Universidade do Minho
Escola de Engenharia

Ana Luísa Parreira Nunes

P2P content-push for the Internet

Dezembro de 2008

Universidade do Minho
Escola de Engenharia

Ana Luísa Parreira Nunes

P2P content-push for the Internet

Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor José Orlando Pereira

Dezembro de 2008

Acknowlegements

To my supervisor who didn't give up on me, and lent a helping hand until the very end.
And to Carlos, for his unwavering support.

Abstract

Syndicated content-push in the Internet is a huge success, and web feeds are being used to convey various types of content: from news headlines, to podcasts and video podcasts, to being a feature in Web 2.0 websites. This diversity lead to the appearance of several frameworks, each tailored to a specific content type. At the same time, interest in social networking exploded, as more and more websites of this purpose were launched. Syndicated content and social networking websites are now intimately connected.
In this work, a generic, modular, p2p content-push architecture is proposed. It provides an evolutionary upgrade path based on the technologies already in use in the Internet for interoperability, thus without disrupting current infrastructure or changing participants' habits. It also leverages social networks for content discovery and aggregation, using peer-to-peer protocols for distribution. A partial implementation of this architecture, dubbed SEEDS, is also presented.

Resumo

O conteúdo agregado na Internet é um grande sucesso, e as "feeds" de conteúdo web estão a ser usadas para transportar vários tipos de conteúdo: desde cabeçalhos de notícias, a "podcasts" e "video podcasts", até aparecendo como funcionalidade de sítios Web 2.0. Esta diversidade levou ao aparecimento de várias plataformas, sendo cada uma pensada para um tipo de conteúdo específico. Ao mesmo tempo, o interesse em redes sociais explodiu, dado que cada vez mais sítios Web com este propósito foram lançados. O conteúdo agregado e os sítios Web de redes sociais estão agora intimamente ligados.
Neste trabalho, uma arquitectura de "content-push" ponto-a-ponto, genérica e modular, é apresentada. Esta proporciona um caminho evolucionário de melhoramento, baseado em tecnologias já em uso na Internet para potenciar a interoperabilidade, sem causar rupturas na infraestrutura actual ou modificar os hábitos dos participantes. Também tira partido das redes sociais para descoberta de conteúdo e agregação, utilizando protocolos ponto-a-ponto para a distribuição. Uma implementação parcial desta arquitectura, denominada SEEDS, é também apresentada.

Contents

List of Figures

Chapter 1

Introduction

Some time ago, the bulk of information available in the Internet was only accessible in the content-pull paradigm. E-mail was pretty much the only widespread application of content-push. Since then, a shift towards making information available through content-push was made, using web syndication as the platform to support it. Social content sharing spawned from this shift in paradigm and from the growing interest in social networking services. In this work, a p2p content-push architecture, which leverages syndication protocols and the social web, is presented.

The rest of this chapter is structured as follows: in Section 1.1 the current syndication architecture and its shortcomings are presented, using news feeds as an example, hinting at the motivation for this work. Then, in Section 1.2 the problem statement for this work is given. Finally, in Section 1.3, an approach to solving the problem stated is given, highlighting its guidelines.
The rest of this document is organized as follows: in Chapter 2, some definitions are given, along with an analysis on related work in content dissemination and social networking. Then, in Chapter 3, the architecture of the proposed system is presented, along with the contributions it entails. In Chapter 4, the feasibility of the proposed system architecture is assesed, detailing a partial implementation. Finally, in Chapter 5

1.1 Background

Syndicated content in the Internet has been a huge success ever since its early days. Currently, it is the cornerstone of content-push, ranging from podcasts to emerging Web 2.0 sites. Unfortunately, the bafflingly simple technology that makes publication and subscription very simple and flexible, thus explaining

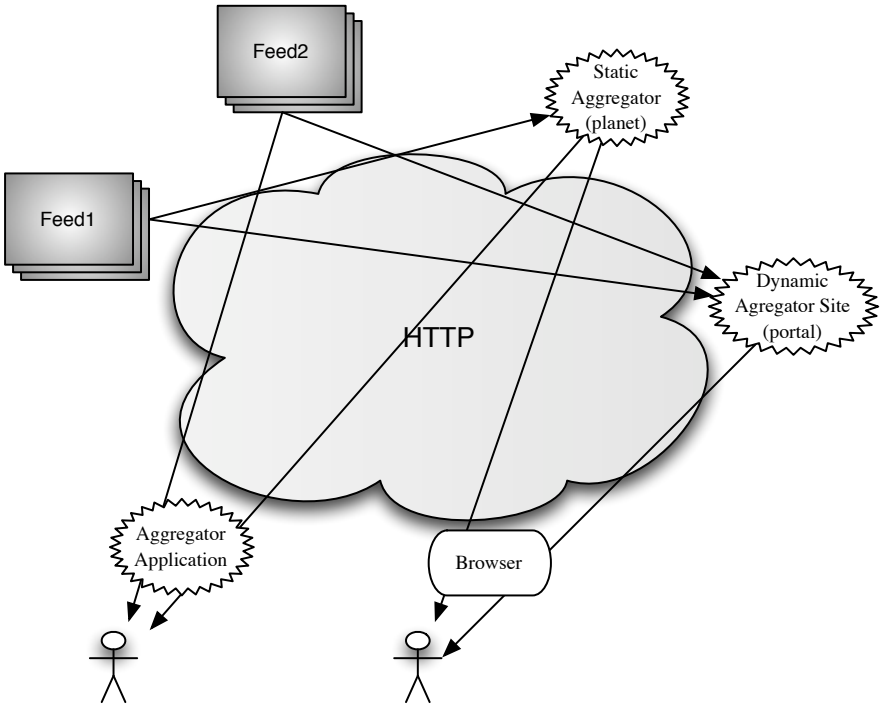

Figure 1.1: Internet news feeds distribution and caching architecture.

in part its success, is also limiting its usefulness in more demanding applications.

In order to illustrate the motivation for this work an example using news feeds is presented. Notice that feeds are an example of the migration to content-push in the Internet.

An example using news feeds

Fig. 1.1 depicts the common Internet feed distribution architecture. Feeds, published in a XML format (e.g. RSS [12]), can be subscribed to directly by the end-user, using an aggregator application (e.g. RSSOwl), by a portal (the original intent), or by a static aggregator (a.k.a. planet)[1]. Feed addresses are shared directly by users, through planets or portals. Notice that even though all aggreators can perform caching, only in portals and planets can cache be shared between multiple users. Notice also that planets are a manual effort to identify communities with shared interests and provide them with commonly accepted content. Portals are much less prone to allow that sharing. But, on the other

[1]See Section 2.1 for definitions.

hand, in planets the end user is not allowed to customize the feeds to be presented.

Even though from the user's perspective news feeds seem to behave like content-push, they are, in fact, a pull technology. The aggregator application simply hides the polling from the user, pushing new entries when those become available.

Consider, for instance, that one tries to stay current with the content of a given website. It would be necessary to keep polling each of the server's pages to check for updates. Now if we add that instead of one client, a website will typically serve several (and a popular website will serve thousands), and also take into account the mere size of the Internet, it soon becomes clear that bandwidth consumption and increased server load pose a problem, as the number of clients and their polling rates rise.

Syndication eases polling by collecting short descriptions of recent updates, with pointers, in a single file that can be polled more easily. Nonetheless, consider the following example. Each Wired Top Stories feed is sized at about 79KB, and from data available at Bloglines[2] it's subscribed by at least 94,951 users. If separately polling the feed with an estimated refresh period of 2 hours, with 12 updates per day, daily polling-related traffic reaches 87.9 GB.

Bandwidth consumption is aggravated by the inability of many servers and clients to retrieve only the updated entries, and that once a user subscribes to a feed, she'll likely maintain it for a long time. Also, stemming from the use of news readers and from the global nature of the Internet, polling will occur persistently.

Some techniques have been proposed to try to minimize this problem [27], although with limited success, as discussed in [37]. In [35] the author points out that "Conditional GET" [36] mechanisms may not necessarily lead to the bandwidth savings expected,

> " since most of these news operations churn out headlines with monotonous regularity. "

The RSS specification foresees the possibility of notifications being delivered to readers, through the rssCloud interface[14], whereby readers register to a cloud, and notifications are delivered by calling back, using either XML-RPC or SOAP, to a procedure registered by the reader. However, by design, this mechanism doesn't work for readers behind firewalls or NAT.

Another interesting point raised in [35] is that even the purpose of ISP-level caching may be defeated as

[2] `http://bloglines.com`, a popular Web based feed reader that provides usage statistics.

> "... if they do it with cached times of more than 10 min, then people will route around the caches."

This article also raises the concern of default loading of RSS feeds by browsers and possible implications for bandwidth consumption.

A look at syndication

The Web has grown tremendously and it's content has evolved to be much more dynamic than in its early days. While some websites still feature essentially static content, others are updated daily, and in some cases (e.g., news oriented websites) the update rate is even higher, amounting to several updates a day.

Syndication has changed the standard paradigm in order to cope with these events, by addressing the problem of knowing when an update is available. This is achieved using a content-push model, based on the publish/subscribe paradigm: content publishers provide a feed, that is, a XML file containing headlines and descriptions, often with links to new articles or blog entry posts, that users subscribe to. Conceptually, a web feed provides a summary of a website's recently added content.

Web feeds are also a major feature of social networking sites, being used to disseminate frequently updated information to interested users. For instance, Facebook [2] uses feeds to highlight changes in social circles and also to disseminate information about any particular user; FriendFeed [4] tries to make Web content more useful by leveraging social connections, as information about Web interaction, for users considered to be interesting, is condensed into a feed.

Current syndication protocols (RSS [12], Atom [1]) allow anyone to easily publish or subscribe to feeds by using simple tools. These protocols prove quite adequate for the large majority of feeds, featuring few subscribers and low data throughput.

Still, a new dimension arised when feeds began to be used to convey audio or video, originating podcasts and video podcasts, respectively.

There are some feed/content distribution architectures/frameworks available, but each bound to specific purposes: file-sharing, micronews delivery, microblogging, etc. Some frameworks even feature social awareness. Also, some proposals geared towards fully decentralized social networks, focusing both on privacy concerns and on data portability, have been presented.

1.2 Problem Statement

These different feed/content distribution architectures each look to a specific subset of p2p-based content push in the Internet. The social dimension, which by enhancing collaboration, could lead to benefits in terms of bandwidth savings, provide content recommendation and aid in establishing trust, is lacking in some and all together too restrictive in others. Also, available toolsets are tightly bound to the specific-purpose architectures. An important issue is the lack of an open and flexible framework to provide generic content-push, while leveraging existent general-purpose social networks to promote collaboration in managing the available resources. This open framework should not reinvent the wheel, building, instead, on composition of different already existent architectures, retaining the ability to cater to diffent needs.

1.3 Approach

The main goal of this work is to obtain a generic content-push architecture for the Internet, addressing the issues raised in Section 1.2. Feed entries are construed as content, and dissemination is done using p2p protocols. Group formation is central to this approach, and social information is key. To accomplish this purpose with the desired properties of openness and flexibility, a set of guidelines were considered:

Multiple Dissemination Protocols Content type and traffic patterns vary widely. For example, some feeds can be mainly composed of text, normally for website related news, or feature richer content (feeds with audio/video attached), generating traffic with different characteristics. File-sharing imposes yet another traffic pattern. Another factor is the number of publishers and subscribers envolved in each instance. It makes sense not to use the same dissemination protocol for such different cases, so it is necessary to differentiate them and choose the right protocol.

This way, we are able to disseminate small content quickly and large content efficiently.

No Change to the Source One of the reasons other feed dissemination proposals haven't met great success (e.g. FeedTree [44]) was that they relied heavily on the source to explicitely join the system. We should not expect the source to collaborate. Also, it should possible to use existent protocols and its architectures without fundamentally altering them.

No Change to the Target Clients shouldn't need to change their habits. Some content dissemination architectures require a specific client, and some use only specific-purpose social networks (eg. Tribler [40]). It should be possible to use pre-existent social networks (eg. Hi5 [3]) and take advantage of the different dissemination architectures without requiring a special client for each one, selecting automatically the most appropriate method.

Enhanced Discovery and Aggregation By leveraging social networks the clients' experience should be enhanced with recommendations and cooperation. These can be content related or stem from social connections.

Click but no Wait Large content, such as media files in podcasting and vodcasting, should be available to be viewed by the user with the same timeliness as the others, avoiding the "click-wait"[45] system.

[3] www.hi5.com

Chapter 2

Related Work

In this chapter, some related work is presented. First, in Section 2.1, definitions for some terms are given. Then, in Section 2.2, a number of proposals that focus either on news or multimedia caching and dissemination, and on content sharing in social networks, all relevant to this work, are presented and analyzed with respect to desired features.

The peer-to-peer content-push system proposed in this work handles diverse content types, where large content is automatically downloaded, published autonomously by a large number of sources, with large numbers of subscribers, which interface with the system using any feed-capable third-party application. Also, content discovery is provided through endorsement and recommendation. Technology standards are respected, including networking ones, such as to handle NAT traversal.

2.1 Some Definitions

In this section, some definitions deemed relevant to the scope of this work are given.

Publish/Subscribe or pub-sub is a message-based communication paradigm in which senders and receivers are decoupled. Receivers (subscribers) indicate their interest in specific channels, thereby **subscribing** to them. Senders (publishers), instead of sending messages directly to receivers, **publish** them to related channels. In an asynchronous manner, subscibers receive messages only from channels they expressed an interest in. The pub-sub paradigm comprises two fundamental processes for selecting messages for reception: topic-based in which named channels exist; and content-based in which selection is done based on subscriber-

defined attributes and/or content. A hybrid approach to selection is also possible. In [24], pub-sub is explained in further detail.

Content-pull vs Content-push Content-pull is the traditional communication pattern in which the receiver requests a data transfer from the sender. Content-push is a communication pattern implementing the publish/subscribe paradigm. In content-push services, the server delivers (**pushes**) content to the client, without being requested to do so by the latter. For example, the SMTP [1] protocol, used for email delivery, conforms to this pattern. For further details on these concepts, see [33].

Overlay Network Overlay networks, or simply overlays, are constructed by creating virtual or logical links between nodes of an underlying network.

Peer-to-peer Network A peer-to-peer (p2p) network is an overlay formed by edge nodes. Nodes cooperate to accomplish different tasks, be it file-sharing or information dissemination. In decentralized p2p networks, each peer simultaneously functions as a client and as a server,

Content Distribution Network (CDN) CDNs are overlays consisting of nodes that cooperate to deliver content to end users. Content distribution networks attempt to optimize content delivery, implementing web caches to store popular content closer to the user, thus reducing bandwidth requirements and server load, increasing reliability and also improving client response times. For example, the Coral Content Distribution Network [26] is an academic, free, peer-to-peer content distribution network that leverages the bandwidth of participating nodes as proxy servers, making it similar to a distributed web proxy.

Broadcatching This refers to automatic downloading of content published in feeds, eg. tv series related feeds.

Web Syndication, Feed, RSS, Atom Web feeds are files composed of entries, which may be headlines, full-text articles, excerpts, summaries, and/or links to content on a web site, along with various metadata. These have been traditionally used to provide users with frequently updated content. The term web syndication refers to how feeds are made available from a website, which is analog to how syndication works in broadcasting. RSS and Atom are XML-based document formats, both used for feeds. For futher details on these concepts and on how they relate to each other see [41] and/or [46].

[1] http://www.ietf.org/rfc/rfc2821.txt

Portal Portals are websites featuring content from a static set of chosen feeds.

Planet Planets are websites featuring aggregated content of interest to a particular community of users.

Aggregator Aggregators, also known as feed readers or news readers, are applications that fetch feeds, usually in an automated manner, and present entries to the user.

Social Networking This concept usually refers to using social network services, "focused on building online communities"[15].

Social Bookmarking This concept usually refers to sharing bookmarks with an online community.

Micro-blogging is "... a form of multimedia blogging that allows users to send brief text updates (say, 140 characters or fewer) or micromedia such as photos or audio clips and publish them, either to be viewed by anyone or by a restricted group which can be chosen by the user. These messages can be submitted by a variety of means, including text messaging, instant messaging, email, MP3 or the web."[7]

Swarming Swarming refers to collaborative file download, in which end users download bits of the file from other end users who already have them.

2.2 Related Work

2.2.1 Content dissemination

In this section, some related work in the areas of feed/content dissemnination, content recommendation, and social implications is discussed. An issue shared by several of the following architectures is the need to use of special-purpose tools and social networks.

YouTube

Youtube [2] is a very popular centralized video repository. Each time a user wishes to see video, it is streamed to the user. This sort of centralized architecture is perfect for when a single user is interested in a particular video. However, if this user wants to share this video with others, and since this is done by providing a link, each of the other users will need to stream the video from the centralized repository. Taking into consideration that each of these users

[2] http://youtube.com

can then share the video with others, or simply watch the video again, streaming from the centralized repository just seems wasteful in terms of bandwidth usage and of time waiting for the streaming itself.

Twitter

Twitter[16] is a popular micro-blogging service with an embedded social network and a large user base, distributed across the world[31]. Also from [31], one of the main user intentions is news reporting and

> "Some automated users or agents post updates like weather reports and new stories from RSS feeds."

Stemming from it's popularity and large user base, Twitter is currently suffering from scalability issues[17]. These issues are also discussed in [29] and in part III of this discussion, there are some suggestions on moving some of the load to the clients. In such a setting, polling in needed once again. A solution like the one proposed in this work could then improve bandwidth consumption, since Twitter's social network could be used to power the social engine for dissemination groups' management.

FriendFeed

FriendFeed[4], is a social networking service that condenses information on Web interaction, for interesting users, in a feed. Content consists of not only content published by friends (users subscribed to), but also of friends of friends. Other feeds are also published. For each user, a feed is created with self published content (eg. YouTube videos, Digg). Another feed features entries published by others that the user marked as "liked". Yet another feed carries the user's comments.
FriendFeed's architecture is fully centralized, leading to a necessary limit on the rate of new entries published by a user, to keep scalability-related performance issues in check. Once again, a website is consistently polled for updates. This proposal could be of help in minimizing polling, harnessing social network information for group management. In fact, FriendFeed already features user-managed groups, which just makes it easier. Also, the "like" mechanism could be construed as content endorsement, and used to generate recommendations.

Pownce

> "Pownce is a way to keep in touch with and share stuff with your friends. Send people files, links, events, and messages and

then have real conversations with the recipients."

The Pownce [3] service integrates social networking with content sharing. However, since it is implemented on a centralized architecture, there are some limits [10] to what users can do, both on the number of posts and on shared files' size.

Flock

Flock [4] is a web browser, but heavily interlaced with social networking and media services. Inspite of this integration, no bandwidth consumption optimization is done, nor improvements over traditional feed readers/aggregators:

> "Every hour Flock will check feeds for updates. You can refresh individual feeds by using the Reload button while viewing a feed."

Mermaid

Mermaid [5] is a suite of pure peer-to-peer desktop applications in which users create their own peer-to-peer networks. Among applications for video/audio broadcasting, a news exchange system [6] is presented, in which members of the peer-to-peer network are able to broadcast news. However, each content type requires a different application and the system is presented as an alternative to RSS.

Portals

Bloglines[7] is a portal, ie., a web-based news aggregator, where users can manage their feed subscriptions. Even though polling is reduced since, globally, each feed publisher is refreshed hourly, this imposes limits on the entries' freshness. Users wishing to monitor feeds at a higher rate cannot do so using this service.

Planets

Planet websites are used to aggregate syndicated web content for online communties and to display it on a single page. There is, without question a social component to this, albeit a static one, as viewers do not control which feeds are aggregated. Also, for different areas of interest, a client has to visit different planets which impose sharp boundaries on communities.

[3] http://pownce.com
[4] http://flock.com
[5] http://mermaid.metaaso.com
[6] http://mermaid.metaaso.com/mermaid/news/overview.html
[7] http://bloglines.com

FeedBurner

FeedBurner [3], provides several services to feed publishers. To do so, it works like a centralized cache, in that the "burned" version of the feed is the one publicized. The original feed is checked for updates every 30 minutes. Although server load is moved away from the publisher's server to FeedBurner's servers, the bandwidth consumption problem remains the same. Also, the published "burned" feed's timeliness is limited.

FeedTree

FeedTree[44], is an approach directed at collaborative micronews delivery. It uses a single communication protocol (Scribe [20]) built on top of a peer-to-peer overlay network (Pastry [43]) and does not promote discovery and aggregation based on interests or social connections.

FeedEx

FeedEx[32] is a news feed exchange system, where peers cooperate by exchanging feed documents with their neighbours. In FeedEx, content recommendation is provided, stemming from two sources: from entries rating (or votes) or by viewing a read as an implicit endorsement; from the feed subscription sets, determining similar interests. Neighbour selection is done at each peer, based on the overlap in subscription sets, considering also other factors like topological proximity.
FeedEx is limited to news feed dissemination, and so, not suited for disseminating multimedia content (large files). It does provide interesting features, such as the construction of a distributed news archive and an incentive mechanism to minimize selfish behaviour. Also, in this proposal, a specifically tailored peer-to-peer protocol is used, and it is heavily intertwined with the application itself. Thus, FeedEx does not support different dissemination protocols.

Tribler

Tribler [40] is an open source BitTorrent client that leverages a social network to provide content recommendations and cooperative downloading, using a mechanism of bandwidth donation. Users can create profiles, groups, add friends and tag content. Tribler's decentralized recommendation process uses an algorithm based on an epidemic protocol, by exchanging download histories with peers known to have similar tastes but also with peers chosen at random. Tribler is geared towards file-sharing, particularly video content and requires custom desktop tools to participate.

Last.fm

Last.fm [5] is a service that can be used for listening to music and that provides content recommendation based on user's profiles. When a user listens to a music, this information is added to a profile page, visible to other users. With this data, Last.fm suggests music to users, also allowing them to create personalized radio stations. In this social network, the notion of friendship exists along side the notion of neighbourhood, taken from the similarity in musical taste. This service is designed to work with musical content only and is restricted to centrally supplied content.

Content Distribution Networks

Content distribution networks attempt to optimize content delivery, implementing web caches to store popular content closer to the user, thus reducing bandwidth requirements and server load, increasing reliability and also improving client response times.
The Coral Content Distribution Network [26] is an academic, free, peer-to-peer content distribution network that leverages the bandwidth of participating nodes as proxy servers, making it similar to a distributed web proxy. However, network nodes are not users and there is no social component associated. Corona is a distributed, peer-to-peer, cooperative resource management framework. It's purpose is to maximize the effective benefit of the aggregate bandwidth of the system, while remaining within server-imposed bandwidth limits. With Corona, any object identifiable by an URL can be monitored for changes. Like Coral, network nodes are either a part of the same adminstrative domain, or can consist of server-class nodes, owned by participating institutions. This framework focuses on balancing the tradeoff between update performance (freshness) and network load.

Flash 10, Stratus, RTMFP

Adobe [8] has just released a "hosted rendezvous service" to enable direct end user to end user communication. This service is based on the Real-Time Media Flow Protocol (RTMFP) [9], a peer-to-peer protocol. However, peers must keep connected to the rendezvous host, even though application data is exchanged directly by them. The end clients can be either Flash Players or Adobe AIR endpoints. This service is directed at real-time communications, such as social networking and multi-user game. File or document-sharing is not supported, nor is swarming.

[8] http://www.adobe.com/
[9] http://www.adobe.com/go/rtmfp_faq

Although this platform may come to be the one with the most peers (one needs only to think about Flash browser plugins), as it is now, the service feels too restrictive in its usefulness.

Azureus/Vuze

Vuze [10] is a GUI [11] for the popular Azureus BitTorrent client, which now features an embedded social network based on the notion of friendship, for collaborative downloading. This friendship is considered to be, mostly, derivative from pre-existent social connections. Some mechanisms are available for friends to cooperate: *friend boost* if one of a user's friends wants content already owned by the user, the latter is able to donate a portion of available bandwidth to download it; torrents can be shared with friends, which can be construed as a form of recommendation.

Miro

Miro [12] is an open source desktop video player and broacatching application that integrates a BitTorrent client to automatically download videos from RSS-based channels.

2.2.2 Social networking

One of contributions of this work is the leveraging of social networking information to perform content recommendation and group management. Several proposals[19, 23] have been presented regarding social network extraction from several types of data, like email exchange, instant messaging, phone calls, publications, web pages, as well as infomation extraction from social networks. An interesting contribution is made in [42], which proposes a framework for using social network connections to establish out-of-band communication channels. The recent proposals for universal identification services like OpenID [13] may be a stepping stone for enabling a more integrated view of social networks. Also, many of the social networking services mentioned in this chapter already expose APIs that can be used to extract useful information. Some even conform to a common specification dubbed "OpenSocial" [14]. These sort of initiatives enable easy access to social networking information.

[10] http://www.vuze.com/app
[11] Graphical User Interface
[12] http://www.getmiro.com
[13] http://openid.net
[14] http://code.google.com/apis/opensocial/

Chapter 3

Proposed Architecture

3.1 Overview

Figure 3.1 shows the proposed architecture for a generic peer-to-peer content-push system in the Internet, providing the features presented.
The peer-to-peer content-push system is composed of the following entities:

Publishers make web feeds available in the Internet, thereby supplying content to the system;

Social Networks exist throughout the Internet, be it specific in purpose or, recently, in dedicated websites;

HTTP is the communication protocol of choice for the World Wide Web, serving as a base for web feeds;

Multicast protocols are used to efficiently deliver compact files to a number of different parties, creating peer-to-peer overlay networks over the Internet;

File Sharing protocols are used to efficiently deliver large files to a number of different interested parties, creating peer-to-peer overlay networks over the Internet.

3rd Party Applications provide the system's users with an interface;

System Proxys provide access to the system, and cooperate towards the established goals;

Social I/O Managers are the gateways to social networking services. The harnessed information is used to establish system-wide manangement poli-

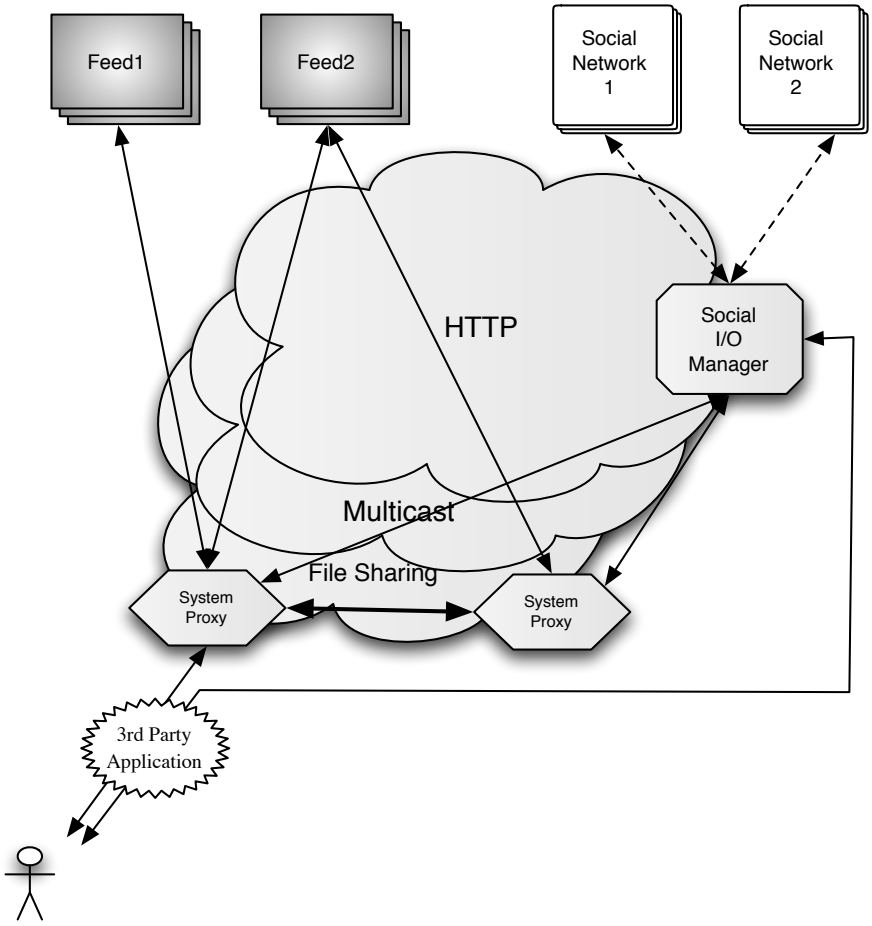

Figure 3.1: Overall Architecture

cies. These can be deployed separately from the rest of the system, as the system can function without them.

A brief summary of how these components interact, starting from a given Proxy, follows: the Proxy identifies a feed request from a third-party application and either returns the correponding cached feed, or requests it from the Publisher. If it is a previously unknown feed, the Proxy contacts a Social I/O Manager to obtain instructions on how to handle its caching and dissemination; if none is available, the Proxy proceeds with its own set of configurations. The Proxy deploys client instances of a multicast protocol for entry dissemination, and if deemed necessary, of a file sharing protocol. Updates to the cached feeds arrive from the protocol instances.
Also, the Proxy gathers user feedback, transmits it to a Social I/O Manager and

also uses it to determine endorsed content. The Proxy also provides a locally generated recommendation feed.
The Social I/O manager posts new content to Social Networking services, and retrieves information on social connections to create groups.

The following subsections summarize the main characteristics of the generic architecture being proposed.

3.1.1 User Interface

The system is offered as a feed service performance booster and a proxy server is deployed locally (client-side) to interpret user requests. Because this service is offered in this manner, any feed-capable third-party applications, like web browsers or feed readers, can be use as an interface to the system. Configuration is as simple as providing the proxy's address to the application.
Since the user interface is exactly the same whether this system is used or not, and little configuration is required, the system is designed to work with large numbers of feed subscribers.

3.1.2 Publisher Interface

In order to be compatible with the proposed system, publishers just need to provide a web feed through an HTTP service. Since this requires no change on the publishers part, the system is set up to handle large numbers of subscribers.

3.1.3 Feedback and Recommendations

User feedback is collected on each entry viewed by the user. Recommendations are offered in two complementary ways: recommended entries are presented to the user embedded in other content and recommendation feeds, are made available for user subscription, by the proxy.

3.1.4 Social Interaction

Some input is taken from social networking services regarding, in particular, social connections for optimizing peer organization. Also, if desired, the services' content regarding user activity in the system is updated. This makes the system two-way compatible with existing social networking services.

Figure 3.2 depicts both the information flow and interaction pertaining to the social features of the system.

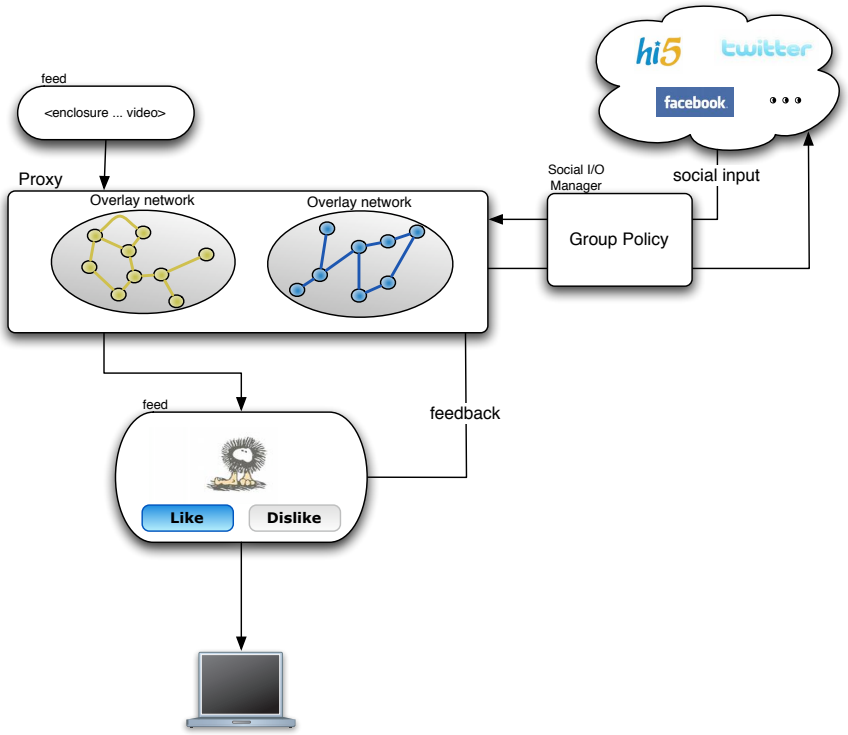

Figure 3.2: The Social Information Flow

3.1.5 Standards

To further interoperability, the system must adhere to standards. For example, NAT [1] or firewalls should not prevent the system from being used.

3.1.6 Automatic Content Cooperative Downloading

Large content is automatically downloaded and made locally available to the user. In order to optimize bandwidth consumption and the timeliness of content delivery, peers cooperate to download these files. The communication paradigms used to accomplish this goal are a part of the Internet.

[1] http://en.wikipedia.org/wiki/NAT_traversal

3.2 Protocols

3.2.1 Multicast

The multicast protocols used to disseminate feed entries must meet the following requirements:

- to be reliable;
- to scale to large numbers of peers;
- to scale to large numbers of entries;
- to handle many different groups (overlays);
- to expose a convenient API for this purpose;

3.2.2 File Sharing

The file sharing protocols used to disseminate large content should meet the following requirements:

- to be reliable;
- to optimize bandwidth consumption by taking advantage of cooperative downloading;
- to feature small latency from when a file is first downloaded to when it is made available for others;
- to expose a convenient API for this purpose;

his way, in most cases, when large feed entry content is to be displayed in other peers, the file will already be available locally, thus increasing the timeliness of delivery, while also optimizing bandwidth usage.
This sort of pre-fetching and caching is already implemented for photofeeds, that is, feeds in which entries feature image enclosures[9]. By integrating Snark with SEEDS in this manner, it is easy to deal with feeds featuring items with diverse content types.

Chapter 4

Evaluation

In order to show that the proposed architecture is, in fact, feasible, a prototype was built and named SEEDS.

4.1 Overview

Figure 4.1 depicts overall architecture of SEEDS. If compared to Figure 3.1, it is clear that this figure depicts an instance of the previous one, now naming the protocols to be used, and identifying the proxy's implementation.

When enabled, the proxy tries to identify feed URLs. Once a new feed is detected, the proxy contacts the Social I/O Manager for a *seed*. A *seed* is simply an XML configuration file containing both caching and dissemination protocol directives. Within these directives comes the identification of the dissemination group this feed belongs too. From this point on, each new entry of this feed received is to be broadcasted to the group, with the purpose of updating other peers. Naturally, the proxy is now listening for updates for this feed from the group. If a *seed* isn't available for any reason, the proxy falls back on local configuration. Two buttons are added to each entry presented to the user, to obtain feedback. This feedback is used to update the user's information on social networking services and it proves quite useful for recommendation purposes. Recommendations are offered in two complementary ways: in a specific feed generated at the proxy with entries of feeds that are not yet being monitored by it; by piggybacking endorsed entries on regular ones. This process is detailed in Section4.5.1.
Simply put, the social I/O manager handles the interaction with social networking services, and defines group policies. This service is presented separately, so it can be deployed independently.

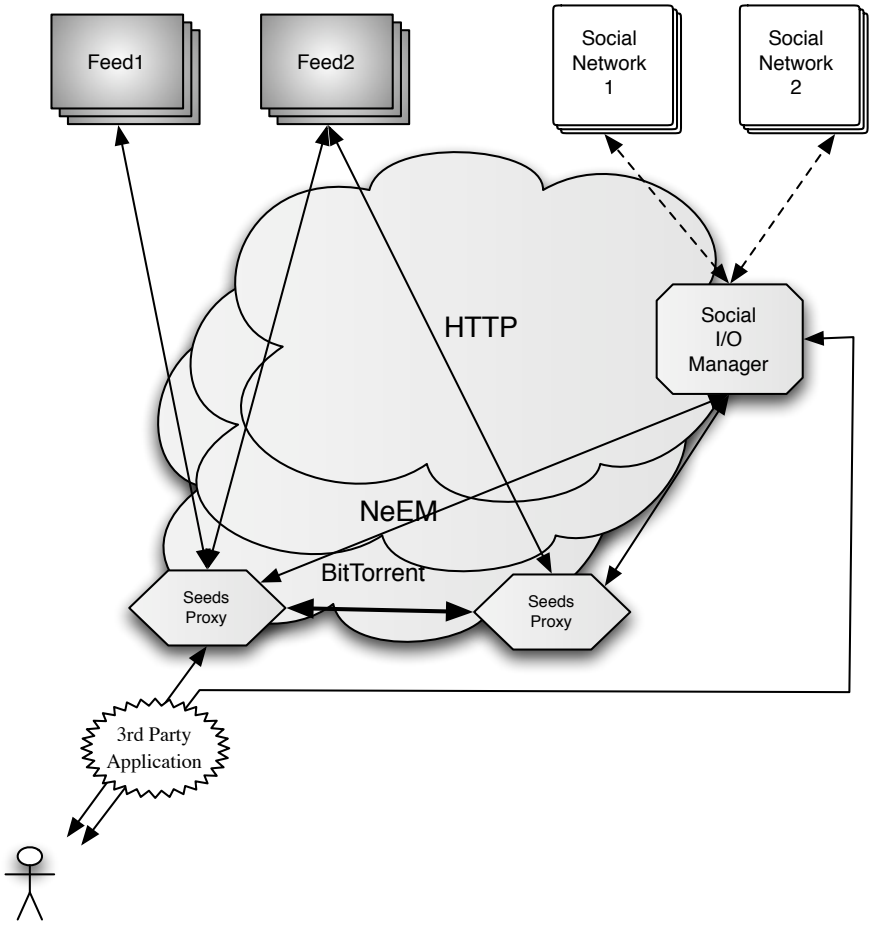

Figure 4.1: Overall Architecture

4.2 The SEEDS Proxy

The proxy's internal architecture is shown in Figure 4.2. The proxy is composed of three main modules: the HTTP Servlet which interprets the requests received by third-party applications; the Cache Manager which handles feed caching and dissemination to the groups; and an HTTP Client, responsible for fetching feeds from the publishers and interacting with the Social I/O Manager. Here is a step-by-step example of how this works:

1. a third-party application makes a feed request;

2. this request is interpreted by the servlet to check whether it is in fact a feed URL, and if so passes it on to the cache manager;

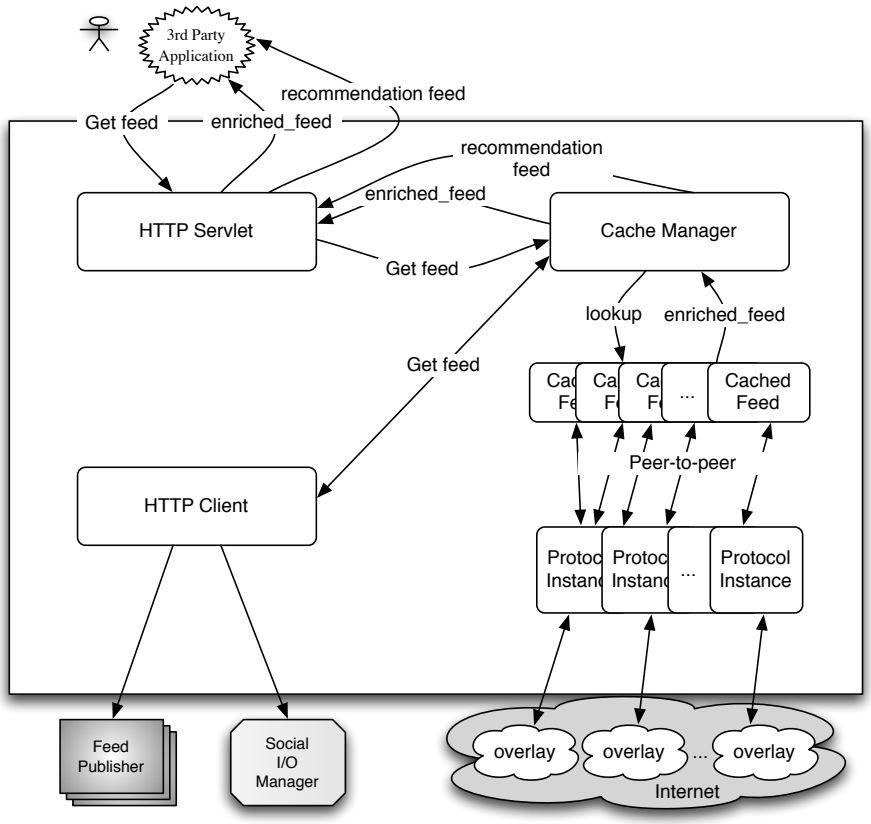

Figure 4.2: SEEDS proxy's internal architecture

3. the cache manager does a lookup for the URL in the cache. If the URL is being cached, it returns the cached feed entries, enriched with the feedback "Like" and "Dislike" buttons for the user to provide feedback, if desired, and also with some recommended entries; otherwise it retrieves the feed from the publisher and also a seed from the social I/O manager. Then, the cache manager deploys the appropriate protocol module, configuring it with a peer-to-peer group, according to the information received in the seed, and to each received entry [1];

4. In the background, the cache manager is both sending to, and listening for, updates from the peer-to-peer groups.

4.2.1 Instantiation

A Feed is assumed to conform to either the RSS or Atom standards.

[1] See Subsection 4.2.2

The 3rd Party Application can be any web feed-capable application, like a browser, or a feed reader;

The HTTP Servlet used in SEEDS is the lightweight Jetty WebServer [2];

The HTTP Client used to retrieve feeds is a part of ROME [3] a Java-based set of tools designed to handle both RSS and Atom web feeds;

The Cache Manager is implemented in Java.

4.2.2 Protocols

In Section 1.3, one of the guidelines presented referred to using multiple dissemination protocols to deal both with the different content types used in feeds, with the possibly different numbers of interested users in each set of feeds and also with diverse feed update rates.
One of the features of SEEDS is the automatic inline replacement of links to media files with the full content. This notion is similar to "broadcatching" enabling SEEDS to act as a client/server in this type of service. The RSS specification [11] identifies, in items, links to media (or simply large) files with the *enclosure* tag, providing the following information:

- the URL
- file size (in bytes);
- MIME type [4]

Similar information can be derived from the link element of an Atom feed. Using this information, SEEDS uses the BitTorrent protocol to disseminate these large files in the groups.
Notice that for feeds featuring a single subscriber, a centralized architecture is best, yet SEEDS can still be used seamlessly, keeping in mind it makes sense not to burden the network with disseminating these updates. Even in this scenario, users benefit from other features of SEEDS, as recommendations are still provided and feedback is gathered and processed.

Multicast

Epidemic multicast[39, 25], also dubbed probabilistic or gossip-based, is based on the simple procedure of relaying each received item to a small random subset of other nodes. This mechanism ensures these procotols are well suited

[2] http://www.mortbay.org/
[3] https://rome.dev.java.net/
[4] http://www.iana.org/assignments/media-types/

to disseminate events to a large number of interested parties, while achieving stable high throughput, and coping with node or network faults. It has been shown that the probability that all nodes are informed can be made as large as desired, without reaching 1, and that messages spread exponentially fast. NeEM[38] is an implementation of epidemic multicast in wide-area networks relying on connection-oriented transport connections and on specific mechanisms to avoid network congestion. The resulting overlay network is automatically managed by the protocol.
Its scalability and reliabilty make this protocol ideal for disseminating feed entries in groups where most peers are interested in the same set of feeds, and with high update rates.
The NeEM protocol exposes the following API [5] :

Class MulticastChannel is instatiated to create a new group;

connect(__peer_address__) is used to add another peer to a group;

read(_msg_) is used to receive a message, removing it from the incoming queue;

write(_msg_) is used to send a message to the members of the group;

For this implementation, the NeEM protocol is used, but there are other protocols, such as Scribe[20] which could also be used, because of SEEDS' modularity.

File Sharing

BitTorrent is a peer-to-peer protocol designed for file sharing, that fosters user cooperation. Each peer is able to request or transmit any computer file over the network. Here is a brief explanation on how this protocol works:

- A peer creates a .torrent file, containing metadata about the files to be shared and about a "tracker", a host to coordinate downloading.
- A peer wanting to download a file must first obtain the corresponding torrent file, and connect to the tracker therein mentioned.
- The tracker will tell the downloading peer which peers to contact to obtain pieces of the file.
- The downloaded file pieces are immediately made available for download by other peers.

[5] Only the most relevant portion of the API, for this purpose, is exposed here

There are several BitTorrent clients available for free download, implementing slight variations of the protocol implemented in the official BitTorrent client [6].
The BitTorrent client chosen to be used in SEEDS is based on Snark [7], a trackerless [8], lightweight implementation of the protocol, featuring multi-torrent support, written in Java. This client needed some small changes to serve its purpose with SEEDS.
Here is the procedure followed by a peer that receives a new entry with an enclosure:

1. this peer proceeds to download the content from the external link supplied in the enclosure;

2. both a torrent file and a tracker are instantiated by this peer's Snark client, and made available in a lightweight webserver;

3. when delivering the feed to the 3rd party application, the remote link is replaced with a link to the already downloaded file;

4. when disseminating the entry to the group, together with the other entry types, using, e.g. NeEM, the enclosure link is changed to the corresponding torrent file, hosted in this peer.

This way, in most cases, when large feed entry content is to be displayed in other peers, the file will already be available locally, thus increasing the timeliness of delivery, while also optimizing bandwidth usage.
This sort of pre-fetching and caching is already implemented for photofeeds, that is, feeds in which entries feature image enclosures[9]. By integrating Snark with SEEDS in this manner, it is easy to deal with feeds featuring items with diverse content types.

4.3 Standards

SEEDS is fully compliant with the standards associated with the technologies in use.
The NAT traversal problem has been addressed with UPnP [9], using the UPNPLib [10] implementation from SBBI.

[6] http://www.bittorrent.com/
[7] http://klomp.org/snark/
[8] Each client is also a tracker.
[9] http://www.upnp.org/
[10] http://www.sbbi.net/site/upnp/

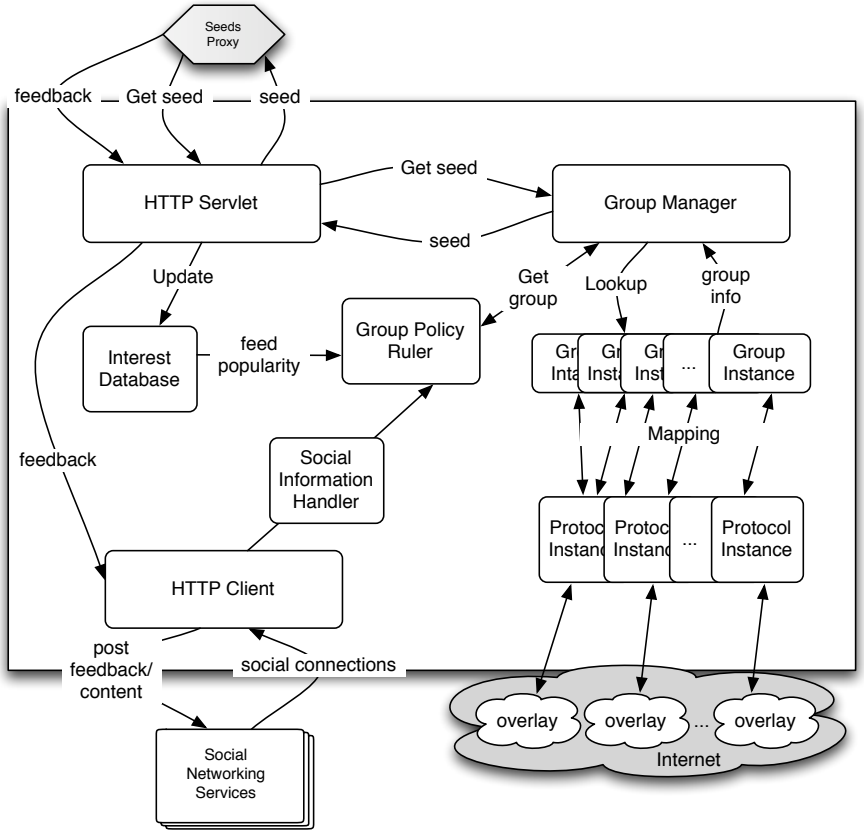

Figure 4.3: SEEDS Conceptual Social I/O Manager's Internal Architecture

4.4 Social I/O Manager internal architecture

Figure 4.3 depicts the conceptual internal architecture for the Social I/O Manager module.

4.4.1 Instantiation

The HTTP Servlet instantiated with the lightweight Jetty Webserver;

The Other Components can be implemented in a programming language of choice, as long as it provides support for the interactions needed. For example, the Social Information Handler may need to use the OpenSocial API, so, a language featuring a library for this purpose should be chosen.

Social networking information is to be used as an input to the Group Management module of the Social I/O Manager. Also, user feedback is to be propa-

gated back to these social networking websites. The social networking services that this module could interact with could be three different types:

- General purpose standalone services, like Facebook, using the OpenSocial API
- Social bookmarking standalone services, like Digg [11]
- Content-oriented services, like FriendFeed, which expose useful APIs

Actually, the FriendFeed service provides the concept of a room, a web feed shared by the room's "ocupants". This could enable a first stage of social I/O.

4.4.2 Group Management Policy

Each dissemination group, and therefore, each overlay network, is associated with a protocol instance.
Notice that BitTorrent groups are dynamically created by the proxys, and managed directly, without interaction with the social module.
Groups using the NeEM protocol are managed by the Group Policy Ruler module of the Social I/O Manager, where policy definition will take into account the following input:

- the number of peers in running groups;
- the interest in specific feeds (number of subscribers);
- information about social connections between peers;

Regarding the first item, when a seed request is received, the interest database gives a measure of peers' interest in a particular feed. For example, when only a single subscriber was previously known, this feed may now be mapped to an existing group. Also, for highly popular feeds, groups can grow to be so large that a specially dedicated new group can be created.
Now considering the second item, in some social networking services like FriendFeed or Twitter [12], social connections are either based on similar interests/tastes, or "real"/external connections are used to determine who receives published content. In an initial phase, the FriendFeed service could be used as a gateway to other social networking websites, since it harvests those other social networks to address content. Also, it provides a simple but powerful API [13] to take advantage of its service. Notice that while to take full advantage of the

[11] http://digg.com
[12] See Subsection 2.2
[13] http://code.google.com/p/friendfeed-api/wiki/ApiDocumentation

social features of SEEDS it is necessary for the user to configure her social networking accounts, this is still an improvement over simply using FriendFeed or switchAbit [14], which provides publishing aggregation, separately. Also, if this is not the case, the SEEDS architecture remains fully functional.

Some work has already been done in identifying possible groups from server logs. In Figure 4.4, statistics collected from an HTTP proxy are displayed: the circles represent user IPs, while the diamonds represent the twenty most popular web feed URLs. Some clusters are identified at a glance, which could easily be used to create groups:

- A group could be created with the peers interested in the most popular feed (1);
- Another group could be formed with the peers interested in bottom cluster of feeds;
- Yet another group could formed with the peers interested in the third most popular feed (3);
- And another group formed with the peers interested in the remaining feeds.

4.5 Recommendation

4.5.1 Feedback and Endorsed Entry Recommendation

To each feed entry, two HTML buttons are added. These are labeled with "Like" and "Dislike", an simply embedded in the entry's description. By clicking the "Like" button, therefore signalling the proxy, the user is actively endorsing the content. This entry is then sent to groups, marked as an endorsed recommendation, to be piggybacked on other entries. Disliked entries will just be ignored for this purpose. At an initial stage, FriendFeed could be used to publish these liked entries, depending on whether a peer is also a FriendFeed user.

4.5.2 Feed Recommendation

Another source of recommendations is group traffic traversing each SEEDS proxy. Each proxy can then publish it's own recommendation feed, where entries are derived from the recommended feed's channel information: the feeds

[14] http://beta.switchabit.com/

circulating in the group but not subscribed by the proxy's user are selected. Users choose to receive this type of recommendations by subscribing to the locally published feed.

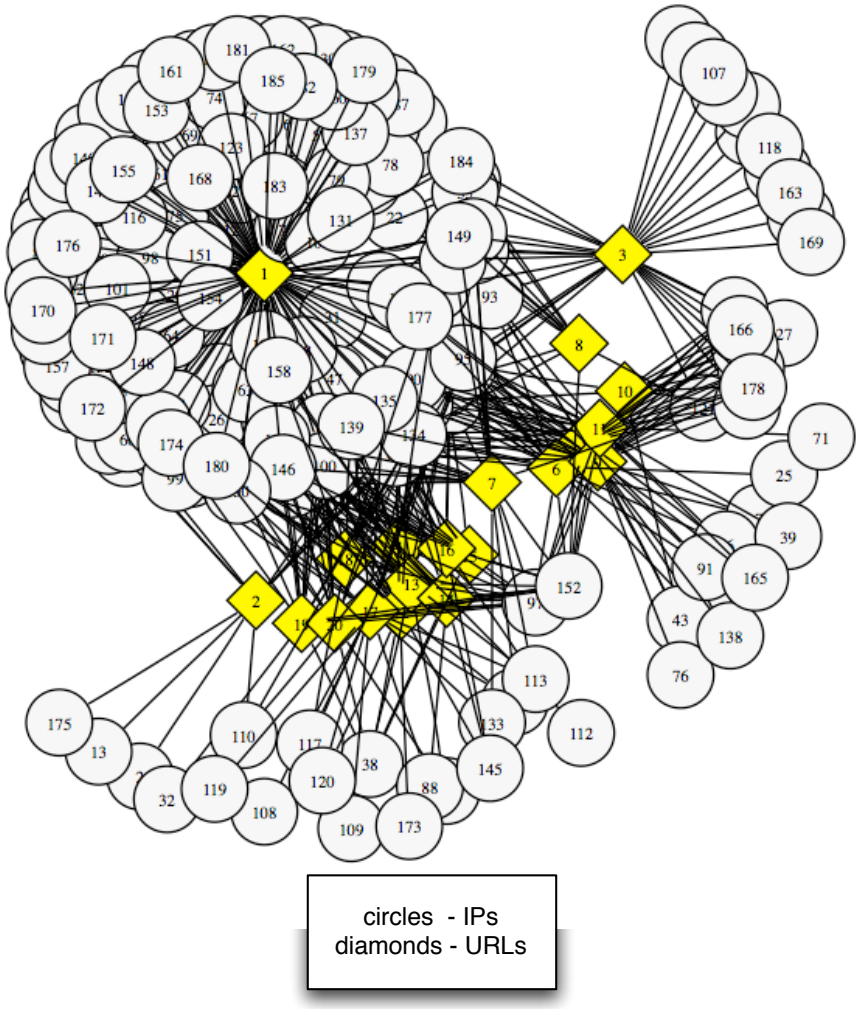

Figure 4.4: Some Statistics

Chapter 5

Conclusions

In this chapter, some conclusions are presented and some future work is discussed.

In this work, several content dissemination platforms, both centralized ones and peer-to-peer ones, were analyzed, regarding flexibility, scalabilty, content discovery, social-awareness, user interface, publisher interface and conformance to standards. None were found to address all of these topics. While some, targeted towards file sharing, require users to use specific desktop tools (like Tribler, Vuze, ...), thus ”leaving” the Internet environment, others, directed at content sharing in a social environment (Twitter,...), are opaque to the user.
The architecture proposed in this work, shows this need not be the case. It's independence of the actual technologies used to implement it, mean it is as flexible as desired. Since it can be implemented using actual technologies in use in the Internet, compatibility issues are avoided. The guidelines “Multiple Dissemination Patterns”, “No Change to the Source”, “No Change to the Target”, “Enhanced Discovery and Aggregation” and “Click but no Wait”, presented in 1.3, summarize the main features of the proposed system and also its main contributions.
The partial implementation of this system (SEEDS) shows that this system is feasible, and hints at how the complete implementation can be achieved. In its hole, the system proposed in this work draws from the composition of smaller systems, but its value is greater than that of the sum of its parts.

5.1 Future Work

A possible improvement would be to embed video/audio content directly in the feeds to be displayed. A possible research direction could be to take ad-

vantage of the Media RSS[6] module defined by Yahoo [1], to make use of a web browser media player console for this purpose.

Another improvement could be to use the SEEDS proxy's web server to act as a tracker for BitTorrent instances.

Initially, the FriendFeed service could be used as a gateway to other social networking websites. However, to further the usefulness of the integration those websites, the liked entries [2] should be posted to them websites. Services like switchAbit which routes content published in one social networking service to others, will probably be the way to go to accomplish this goal.

Finally, in order to gather meaningful statistics on the behaviour and performance of the proposed system, implemented in SEEDS, a large-scale deployment in a platform like PlanetLab [3], could prove instrumental.

[1] http://yahoo.com
[2] See Subsection 4.5.1
[3] http://www.planet-lab.org/

Bibliography

[1] Atom protocol (ietf draft). http://bitworking.org/projects/atom/draft-ietf-atompub-protocol-04.html.

[2] Facebook. http://www.facebook.com.

[3] Feedburner. http://www.feedburner.com/fb/a/home.

[4] Friendfeed. http://friendfeed.com/.

[5] Last.fm. http://www.last.fm/.

[6] Media rss module - rss 2.0 module. http://search.yahoo.com/mrss.

[7] Micro-blogging (wikipedia entry). http://en.wikipedia.org/wiki/Micro-blogging.

[8] Miro. http://www.getmiro.com/.

[9] Photofeed (wikipedia entry). http://en.wikipedia.org/wiki/Photofeed.

[10] Pownce community wiki / api documentation2-0. http://www.hueniverse.com/hueniverse/2008/04/scaling-a-micro.html.

[11] Rss 2.0 specification. http://cyber.law.harvard.edu/rss/rss.html.

[12] Rss protocol (wikipedia entry). http://en.wikipedia.org/wiki/RSS.

[13] Seeds: server-friendly epidemic feeds. http://seeds.sourceforge.net.

[14] Soap meets rss. http://cyber.law.harvard.edu/rss/soapMeetsRss.html#rsscloudInterface.

[15] Social network service. http://en.wikipedia.org/wiki/Social_network_service.

[16] Twitter: What are you doing? http://twitter.com/.

[17] Twittering about architecture. http://dev.twitter.com/2008/05/twittering-about-architecture.html.

[18] *22nd Symposium on Reliable Distributed Systems (SRDS 2003), 6-8 October 2003, Florence, Italy*. IEEE Computer Society, 2003.

[19] L.A. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, 2003.

[20] M. Castro, P. Druschel, A.-M. Kermarrec, and A.I.T. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *Selected Areas in Communications, IEEE Journal on*, 20(8):1489–1499, Oct 2002.

[21] Bram Cohen. Incentives build robustness in bittorrent, May 2003.

[22] Bram Cohen. The bittorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html, January 2008.

[23] A. Culotta, A. McCallum, and R. Bekkerman. Extracting Social Networks and Contact Information From Email and the Web, 2005.

[24] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[25] P.T. Eugster, R. Guerraoui, A.M. Kermarrec, L. Massoulie, and AJ Ganesh. From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, 2004.

[26] M. Freedman, E. Freudenthal, and D. Mazi. Democratizing content publication with coral, 2004.

[27] Peter Freitag. 8 ways to save bandwidth on your rss feed. http://www.petefreitag.com/item/642.cfm.

[28] Steve Gilmor. Bittorrent and rss create disruptive revolution. http://www.eweek.com/c/a/Messaging-and-Collaboration/BitTorrent-and-RSS-Create-Disruptive-Revolution/1/.

[29] Eran Hammer-Lahav. Scaling a microblogging service. http://www.hueniverse.com/hueniverse/2008/04/scaling-a-micro.html.

[30] Matthew Hicks. Rss comes with bandwidth price tag. http://www.eweek.com/c/a/Messaging-and-Collaboration/RSS-Comes-with-Bandwi dth-Price-Tag/.

[31] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, New York, NY, USA, 2007. ACM.

[32] Seung Jun and Mustaque Ahamad. Feedex: collaborative exchange of news feeds. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 113–122, New York, NY, USA, 2006. ACM.

[33] Julie E. Kendall and Kenneth E. Kendall. Information delivery systems: an exploration of web pull and push technologies. *Commun. AIS*, page 1.

[34] H. Liu, V. Ramasubramanian, and E.G. Sirer. Client Behavior and Feed Characteristics of RSS, a Publish-Subscribe System for Web Micronews. In *Proc. of ACM Internet Measurement Conference*, 2005.

[35] Om Malik. Rss, tiger safari and the bandwidth bottleneck. http://gigaom.com/2005/04/28/rss-tiger-safari-and-the-bandwidth-bottleneck/.

[36] Charles Miller. Http conditional get for rss hackers. http://fishbowl.pastiche.org/2002/10/21/http_conditional_get_for_rss_hackers/.

[37] Randy Charles Morin. Howto rss feed state. http://www.kbcafe.com/rss/rssfeedstate.html.

[38] J. Pereira, L. Rodrigues, M. J. Monteiro, R. Oliveira, and A.-M. Kermarrec. NEEM: Network-friendly epidemic multicast. In *SRDS* [18], pages 15–24.

[39] J. Pereira, L. Rodrigues, M. J. Monteiro, R. Oliveira, and A.-M. Kermarrec. Neem: Network-friendly epidemic multicast. *Reliable Distributed Systems, IEEE Symposium on*, 0:15, 2003.

[40] J.A. Pouwelse, P. Garbacki, J. Wangand A. Bakker, J. Yang, A. Iosup, D. Epema, M.Reinders, M.R. van Steen, and H.J. Sips. Tribler: A social-based based peer to peer system. In *5th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2006.

[41] S. Powers. *What are Syndication Feeds*. O'Reilly, 2005.

[42] Anirudh V. Ramachandran and Nick Feamster. Authenticated out-of-band communication over social links. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 61–66, New York, NY, USA, 2008. ACM.

[43] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.

[44] D. Sandler, A. Mislove, A. Post, and P. Druschel. Feedtree: Sharing web micronews with peer-to-peer event notification, 2005.

[45] Dave Winer. Payloads for rss. http://www.thetwowayweb.com/payloadsforrss.

[46] H. Wittenbrink. *RSS And Atom: Understanding And Implementing Content Feeds And Syndication*. Packt Publishing, 2005.