Peer-to-Peer driven Software Engineering considering Security, Reliability, and Performance

Kristian Beckers
paluno - The Ruhr Institute for Software Technology – University of Duisburg-Essen, Germany
Email: {Kristian.Beckers}@paluno.uni-due.de

Stephan Faßbender
paluno - The Ruhr Institute for Software Technology – University of Duisburg-Essen, Germany
Email: {Stephan.Fassbender}@paluno.uni-due.de

***Abstract*—Internet-scale applications require scalability that peer-to-peer (P2P) architectures provide. Traditional software engineering processes start with requirements and move on to architectures, software design, implementation, and testing. Choosing a P2P architecture, however, has significant constraints on the requirements of a given software engineering process in terms of security, reliability and performance. In addition, requirements for P2P architectures have to be expressed in notions of network engineering, because these architectures rank from the application layer to the IP network layer. Thus, engineering P2P systems is a cross-disciplinary task between software and network engineers.**

We explain the ramifications P2P applications have on requirements of a given software engineering problem. A structured method supports software engineers in understanding the constraints of different kinds of P2P architectures and protocols on requirements. In addition, we present patterns of how the requirements have to be expressed, so that they contain all required information for network engineers that implement the P2P architectures.

***Keywords*-P2P, security, requirements engineering, software architecture, performance, reliability, network engineering**

I. Introduction

Aligning software systems to meet requirements is a challenging task, which is even more difficult when a Peer-to-Peer (P2P)-based software system shall be developed. For example, the effects of churn, the random leaving or joining of peers in the system, can cause data loss. If a requirement is that no data loss shall occur in the system, then churn presents a constraint on this requirements. A software engineer can design a countermeasure, if she is aware of the constraint. This, however, is difficult, because numerous constraints are caused by attributes of the P2P protocol or even the network layer.

We present a pattern-based method to identify existing constraints in a P2P-based software system. An initial pattern considers all layers of a P2P architecture and offers more detailed patterns for, e.g., P2P protocols. The instantiation of these patterns enables an analysis of the system's constraints and reveals the information in which layer each constraint originates.

The following research questions shall be answered with our approach: Has the decision to use a P2P architecture constraints for previously elicited security requirements? Has also the decision for a specific P2P protocol constraints on security, performance, and reliability requirements? Is it possible to extract a pattern for security, reliability, and performance requirements for P2P architectures? Which information does a P2P security, reliability, and performance requirements must contain, in order to support the choice of an protocol?

The rest of the paper is organized as follows: Sect. II presents a P2P system pattern and Sect. III, Sect. IV, and Sect. V present templates that specify elements of the pattern. Sect. VI shows a method that specifies how to use the pattern and the templates to analyze the ramifications of P2P systems on quality requirements. An example of the method is presented in Sec. VII. Sec. VIII presents related work. Section IX concludes and gives directions for future research.

II. A Pattern-System for Analyzing the Ramifications of Peer-to-Peer Architectures on Quality Requirements

Peer-to-Peer (P2P) architectures design distributed systems, in which identical software works on every peer. These systems are distributed without any centralized control or hierarchical organization that form a self organizing overlay network on top of the Internet Protocol (IP) [1], [2]. An *Overlay* is a network which is built on top of one or more existing networks. This adds additional properties to the underlying network, e.g., more efficient search of data or adding locality information to peers. It provides a communication infrastructure for all peers in the P2P architecture [3]. P2P systems can be clients and servers at the same time. They provide access to their resources by other systems and support resource sharing on an internet scale. This requires fault tolerance, self-organization, and significant scalability properties [2]. One of the obstacles a P2P system has to overcome is *Churn*, the joining and leaving of peers in a P2P architecture without prior notification. Two fundamen-

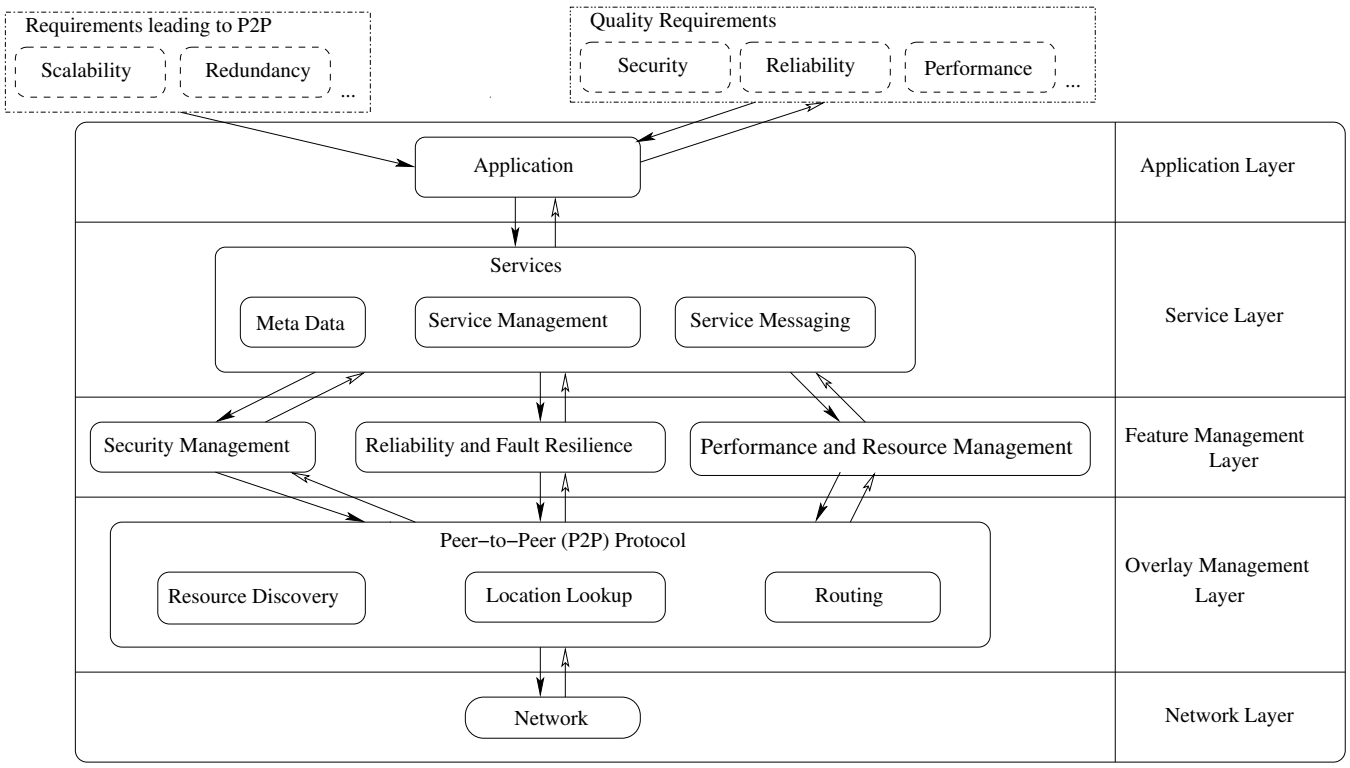

Figure 1. A P2P System Analysis Pattern

tally different types of P2P systems exist. *Structured P2P systems* organize peers via an algorithm, which leads to an overlay with specific properties. They are often based upon a distributed hash table. *Unstructured P2P systems* organize their peers in a random graph in a flat or hierarchical manner (e.g. Supernodes exist that outrank normal nodes). They are based upon techniques like flooding, random walks etc. [2]. Hence, P2P architectures are fundamentally different from *stand alone* or *client server* architectures. We present a method that supports the identification of constraints that P2P protocols cause on requirements. The method shall be used in a given software development process after the requirements for a software have been elicited and one or more requirement(s) lead to the design decision to rely on a P2P protocol within the software-to-be. We propose a P2P-pattern for identifying constraints on quality requirements caused by P2P architectures.

The pattern contains the layers of a P2P architecture, shown in Fig. 1. The instantiation of the pattern begins with one or more requirement(s) that lead to the decision to use a P2P protocol. For example, scalability or redundancy requirements can result in this decision. The pattern is based upon the P2P architecture from Lua et al. [2], which is derived of a survey of existing P2P systems. The requirements and the layers of a P2P system are connected with arrows. Arrows with a black arrowhead present the collection of information through the P2P layers, getting more detailed on every level. The white arrowheads present constraints that are derived from all the information collected before.

The *Application Layer* concerns applications that are implemented using the underlying P2P overlay. For example, a Voice-over-IP (VoIP) application. The *Service Layer* adds application specific functionality to the P2P infrastructure. For example, for parallel and computing intensive tasks or for content and file management. Meta-data describes what the service offers. For instance, content storage using P2P technology. Service messaging describe the way services are communicating. The *Feature Management Layer* contains elements that deal with security, reliability and fault resiliency, as well as performance and resource management of a P2P system. All these aspects are for maintaining the robustness of a P2P system. We renamed the resource management from the original architecture from Lua et al. [2] into performance and resource management, because the quality of the access and distribution of resources is the main performance property of P2P architectures. The *Overlay Management Layer* is concerned with peer and resource discovery and routing algorithms. We aim to explain these in terms of software engineering and not network engineering. The *Network Layer* describes the ability of the peers to connect in an ad hoc manner over the internet or small wireless or sensor-based networks.

For reasons of space we focus on the Feature Management, Overlay Management and Network Management Layer in this work. We present templates for these layers in the following chapters, which specify these parts of the pattern in more detail. Templates can also be instantiated.

III. Feature Management Templates

The feature management part of the pattern consists of three templates for performance and resource management, reliability and fault resilience, and security management. We present pattern for these in the following subsections.

A. Performance and Resource Management

Performance measures in P2P systems are based upon Quality of Service (QoS) constraints [5]. P2P systems are designed to work on an internet scale. They consist of numerous peers communicating via protocols using the internet in particular IP, which is based upon the transmission of small packets. Communication between peers is essential in a P2P systems. This is the only link peers have to one another. The P2P system is not functional anymore, when the link breaks. The P2P overlay uses IP as an infrastructure for this communication. The quality of the packet delivery of IP over the internet is the so-called *Quality of Service (QoS)*. This transmission performance is measured in *delay (latency)*, *jitter* and *packet loss rate* [6]. Delay is the end-to-end delay of a data transmission. IP packets are often routed on different path throughout the internet. Jitter is difference in transmission time from sender to receiver of different IP packets. Packet loss rate is the number of lost packets of a transmission within a specific timeframe.

Requirements for these different types of QoS arise from different kinds of applications. Peterson and Davie [4] published a taxonomy of applications and their relation to QoS requirements, shown in Fig. 2. Applications that have QoS requirements are so-called *real time* applications, otherwise these are *elastic* applications. This means these applications take no harm, when a packet arrives late or out of order, e.g., file sharing applications. Real time applications can be *tolerant* to packet loss, e.g., a Voice-over-IP (VoIP) application is able to compensate for a missing or late packet via interpolating its content from the surrounding samples. Otherwise it is a so-called *intolerant* real time application. Real time applications that are adaptable to QoS restriction during run time are called *adaptive*. These are either adaptive to *delay* or to the available *rate* (bandwidth) with which packets are transferred. For example, a video playback application can switch to a lower resolution of the video, when the bandwidth decreases. The classification of an application is important when eliciting requirements for a p2p application. For example, a real time application has to have a constraint on all QoS measures. In addition, adaptive applications can vary in their constraints. Thus, the QoS have to be given in intervals.

We accompany the pattern presented in Sect. II with templates. The first template is the performance and resource management template for the Feature Management Layer:

Name State the identifier for the performance element of the Feature Management Layer.

Description Describe the characteristics of the service relevant for QoS, e.g., if the service has real time QoS constraints.

Relation to Services State the relation to the service(s) of this element of the Feature Management Layer.

Relations to Overlay Management and Network Layer Name the relation the application has to the Overlay and subsequent to the Network Layer. For example. is the Overlay used to transport data or simply used for signaling of e.g. a VoIP application.

Resources State the resources that this service manages.

Determine QoS Type Conclude from the information collected in the previous parts of the template the QoS type.

Constraints List the constraints resulting from the chosen P2P protocol and network protocols. Use the information in the instantiated overlay management and network layers.

After determining the QoS type using the pattern Tab. I supports the identification of the relation type between the QoS type and QoS relevant specification elements. These elements need to be specified for a specific QoS type. A "+" in Tab. I marks the QoS elements that have to be specified for the QoS application type. However, the free cells of the table do not imply that a QoS element is not relevant for a QoS application type.

B. Reliability and Fault Resilience

Reliability of P2P systems is measured in terms of availability [7]. This means the percentage of uptime of the system. P2P systems rely on the ability that peers can communicate to each other. Messages have to be relayed throughout the network, peers have to be found and data has to be retrieved. If one of these is not possible, the reliability of the system is not there. Based upon the type of applications further functionalities apply. For example, in a real time application the ability to create and maintain communication sessions is necessary. In the case of VoIP applications the number of completed calls is a metric for reliability [7].

Increasing reliability of a P2P can be increased by adding redundancy and diversity. This measure can be integrated into all levels of the P2P architecture. For example, Fessi et al. [8] presented a VoIP communication infrastructure that relied on a structured P2P system and has a fallback server for the VoIP signaling. This central server provides redundancy in case the P2P VoIP signaling fails, in this case the peers can contact the fallback server. In addition, this is also diversity, because the P2P system is essentially

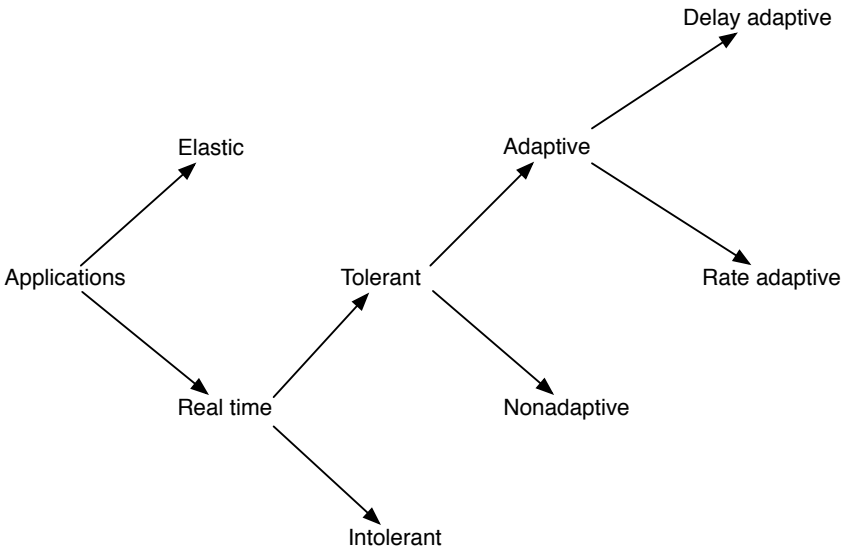

Figure 2. Taxonomy of QoS types taken from Peterson and Davie [4]

Table I
RELATIONS BETWEEN QOS APPLICATION TYPES AND REQUIRED SPECIFICATION ELEMENTS

QoS Type	Bandwith	Packet Loss Rate	Delay	Jitter	Compensation Rate	Adaptation Rate
Application	+					
Elastic		+				
Real Time		+	+	+		
Tolerant		+	+	+	+	
Adaptive		+	+	+	+	+

transformed into a client server system, when the fallback server is used [8].

One strength of P2P systems is their ability to recover from the failure of several peers. For example, every peer has several links to other peers. Should one peer fail, data can easily be routed through the remaining peers. In addition, data can be replicated as part of a peer-to-peer protocol. Thus, numerous replica sets exist on different peers of data. This can significantly decrease the effects of churn [8].

We accompany the pattern presented in Sect. II also with a template for reliability and fault resilience:

Name State the identifier for the performance element of the feature management layer

Description Describe the characteristics of the service relevant for availability, e.g., if the availability depends upon established communication sessions or concluded communication sessions

QoS Type State the QoS Type from the Performance and Resource Management template.

Relation to Services State the relation to the service(s) of this element of the feature management layer

Relations to Overlay Management and Network Layer Name the relation the application has to the Overlay and subsequent to the Network Layer considering reliability. Answer the question, which relations are essential to the operation of this element. For example. when the overlay used to transport VoIP signaling data the relation states that signal messages. are delivered to recipients.

Availability types Conclude from the information collected in the previous parts what are the essential elements that can specify availability criteria for the availability of this element.

Redundancy and Diversity State the redundancy and diversity options offered by this component. This information is important for choosing a P2P protocol in the Overlay Management Layer. The reason is that properties of the protocol support these features. For example, redundancy can be integrated into a P2P system by making data replication to neighboring peers part of the protocol. Moreover, diversity can be realized via Superpeers in an unstructured P2P system.

Table II
RELATIONS BETWEEN QOS TYPES AND AVAILABILITY SPECIFICATION ELEMENTS

QoS Type	Peer Connections	Data Transfers	Communication Sessions	Compensations	QoS Adaptations
Application	+				
Elastic		+			
Real Time			+		
Tolerant			+	+	
Adaptive			+		+

Constraints List the constraints resulting from the chosen P2P protocol and network protocols. Use the information in the instantiated overlay management and network layers.

Tab.. II supports the identification of an availability type. A "+" in Tab. II marks the availability element that have to be specified for a QoS type. For example, the QoS type However, the free cells of the table do not imply that a availability type is not relevant for a QoS application type.

C. Security Management

Dynamics in P2P systems cause several difficulties for security. For example, churn in P2P systems makes guarantees that a specific file reaches its target difficult. Central entities alone in the routing path can increase the effects of churn. However, these become single points of failure and decrease the scalability of the system. Moreover, P2P systems aim to operate at an internet scale. Numerous nodes in these systems must be regarded as not trustworthy [9].

Gheorghe et al. [10] identified a number of attacks, classified by targets in the P2P systems. The attacks are each classified against a P2P-specific security goal:

Node Autonomy Each node should be able to perform its functions as a peer and its autonomous functionalities. In *collusion attacks* a malicious node comprises other nodes to carry out a correlated attack. The difficulty in preventing this attack is to identify the node that is causing the attack. *Neighbor selection attacks* enable attackers to control the selection of peers of several nodes. This can affect the overlay communications and control traffic. *Membership and Eclipse Attacks* effect the way nodes are admitted into the overlay. An eclipse attack is a specific kind of membership attack, where an attacker gains control over a part of the P2P systems. Hence, the attacker is able to drop or reroute every message in this part of the P2P system.

Confidentiality and Integrity *Forgery Attacks* break the confidentiality and integrity of data in the P2P system. These attacks focus on tampering with data that is transmitted in the P2P system. In *pollution attacks* attackers mix junk pieces of data into the P2P system. This can have significant effects on the QoS of a P2P system. These attacks also scale with the size of the P2P network.

Node Authentication These attacks focus on the reputation systems of a P2P network. These networks often keep a list of how peers perform their expected functionality, e.g., forward data to other peers. The reputation of a peer rises if it performs its functionality as expected. Otherwise the reputation decreases. In *Sybil Attacks* an attacker has found a way to temper with peers reputations.

Dependability and Availability *Denial of Service (DoS) Attacks* consist of numerous malicious nodes, which send excessive amounts of requests or duplicate packets to peers. This exhausts the resources of the P2P system. *Omission Attacks* are the opposite of DoS attacks. These cause that packets or data is not sent further through the P2P system or that peers do not function according to the P2P protocol anymore.

Security in P2P systems also depend upon the QoS application taxonomy introduced in Sect. III-A. Security of systems have been studied for elastic P2P applications, mostly file sharing applications [9]. Elastic applications have to be protected against the attacks above. In these systems the functionality of the P2P systems has to be kept intact. Real time applications have significant QoS constraints. Attackers of these systems can already succeed with delaying a message instead of deleting it. For example, video streaming applications demand low delay and high constant bandwidth [9]. Thus, real time applications are more susceptible to attacks on availability. In addition, unsolicited communication, e.g., Spam over Internet Telephony (SPIT) in real time communications is a bigger threat, because these cause a ringing tone or play an offending video. SPIT is difficult to counteract, due to decisions that have to be done in real time. Dynamic membership and frequently changing routing path in P2P systems cause that countermeasures are difficult to apply. Moreover, spammers can harvest target locations by simply trying lookup requests in the P2P system [9].

Privacy concerns are also an issue in P2P systems. For example, a malicious node can log the routing path of VoIP messages in a P2P system. Thus, the attacker can establish a list of who called whom and further more. This is considered personal information and is hidden to users in common

telephone systems [9]. In addition, when peers can figure out the routing paths, they might also figure out the overlay node IDs. Hence, an attacker might also learn the IP address of the peer. Moreover, the IP can then be linked to a person and this can cause a significant privacy breach.

Thus, when designing a P2P system the usage of these patterns and their protection has to be a key focus of the P2P engineering efforts. In real time systems these patterns have to able to fulfill the QoS requirements. The specification of a P2P system should evolve around how well a P2P system has to perform. The template for security is shown in the following:

Name State the identifier for the security element of the feature management layer

Description Describe the characteristics of the service relevant for security.

QoS Type Specify the QoS type according to the Performance and Resource Management template. According to the QoS type countermeasures to specific attacks will become relevant. For example, unsolicited communication has to be prevented in real time applications, e.g., in the form of SPIT for VoIP.

Relation to Services Describe the security requirements for the service(s).

Relations to Performance and Reliability The security of a P2P system is related to the performance and reliability of the P2P system. Hence, the information in these templates has to be considered (see Sect. III-A and Sect. III-B. These are relevant because availability goals, which security and reliability share, are expressed in QoS terms. Hence, specify dependability and availability goals derived from the other templates of the features management layer.

Relations to Overlay Management and Network Layer Describe the relation to the underlay, e.g., which information is transported between Overlay and Feature Management Layers.

Security goals Specify the security goals availability, integrity and confidentiality. In addition, the P2P specific security goals node authentication and node autonomy.

Constraints List the constraints resulting from the chosen P2P protocol and network protocols. Use the information in the instantiated Overlay Management and Network layer.

IV. Overlay Management Template

We describe important P2P protocol attributes derived from [2], [11], [3]:

Churn The leaving of peers of the P2P system can cause a loss of data, which is routed through this peer or stored in it. The joining of peers in large numbers at a given point in time can affect the QoS of the system, e.g., because routing path for data increases or existing data has to be distributed to more peers.

Overlay-Underlay Relation In a P2P network there is no direct relation between the network layer and the peers. This can cause performance issues, e.g., on routing of data in the P2P system. Two peers might be located next to each other in the network, but in the P2P network data might be routed through several other peers that are further away. This also effects security, because the more peers route the data, the more gather knowledge about it. Reliability is affected because of the increased network load that can cause an instability of the network. Moreover, the more peers the data is routed through, the greater is the chance that it is lost due to churn.

Structured P2P Architecture Structured P2P architectures arrange all the peers in a graph. Maintaining this graph or distributed hash table is difficult, because of churn. These architectures have to use a considerable amount of resources to keep the graph intact. For instance, chord uses an algorithm that just checks if the ring and the routing tables on each node are intact. The benefit is the level of control in the architecture is higher than in an unstructured P2P architecture. There is a structured way to interact with a specific node. For instance, the hash function in a distributed hash table provides information how to find a specific peer. The performance of a structured p2p architecture for distributing content is assumed to be slower than the performance of a unstructured p2p network, because the path for finding specific data and routing it is predetermined in a structured p2p architecture. Thus, if many user request the same data, the performance of the peers in that route take a performance hit. On the other hand finding specific and rare data is more likely to succeed in a structured P2P architecture.

Unstructured P2P Architecture Unstructured P2P architectures work via random walks, e.g., flooding techniques. These cause a performance load upon the network, because every peer in this architecture always sends messages to many known peers. On the other hand this allows to use numerous peers for a task, e.g., the sharing of data. Thus, also the workload gets distributed. Unstructured P2P network have to deal with the problem of clustering. Each peers learns the location of a few other peers in unstructured P2P architectures and the peers learns from these peers the location of further peers. When all peers do not learn the location of others peers they form a cluster that cannot contact the remaining peers in the architecture.

Flat P2P Architecture P2P architectures have either the constraints of structured or of unstructured P2P architectures.

Hierarchical P2P Architecture In these architectures peers with different properties are introduces to change the constraints of structured or unstructured P2P

architectures. For instance, a Superpeer in a structured P2P architecture can be a server that never leaves the P2P architecture. Thus, eliminating the effects of churn on, e.g., some specific data. In an unstructured P2P architecture a Superpeer could eliminate the clustering problem by realizing which clusters exist and provide a connection between these.

We also present a template for P2P protocols:

Name State the name of the P2P protocol

Description Types Describe the type of P2P protocol: is it a structured or unstructured P2P protocol and the used hierarchy: is it flat or hierarchical.

Description Mechanisms Describe how the resource discovery, location lookup and routing works (see Sect. II).

Churn State the QoS Type from the Performance and Resource Management template in relation to the effects of Churn in the protocol.

Relations to the Network Layer Name the Overlay-Underlay-Relation the P2P protocol has towards the Network Layer. For example. the overlay uses the TCP protocol on the network layer to transport network packets.

Extensions of the P2P protocol Was the P2P protocol extended, e.g., in order to improve a specific functionality, for example, increase routing performance via a modification of the algorithm in an structured P2P protocol.

V. Network Template

The Network Layer enables peers to send messages to each other over the internet. The packet-based IP network is used via UDP [12] or TCP [13] protocols. TCP contains several quality functionalities, e.g., checks that a packet arrived at its destination, while UDP lacks these. However, due to UDPs simplicity the data transfer is considerably faster. This is often required for real time applications, e.g., VoIP. We present a template for the p2p network layer:

Name State the name of the network layer

Protocol State the used network protocol: UDP or TCP.

Description Describe the characteristics of the network layer. Are the functionalities of the used protocol enhanced, e.g., with a software providing functionality on top of the protocol.

VI. A Method for Eliciting the Constraints a P2P Architecture causes on Quality Requirements

The method for using the pattern depict in Fig. 1 starts with requirements leading to the decision to use a p2p protocol. The method follows first the arrows with the black arrowheads to the Network Layer. Hence, identifying and collecting constraints on the requirements. Afterwards the method follows the white arrowheads from the Network Layer back to the requirements. In this part of the method possible conflicts with requirements are identified.

Specify the application The first step requires a set of requirements as an input and a functional description of the software-to-be. The result is a description of the application.

Specify each Service The application is refined into services. These have each meta data that describes their functionality, a management element that provides the functionality and a messaging component to exchange data with, e.g., distributed parts of the service. For example, a remote data storage. The service level can also add further requirements.

Describe the Feature Management for each Service Specify the features of a service containing security, reliability and fault resiliency, and performance and also resources. We represent these each as specific templates that are instantiated using the requirements from the previous steps.

Choose a P2P Protocol Choose a suitable P2P protocol and instantiate the P2P protocol template. The template requires a description of the resource discovery, peer location lookup and routing functionalities, as well as the protocol types.

Specify the Network Layer Instantiate the template for the network layer. List the type of protocol used on top of IP ,e.g., UDP or TCP and a possible enhancement of the features of these protocol in a messaging layer on top of these protocols.

Consider the Network-Overlay-Relation Describe the effect the chosen type of network protocol has on the P2P protocol. For example, using UDP increases the transmission speed, but the protocol has no mechanisms to ensure a packet is delivered. These effects have to be noted and resulting issued documented and especially the layer on which these occurred. These information has to be added to the p2p protocol template.

Identify constraints for the Feature Management List the constraints coming from the overlay and the network on the features for security, performance and reliability in the templates.

Analyze the Effects on the Services State the relation that the identified issues on the feature layer have on the services.

Analyze the Effects on the Application State the relation that the identified issues on the services have on the application.

Describe the Constraints om Requirements Describe the resulting constraints the application now has on the quality requirements, e.g., security, performance and reliability. If it is not possible to fulfill the requirements considering the constraints the software has to be changed. Furthermore, mechanisms could be considered in the Feature Management Layer, e.g., to change the P2P protocol or to implement further mechanisms in the Network Layer or even to exchange the protocol.

Identify were the issue occurred and re-iterate these parts of the pattern.

VII. Example

We use a distributed early warning system on an internet-scale that is able to prevent, e.g., cyber warfare, for example in the form of DDOS attacks as an example for this work. An intrusion detection system (IDS) consists of at least the following components *sensor*, *analyzer*, and *manager* according to RFC 4765 [14] the intrusion detection message exchange format. The RFC was written by the IETF Intrusion Detection Working Group (idwg).

A sensor collects data from a data source. For example, a probe in a network that collects packets from a specific network protocol. The sensor looks for specific events or patterns and when these occur it sends collected data to the analyzer. This component analyzes the collected data for signs of unauthorized or undesired activity or for events that might be of interest. The manager component controls all components of an IDS, configures sensors and analyzers, and also the event notification management, data consolidation, and reporting.

Figure 3 presents an instantiation of the P2P systems pattern. The requirement leading to the decision of using a P2P protocol is the scalability demand of the IDS application. In addition, the system has several quality requirements. The system has the security requirement to be available at all times and to resist a distributed denial of service (DDoS) attack. The system has also reliability as a quality requirement. The performance requirement of the system is that the system shall be initialized fast. The application consists of three basic types of services. These are the sensor, analyzer, and manager service according to the IDS description. These services rely upon the security management component for DDoS detection. This means detection DDoS in progress on the network and against the IDS. In addition, a reliability component provides the functionality that a certain stability in the network is ensured. The instantiated performance and resource management component provides the functionality for a high speed alert notification. We choose the Gnutella [15] P2P protocol, an unstructured P2P system with a flat hierarchy.

The Gnutella network is based upon some loose rules that manifest in a number of communication protocols. **Join the network:** Peers send PING messages to announce their presence. These PING messages are forwarded throughout the network. Peers reply with PONG messages that state their IP address and additional information. **Locate data:** In order to locate a data item peers flood the network with QUERY messages. Peers that have the data respond with a QUERY RESPONSE message that contains the information, where to download the data. **Transfer files:** GET and PUSH messages initiate file transfers between peers.

We choose to use the UDP protocol in the network layer. A messaging layer between the UDP layer and the overlay management is not added in this system.

We present instantiations of the Feature Management templates for performance, reliability, and security. For reasons of space we present only the templates for the sensor service. In addition, we left out redundant or empty fields in the template.

We present the instantiated performance template:

Name Alert Notification
Description Real time constraints
Relation to Services The sensor service sends alert notifications and the analyzer receives these.
Relations to Overlay Management and Network Layer The Overlay Management has to transport the messages using the Network Layer.
Resources None.
Determine QoS Type Real Time Application
Constraints UDP protocol gives no guarantees that a package is delivered, performance is improved using UDP then using TCP, Gnutella cannot guarantee that an alert message is delivered in time,

We present the instantiated reliability template:

Name Reliability
Description Availability depends upon the ability to transfer alert messages
Relations to Overlay Management and Network Layer The Overlay and Network Layers are used to transport the alert messages.
Availability types The availability depends upon the reliability of the P2P network. In this case it is an unstructured P2P system, hence the delivery of an alert message cannot be guaranteed.
Redundancy and Diversity The Gnutella protocol does not offer redundancy or diversity. These features would have to be implemented in the Feature Management Layer.
Constraints The protocol does not offer redundancy or reliability features in its original form. In addition, neither the Gnutella nor the UDP protocol can guarantee the delivery of an alert messages.

We present the instantiated security template:

Name Distributed Denial of Service Detection
Description The availability security goal has to be fulfilled at all times.
QoS Type Real Time application.
Relation to Services The sensor service sends alert notifications, the analyzer receives these.
Relations to Performance and Reliability The availability condition is expressed within the performance conditions.
Relations to Overlay Management and Network Layer Neither the Overlay Management nor the Network

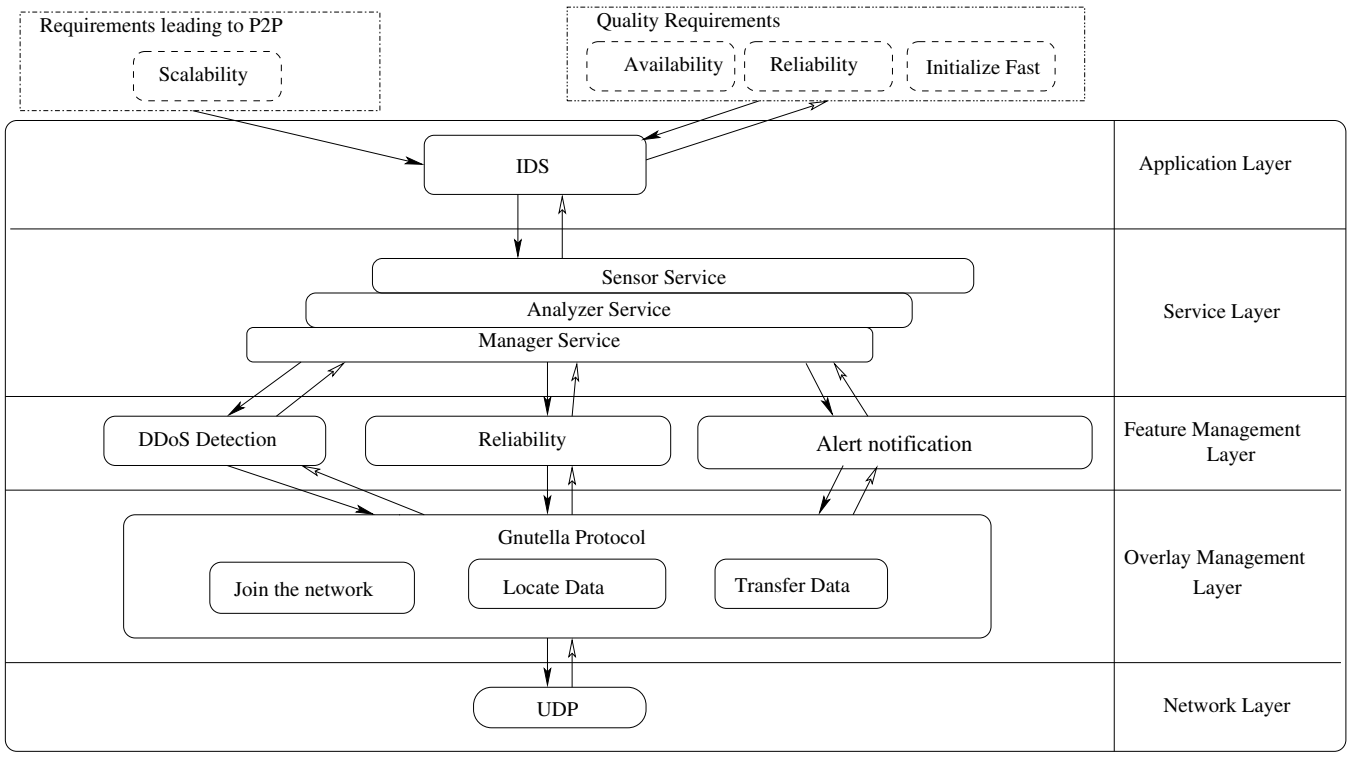

Figure 3. A P2P System Analysis Pattern Instantiation

Layer can guarantee the availability of the system.

Security goals The availability security goal has to be fulfilled at all times.

Constraints The availability of the system cannot be guaranteed due to the attributes of the P2P protocol and the network protocol.

The instantiation of the pattern and the feature management template for one service already shows that the availability requirements of the software cannot be fulfilled within the system architecture. A re-design of the architecture has to happen on the Feature Management Layer or even on the layers below in order to solve the problem.

VIII. Related Work

Tsalgatidou et al. [16] present a UML profile for P2P architectures. The profile builds upon the QoS UML profile and offers specific support for dependability properties, e.g., security, availability, and reliability. However, the work does not offer a structured approach for identifying the constraints a P2P-system can cause on requirements. On the other hand, the UML profile can complement our work.

Nakajima et al. [17] investigated the trustworthiness of P2P overlay networks. The authors investigate how faulty peers can be identified in a network and how likely it is to remove unwanted content from a P2P system. Their work can complement our own.

Grolimund and Müller [18] present a pattern language for overlay networks in P2P systems. The authors describe the usage of existing design patterns in P2P systems and what further design patterns are required in order to describe a P2P system. Verma [3] describes the general architecture of P2P systems in contrast with traditional client server architectures. The author describes basic components of every P2P system and compares the strengths and weaknesses of the two approaches. These works can also complement the work presented in this paper.

Sanders et al. [19] proposes organizational patterns for P2P applications and frameworks that address the problem that global knowledge and naming conventions do not exist in these applications. We are considering only software requirements without the business domain.

Angelaccio et al. [20] introduce a model-driven QoS management framework using UML. The authors also address adaptive QoS management in P2P systems. We are looking only into the augmentation of the software development processes with a method to investigate the constraints P2P systems have on requirements.

IX. Conclusions and Future Work

We showed a method that allows software engineers to identify constraints on requirements when designing a P2P-based software. Thereby the constraints on software requirements are made explicit and the software engineer gains

information on which layer in the P2P system the problem originates. This information can be used to extend the system with a component that lifts the constraint in the best case. At the very least software engineers can state that a specific requirement cannot be fulfilled when using a P2P-system.

Our approach offers the following main benefits:

- A structured, pattern-based method that can be instantiated for a given P2P-based-software system
- Systematic identification of constraints on requirements caused by P2P architectures
- Improving the outcome of distributed system implementation by supporting the identification of constraints on requirements, which are caused by P2P systems
- The approach can be used with a given software development process in order to elicit the ramifications of using a P2P-protocol

The work presented here will be extended to support further types of distributed systems. Moreover, we will look into refining the presented patterns into further technical patterns that are closer to specific implementations.

REFERENCES

[1] IETF, "Internet protocol," Internet Engineering Task Force (IETF), IETF RFC 791, 1981. [Online]. Available: http://tools.ietf.org/rfc/rfc791.txt

[2] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol. 7, pp. 72–93, 2005.

[3] D. Verma, *Legitimate Applications of Peer-to-Peer Networks*. Wiley-Interscience, 2004.

[4] L. L. Peterson and B. S. Davie, *Computer Networks - A Systems Approach*, 4th ed. Morgan Kaufmann, 2007.

[5] S. Agarwal, J. Singh, A. Mavlankar, P. Baccichet, and B. Girod, "Performance and quality-of-service analysis of a live p2p video multicast session on the internet," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, june 2008, pp. 11 –19.

[6] ITU-T, "Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors, Quality of telecommunication services: concepts, models, objectives and dependability planning – Terms and definitions related to the quality of telecommunication services," International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ITU-T E.800, 2008.

[7] S. A. Baset and H. Schulzrinne, "Reliability and relay selection in peer-to-peer communication systems," in *Principles, Systems and Applications of IP Telecommunications*, ser. IPTComm '10. New York, NY, USA: ACM, 2010, pp. 111–121.

[8] A. Fessi, H. Niedermayer, H. Kinkelin, and G. Carle, "A cooperative sip infrastructure for highly reliable telecommunication services," in *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*, ser. IPTComm '07. New York, NY, USA: ACM, 2007, pp. 29–38.

[9] J. Seedorf, "Security Issues for P2P-Based Voice- and Video-Streaming Applications," in *iNetSec 2009: Open Research Problems in Network Security*, ser. IFIP Advances in Information and Communication Technology. Springer, 2009, vol. 309, ch. 10, pp. 95–110.

[10] G. Gheorghe, R. Lo Cigno, and A. Montresor, "Security and Privacy Issues in P2P Streaming Systems: A Survey," *Peer-to-Peer Networking and Applications*, pp. 1–17, 2010.

[11] A. Tsalgatidou, G. Athanasopoulos, and P. Liaskovitis, "A uml profile for software architectures and peer to peer dependable applications," in *CAiSE Short Paper Proceedings'05*, 2005, pp. 9–14.

[12] IETF, "User datagram protocol," Internet Engineering Task Force (IETF), IETF RFC 768, 1980. [Online]. Available: http://tools.ietf.org/rfc/rfc768.txt

[13] ——, "Transmission control protocol," Internet Engineering Task Force (IETF), IETF RFC 793, 1981. [Online]. Available: http://tools.ietf.org/rfc/rfc793.txt

[14] ——, "The intrusion detection message exchange format (idmef)," Internet Engineering Task Force (IETF), IETF RFC 4765, 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4765.txt

[15] Gnutella, "The gnutella protocol specification," http://rfc-gnutella.sourceforge.net/developer/stable/index.html.

[16] A. Tsalgatidou, G. Athanasopoulos, and P. Liaskovitis, "A uml profile for software architectures and peer to peer dependable applications." in *CAiSE Short Paper Proceedings'05*, 2005, pp. 9–15.

[17] Y. Nakajima, A. G. Nemati, T. Enokido, and M. Takizawa, "Trustworthiness and confidence of peers in peer-to-peer (p2p) network," in *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, ser. AINAW '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 313–318.

[18] D. Grolimund and P. Müller, "A Pattern Language for Overlay Networks," ETH Zürich, Tech. Rep., Dec. 2005. [Online]. Available: http://www.inf.ethz.ch/research/disstechreps/theses/show?lang=en&serial=503

[19] R. T. Sanders and R. Braek, "Modeling peer-to-peer service goals in uml," in *Proceedings of the Software Engineering and Formal Methods, Second International Conference*, ser. SEFM '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 144–153.

[20] M. Angelaccio and A. D'Ambrogio, "A model-driven framework for managing the qos of collaborative p2p service-based applications," in *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 95–102.