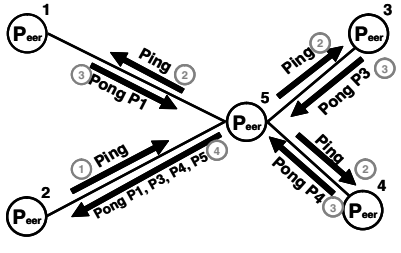

Fig. 3a. Gnutella conventional forwarding

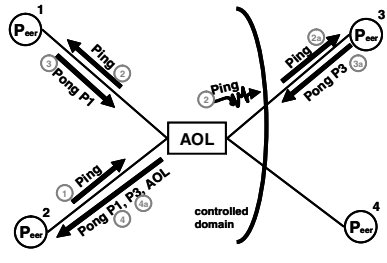

Fig. 3b. New routing by AOL proxylet

The access control as implemented by an AOL proxylet can be better illustrated by the following scenario, depicted in Figures 3a and 3b, and examined more extensively in [26]. In Figure 3a, Peers 1 to 5 reside inside the global Gnutella overlay. Peer 2 sends out a Gnutella "Ping" message in order to discover other peers. Under the Gnutella protocol, a Ping has to be forwarded by the receiving peer, i.e. Peer 5, to any peer in its vicinity, who in turn has to respond with a "Pong" message. Thus, Peer 2 receives "Pongs" from Peers 1, 3, 4, and 5.

In the access controlled scenario, see Figure 3b, an AVP forms a *controlled domain* (CD) for Peers 1 and 2. In order to facilitate the access control, the AOL proxylet establishes connections with all peers. When Peer 2 sends out a "Ping" to discover other peers, the AOL proxylet intercepts the "Ping" message and forwards it unmodified to Peer 1 which is part of the CD. In addition, it modifies that Ping so it seems like it was initiated by the proxylet and relays it to the outside world. Peer 2 receives "Pongs" by Peer 1 and the AOL proxylet and concludes that only these two peers comprise its neighborhood. The AOL proxylet captures all messages originating from the global Gnutella network, modifies them if necessary, and forwards them inside the controlled domain.

Routing Control and Load Balancing. The AOL router represents the core mechanism for application of control. One of its main features is the ability to handle multiple protocols at the same time. To facilitate these different protocols in an effective and expandable way, the implementation of the router is divided into multiple, partly autonomous elements. In the current version of the AOL proxylet, two different mechanisms and protocols have been implemented: the Gnutella protocol version 0.6 [5] and an AOL intercommunication protocol, denoted as the *AOL-to-AOL protocol*. A major feature of the AOL-to-AOL protocol is the tunneling of other protocol messages between AOL proxylets, in our example the Gnutella packets. The current version of the AOL has the following routing capabilities:

- Routing of Gnutella packets
- Routing of AOL-to-AOL packets
- Routing between local Gnutella and AVP networks

The routing of Gnutella packets follows the specification of Gnutella version 0.6 but is significantly enhanced. The major enhancement lays in the "Probabilistic Routing" module, which drops broadcasted packets, e.g. Query messages, Ping messages, etc., based on a random value compared to a given threshold per connection. If the random value for a packet is larger then the configured threshold the packet is discarded. As a result, certain links become less loaded than others. Since the Gnutella protocol is based on event-triggered responses, discarding of a limited amount of packets doesn't sacrifice the file locating capability of the system when sufficient responses are still available, e.g. by receiving responses on multiple paths and from multiple sources. An example of probabilistic routing is depicted in Figure 4.

In this example four Peers (1, 2, 3, 4) are directly connected to an AOL proxylet. The proxylet has configured different threshold values for the links to Peer 2 (threshold is 0.3), Peer 3 (threshold is 0.6) and Peer 4 (threshold is 0.0). Peer 1 sends a message, e.g. a Query[1], to the AOL proxylet. The proxylet determines a random value of

[1] Query: Gnutella protocol message containing search criteria, used to search the P2P network for files [5].

0.5 for this packet[2]. Since the random value is smaller than the threshold value on the link to Peer 3, the AOL proxylet drops the packet along this connection whereas it keeps the packet on the links Peers 2 and 4.

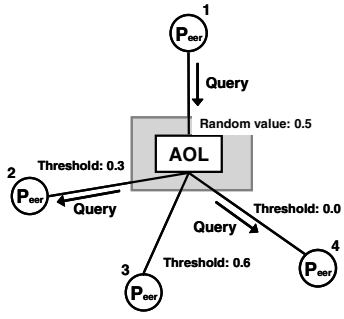

Fig. 4. Operation of the "Probabilistic Routing" feature of the AOL.

The AOL monitors and evaluates constantly the condition of the overlay, e.g. it measures and analyses the virtual link state or the virtual peer state. If these states degrade, the AOL may adjust the thresholds on the different proxylets and overlay connections. Through adaptive probabilistic routing, the AOL performs dynamic load control. Finally, by distinguishing messages between *implied events* (i.e. responses like Pongs and QueryHits) and *initial events* (i.e. requests like Pings and Queries), the probabilistic routing module makes sure superfluous traffic is not generated.

Topology Control. As mentioned earlier, topology control as enabled by the AVP enforces optimal P2P relations inside an overlay, based on a variety of metrics such as virtual peer state and virtual link state. The AOL proxylet achieves topology control by selectively setting up or closing connections to other AVPs and ordinary peers. By shaping the way peers are connected and communicate inside the overlay, an AVP can give it certain characteristics like better performance or greater stability.

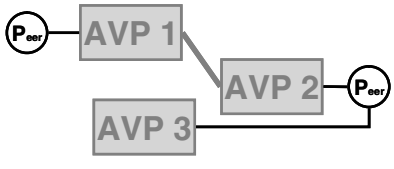

Fig. 5a. Dynamic overlay topology control (before)

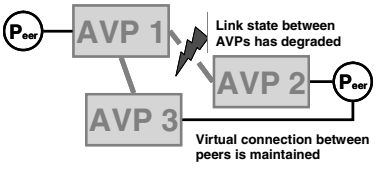

Fig. 5b. Dynamic overlay topology control (after)

Based on the virtual peer state, the AOL can initiate or terminate overlay connections between AOL proxylets in order to maintain good connection characteristics inside the overlay, e.g. more durable overlay connections. The virtual peer state can be monitored using parameters like the number of overlay connections maintained, routing capability of the peers, processing load etc. Let's examine the following scenario,

[2] Without the loss of generality, the random value is equally distributed in the interval form [0, 1].

as depicted in Figures 5a and 5b. Figure 5a shows three AVPs and two peers existing in that part of the overlay. The link between AVP 1 and AVP 2 is significantly degraded, affecting the stability and performance of the information exchange between the two peers.
AVP 1 discovers that the virtual link is degraded and decides to re-structure the overlay topology in order to maintain good overlay characteristics. Therefore, AVP 1 establishes a link with AVP 3 which is in proximity, cf. Figure 6b, and shuts down the overlay connection to AVP 2. This way, AVP 1 manages to maintain the connection between the two peers in the desired levels of connectivity and quality without any knowledge or action taken from their part.

This scenario shows how the AOL proxylets create and terminate overlay connections in order to enable dynamic topology management of the overlay by means of self-organization. It has to be noted that similar schemes where certain peers have some influence on the way the overlay is formed have been proposed elsewhere, like the Gnutella "Ultrapeer" concept, see Section 5. However, an AVP achieves improved adaptivity and flexibility due to its self-organization features and the coordination between multiple AVPs or multiple AOL proxylets.

Resource Management and Caching using the VCC. An AVP may contain a VCC (Virtual Control Cache) proxylet. Its task is to provide content caching on the application-level by maintaining often-requested content in close proximity. This feature is illustrated in Figure 6.

An AVP that has spawned and configured a VCC, controls a domain of peers by applying routing and access controls as shown previously. Each time a query is made by the peers inside the domain, it is only visible by other peers in the domain and the VCC. Moreover, the AOL does not forward the Query message outside the domain but modifies it accordingly, so that peers outside the domain see the AOL as the actual initiator of the query. This way, the peers inside the domain receive QueryHit replies only by other peers in the domain and by the VCC. If the content is available locally, a direct download connection may be established. Otherwise, the AOL upon receiving a QueryHit from outside the domain downloads the content on behalf of the VCC where it is ultimately stored. Then, the AOL sends a QueryHit to the peer that demanded the content pointing to the VCC. If the file is requested in the future it can be retrieved directly from the VCC.

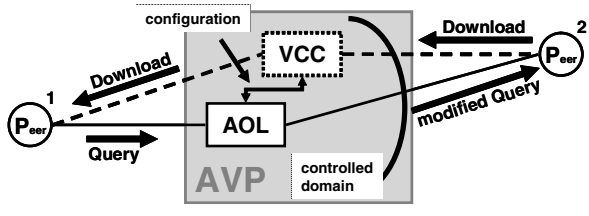

Fig. 6. Caching by the VCC proxylet.

Implementation details. AOL-to-AOL communication is achieved via TCP connections, although UDP may be used for shorter types of communications in order to decrease bandwidth use and other overheads. Table 1 below, lists all types of connections an AVP may currently implement.

The AOL protocol, used for AOL-to-AOL communications, allows the exchange of information vital for the communication and self-organization of the AVPs. This protocol was designed to be independent from existing protocols and realizes a simple and flexible way to communicate between AOL proxylets. An AOL protocol packet contains the following fields: *Source connection attributes* (AOL ID, IP address, Port number), *Destination connection attributes* (AOL ID, IP address, Port number, alternative route), *Type of payload*, *Payload* and *Priority*.

Table1. AOL-implemented types of connections.

Connection	Connection details
AOL-to-AOL	TCP/IP UDP for short information exchange planned
Gnutella-to-AOL	TCP/IP (Gnutella Protocol v0.6)
AOL-to-Gnutella	TCP/IP (Gnutella Protocol v0.6) UDP Ping for Availability Test
Telnet-to-AOL (Statistics interface)	Configuration / Statistic View
EEP-to-AOL	ALAN proxylet technology

Because of the "Type of payload" and "Payload" fields, AOL protocol packets allow the exchange of various types of data. Route advertisement and topology information are the most important for topology control. However, the payload may be an encapsulated Gnutella packet that will be tunneled through the AOL overlay.

All elements within this version of the AOL proxylet are connected via a modular design. So it is possible to include custom needs at different positions within the code. The main parts of the AOL proxylet are:

- Connection Manager (manages all connections, outgoing and incoming)
- Router (tree like connected modules, see Figure 8)
- Protocol (packet interfaces, protocol specific implementations)
- Configuration (create specific configurations for included modules)

Different connections can be added to the connection manager. The connection manager will manage all connection listeners as well as all active connections. If a connection is shut down, either by a user, due to a network error, or by the topology control mechanism, the connection manager will "clean up" this connection. Besides that, the connection manager provides information about all active connections, in order to support control loops. There are values like connection type, destination, simple statistics, errors logs kept within this module etc.

The router is the central module of an AOL proxylet. It is itself internally organized in a modular way. As illustrated in the Figure 7 below, the *root element* is the entry node of the entire router. Every connection module delivers the received packets to the root element of the router, where all active connections are added as possible routes. All elements after the root element process the packets according to their specific capabilities, e.g. Gnutella protocol routing or AOL protocol routing. Every element decides on its own whether it should handle a packet or not. The different elements are grouped according to their purpose (Gnutella Router, AOL proxylet

Router). At the end of each path the *send element* is found. The send element is either forwarding the packet to another peer or is able to relay the packet via tunnel to another AOL proxylet.

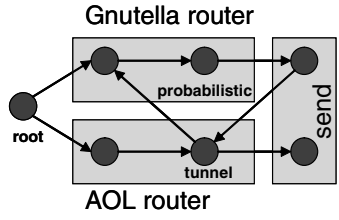

Fig. 7. Information flow in the router module of the AOL proxylet.

An AOL-specific module is the configuration module, which allows the transmission of commands via a telnet console from an administrator over the network. It will also contain in future versions an interface to receive configuration data through other AOL proxylets from the same AVP. This way, it will be possible for the AOL proxylet to establish a connection to another AOL proxylet or peer to support self-organization mechanisms.

5 Performance of AVP Concept

The performance of the AVP concept has to be characterized within the context of three areas: *a)* the overhead introduced by the AVP concept in relaying P2P protocol messages, *b)* the impact of the new routing strategies implemented by the AVP on network load, and *c)* the change of the overall performance of the P2P application, e.g. the boost of stability in the P2P overlay by using the AVP.

Overhead: For the intra-AVP and inter-AVP communication, cf. Section 4, the following protocol overhead is introduced due to packet encapsulation. Each relayed packet, i.e. Gnutella packet in the case of this prototype, will be extended by an additional 40 bytes header. The overhead introduced by the encapsulation is considerable but permits the distinct handling of messages without holding state information as for tunnels needed.

Impact of AVP Routing Strategies on Network Load: If necessary, the network load can be reduced immediately and locally by the AVP concept. If an intra-AVP or inter-AVP connection is overloaded, i.e. an AVP experiences reduced packet throughput on that connection, the AOL component of an AVP has two choices as examined in Section 4 (“Routing Control and Load Balancing“ and “Topology Control”): It may alter the path, e.g. set up a new virtual connection to another AOL which can relay the messages more appropriately, or it may drop distinct signalling packets randomly on a path. The latter mechanism can be applied in Gnutella without severe degradation of the service since Gnutella applies multi-path broadcasting instead of unicast communication.

P2P Service Performance and Overlay Stability: Previous studies have revealed a high variability of the original Gnutella overlay [3]. For example, the observed average overlay connection holding time in this study was 405 sec with a 90% interval of approximately 10^-1 sec and 10^3 sec, cf. Figure 8. This average is very short compared with typical overlay architectures such as VPNs (Virtual Private Networks). The high variability is, of course, a result of the key P2P characteristic that a peer may leave or join the network arbitrarily, a feature which should be maintained. The AVP concept is capable of decoupling the peer behaviour from the AVP behaviour and the AVP overlay behaviour in particular. The connection holding time of the intra- and inter-AVP connections is can be chosen to be much larger than that of ordinary peers. This way, the AVP concept offers always stable overlay connectivity for peers. The increased the stability of the AVP overlay, in turn, boosts the stability of the Gnutella service.

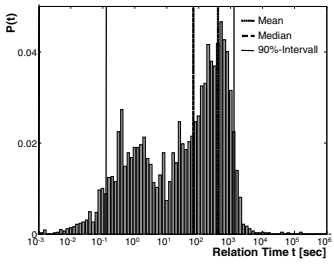

Fig. 8. Observed Gnutella Overlay Connection Holding Time Distribution [3].

In addition, an AVP is composed of multiple nodes, holding redundant information in an effective way. So it is possible that an AVP can leave the structure only with loosing a minimum of information, for example about routing or alternative available nodes. This way, the AVP concept may permit a more failsafe overlay and increases the resilience performance feature on P2P systems.

Finally, the performance and scalability of other P2P services was significantly boosted with the introduction of self-organizing hierarchy. Gnutella is the most famous example. With serious scalability problems and inefficient use of the resources in its first versions, it experienced serious performance gains upon the introduction of the “ultrapeers”. Similarly, Kazaa became the leading P2P file-sharing application because of the efficient use of network resources, that the users viewed as better performance, enabled by the use of the “superpeer” concept. The AVP allows amongst other things the creation of a two-level or even multi-level hierarchy, so we anticipate significant performance gains offered to P2P applications.

6 Related Work

Earlier work on the virtualization of resources and group management of has been investigated by Birman et al. [17] in the ISIS toolkit. While appearing similar in architecture that approach provides limited support for autonomous node operation and self-organization.

The inability of Gnutella and most other P2P applications to maintain topology and membership information in an efficient manner has been partly acknowledged by Limewire, developers of Gnutella client software, who proposed the concept of Gnutella "ultrapeers" [9]. This concept suggests the creation of a two-level node hierarchy inside the GNet, where the "ultrapeers", i.e. nodes possessing better networking capabilities and processing power, take charge of much of the load from the slower peers by maintaining more overlay connections. The decreased number of nodes responsible for message handling and routing, reduces the signaling traffic significantly, as well as it makes the Gnutella network more scalable. The concept of AVPs is similar to the "ultrapeers" since both apply a peer hierarchy and reduce signaling traffic. AVPs differ from "ultrapeers", however, because of their overlay load control capability and adaptivity to the underlying network structure. The well-known Kazaa P2P filesharing service [18] applies a concept similar to "ultrapeers". In Kazaa these distinct nodes are denoted as "superpeers".

The Gnutella2 framework [10] extends the "original" Gnutella protocol beyond file sharing. The Gnutella2 architecture promotes "leaf mode" node operation, supports reliable UDP communication, enables bandwidth management schemes, and aims to create an efficient, self-organized P2P network. However the access control features in Gnutella2 are limited.

In [23], the authors suggest the use of dedicated P2P application "boosters" at the ingress/egress links of administrative domains with the aim of improving scalability and reducing signaling traffic of P2P file-sharing services. While they envisage multiple scattered boosters to create a two-level hierarchy achieving the aforementioned goal, their concept dramatically lacks flexibility. The formation of sets (groups) of nodes, based on application level and network level information, is examined in [24]. The authors describe many possible uses of that service, with P2P networking one of them. While in areas such as reliable multicast such a service may be beneficial, we argue that it is very demanding in terms of underlying network infrastructure (it assumes Ephemeral State Processing capabilities) and considerably inflexible when applied to P2P. Both approaches, [22] and [23], however, are limited in their capability to adapt toward varying network load condition in contrast to features of the AVP concept.

The OverQoS architecture [25] aims to provide QoS services for overlay networks. Dedicated OverQoS routers are placed at fixed points inside an ISP's (Internet Service Provider) network and connected through overlay links. The aggregation of flows into controlled flows of an overlay enables this architecture to adapt to varying capacities of the IP network and ensure a statistical guarantee to loss rates. This OverQoS approach complements and extends the limited load control provided so far in the AOL proxylet. However, it lacks any adaptivity to the varying network topology as addressed by the AVP.

Resilient overlay networks (RONs) [24] provide considerable control and choice on end hosts and applications on how data can be transmitted, with the aim of improving end-to-end reliability and performance. However, RONs are mostly restricted within single administrative domains..

Finally, the characterization and quantification of the connectivity topology characteristics is widely investigated. Power-law type or small-world topologies are widely identified with P2P systems [11,12] and are extensively examined as part of the AVP concept. Distributed Hash Tables have recently been widely introduced for search of information in P2P systems [13]. Examples include Pastry of Microsoft Research

[14], Chord from UC Berkeley and MIT [15] or CAN (Content Addressable Networks) form ICSI at UC Berkeley [16]. Measurement of duration and size of P2P connectivity in Gnutella overlays as well as topology control have been investigated in [3]. Understanding of P2P traffic volume and connectivity behavior is essential for network planning, traffic regulations, security assurance, performance guarantees and, possibly, revenue generation.

7 Conclusions

We have presented a new framework and an implementation technique for a flexible management of peer-to-peer overlays. The framework provides means for self-organization to yield an enhanced flexibility in instantiating control architectures in dynamic environments, which is regarded as being essential for P2P services.

Application level active networking (ALAN) was chosen as a natural vehicle to enable evolutionary adaptation on the application layer. P2P services can be predominately viewed as overlay networks which lend themselves to application level management and control in order to maintain their beneficial characteristics. In particular, the incorporation of ALAN maintains the decoupling of network and application layers while providing operational support at the same time. Furthermore, the ALAN infrastructure enables the AVP to respond to changing network conditions on the time scales that match network scales as well as native P2P application behavior by decoupling the P2P behavior from the AVP behavior.

The proposed concept relies on Active Virtual Peers as the main building block. The presented AVPs for Gnutella implement means for overlay control with respect to access, routing, topology forming, and application layer resource management. The AVP concept not only allows the combination of algorithms and techniques with proven merit that address only individual issues but allows them to operate over a flexible and adaptive framework. The significance of this approach lies with the expandability and adaptivity of the system as P2P services evolve.

In order to facilitate a more complete control, additional AVP features will be implemented in future work such as mechanisms that allow monitoring of overlay traffic and strategies for topology self-organization.

References

1. A. Ghosh, M. Fry, J. Crowcroft: “An architecture for application layer routing”, Active Networks, LNCS 1942, H. Yasuda, Ed 2000, pp 71-86 Springer.
2. M. Fry, A. Ghosh: “Application level active networking”, Computer Networks, 31 (7), 1999, pp. 655-667.
3. H. De Meer, K. Tutschku, P. Tran-Gia: ``Dynamic Operation in Peer-to-Peer Overlay Networks'', Praxis der Informationsverarbeitung und Kommunikation, (PIK Journal), Special Issue on Peer-to-Peer Systems, Volume 26 No 2, pp 65-73, June 2003.
4. H. De Meer, K. Tutschku, “Dynamic Operation in Peer-to-Peer Overlays'', poster session, proceedings of 4th Annual International Working Conference on Active Networks, ETH Zurich, Switzerland, 4-6 Dec 2002

5. T. Klingberg, R. Manfredi, “The Gnutella protocol version 0.6 draft”, Gnutella developer forum (http://groups.yahoo.com/group/the_gdf/files/Development/), 2002
6. Anonymous, “Gnut: console Gnutella client for Linux and Windows”, http://www.gnutelliums.com/linux_unix/gnut/, 2001
7. B. Fortz, M. Thorup, “Internet traffic engineering by optimising OSPF weights”, Proceeding of IEEE INFOCOM 2002, pp 519-528, 2000
8. X. Xiao, A. Hannah, B. Bailey, S.Carter, L.M. Ni, “Traffic engineering with MPLS in the Internet”, IEEE Network Magazine, vol. 14, no. 1, pp 28-33, 2000
9. A. Singla, C. Rohrs, “Ultrapeers; another step towards Gnutella scalability”, Gnutella developer forum (http://groups.yahoo.com/group/the_gdf/files/Proposals/Ultrapeer/ Ultrapeers_1.0_ clean.html), 2002
10. M. Stokes, “Gnutella2 specification document – first draft”, Gnutella2 website (http://www.gnutella2.com/gnutella2_draft.htm), 2003
11. A.L. Barabasi, R. Albert: “Emergence of Scaling in Random Networks”, Science Vol. 286, 1999.
12. E. Cohen, S. Shenker: “Replication Structures in Unstructured Peer-to-Peer Networks”, ACM SIGCOMM, Pittsburg 2002.
13. H. Balakrishnan, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica: “Looking up Data in P2P Systems”, Communications of the ACM, Vol 43, No. 2, February 2003.
14. M. Castro, P. Druschel, A.M. Kermarrec, A. Rowstrom: “One ring to rule them all: Service discovery and binding in structured peer-to-peer overlay networks”, SIGOPS, France, September 2002.
15. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan: “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, ACM SIGCOMM`01, San Diego, September 2001.
16. S. Ratnasami: “A Scalable Content-Addressable Network”, PhD Thesis, UC Berkeley, October 2002.
17. K. P. Birman et al, Isis – A Distributed Programming Environment: User’s Guide and Reference Manual Verion 2.1, Dept. Computer Science, Cornell Univ., Ithaca, N.Y., Sept. 1990.
18. Kazaa Media Desktop http://www.kazaa.com.
19. Q. Lv, S. Ratnasamy, S. Shenker: “Can Heterogeneity Make Gnutella Scalable?”, Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002.
20. MetaMachine Inc. http://www.edonkey2000.com
21. I. Clarke, S. Miller, T. Hong, O. Sandberg, B. Wiley: ”Protecting Free Expression Online with Freenet” IEEE Internet Computing, Vol. 6, No. 1, pages 40-49, January/February 2002.
22. S. Rooney, D. Bauer, P. Scotton, “Efficient Programmable Middleboxes for Scaling Large Distributed Applications”, IEEE OpenArch 2003, 2003
23. A. Sehgal, K.l. Calvert, J. Griffioen, “A Generic Set-Formation Service”, IEEE OpenArch 2003, 2003
24. D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, “Resilient Overlay Networks”, Proc. 18th ACM Symposium on Operating Systems Principles, 2001.
25. L. Subramanian, I. Stoica, H. Balakrishnan, R. Katz: ”OverQoS: Offering Internet QoS Using Overlays“, Proc. of 1st HotNets Workshop, October 2002.
26. A. Galis, S. Denazis, C. Klein, C. Brou, “Programmable Networks and their Management”, Artech House, 2004