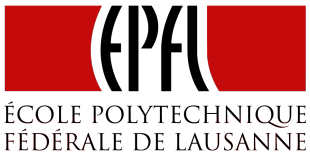
Indico: Behind the Scenes of CERN events



Ferhat Elmas
CERN Supervisor: **Pedro Macedo Alves Ferreira**
EPFL Supervisor: **Karl Aberer**
School of Computer and Communication Science
EPFL

Master Project Report
Distributed Information Systems Laboratory
14 Mar 2014

Abstract

Indico is a web application to schedule and organise events, from simple lectures to complex meetings, workshops and conferences with sessions and contributions. The tool also includes advanced user delegation mechanism, paper reviewing, archiving of event materials. More tailored features such as room booking and collaboration are provided via plug-ins.

Indico is heavily used at CERN and in more than 10 years, very different features are added according to the needs of increasing number of users. However, data store(ZODB) have never changed. ZODB has been very important for Indico to have been Indico but it's a bottleneck now in terms of scalability and elapsed time in feature development.

Indico demands more scalable backend to serve increasing number of events with fancy features. The aim of the project is to replace ZODB with a highly scalable data layer which will enable global Indico, bringing ∼120 different Indico servers around the world together. First part of project involves the survey of different technologies according to the needs of Indico and second part is composed of preparation of transition environment with a prototype.

Keywords: SQL, NoSQL, Python, Web Development

Contents

1

Introduction

1.1 Overview - What is Indico?

Indico (Integrated Digital Conferencing) is a web application for event management, initially developed by a joint initiative of CERN, SISSA, University of Udine, Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek (TNO), and University of Amsterdam. Currently, main development is going on at CERN with occasional contributions from outside such as time zone support by Fermi Lab.

Indico events are divided into 3 types: lecture, meeting and conference as from simplest to the most complex, respectively. Lectures are one time talks but conferences provide all nice features of Indico.

In addition to events having a type, each event belongs to one category which enable category-driven navigation between events. Since finding interested event in possibly huge number of categories, time-driven (day/week/month) view is also supported. Even time-driven navigation isn't enough because a user, extensively uses Indico, can easily see tens of events in a daily view. To remedy this problem, personal dashboard is being developed in which approaching events are shown in sorted order for the user. Moreover, users can mark some categories as their favorites and Indico recommends categories to users by analyzing this information and the event history of users.

Indico manages full conference cycle starting with customization of website. Every part of conference management requires different access rights. One contributor should be able to modify her contribution but she shouldn't be able to others, for instance. Indico makes setup of access and modification rights easy. Conferences are nothing without registrants and Indico provides a powerful fully customizable wysiwyg registration form. Indico supports creating custom forms in addition to registration form such as to prepare a survey. Call for abstracts, rich text editing in Latex and Markdown, and paper reviewing are enabled by Indico. Indico is a document store as much as being metadata store but saving different kinds of files; pictures, slides, videos, etc. Indico is even capable of converting documents betweeen different formats such as automatic PDF generation from Microsoft Office slides. Everybody has her different flow and tools to manage her calendar. Indico allows different views of same event and exporting events in multiple formats such as HTML, XML, iCal, PDF, etc. Last but not least, Indico is extensible via plug-ins to integrate with other systems or to provide more localised services so that only Indico is sufficient for everything. Room booking to arrange event place or Vidyo for video conference may be given as examples of notable plug-ins.

Extensible architecture of Indico have enabled easy integration with different services. As time advanced, search engine (Invenio) is added to find catgories and events easily. Indico totally embraces mobile and recently mobile check-in application is developed. Video conference (Vidyo and EVO) and instant messaging (XMPP) are easy to use within Indico. Payment processing for registrations in different ways (PayPal, PostFinance, WorldPay, etc.) is possible. Location, room and booking management are made possible with a clear interface for places and schedule of events. As a result, Indico is absorbing new feature sets by integrating with other services and steadily advancing to become the de facto interface for most of the needs.

1.2 Motivation - Why change?

Main driving force in development of Indico was a required service to manage event as following agenda of CDS, CERN Document Server. Thus, rapid production ready software was a necessity. Since there were many features to be implemented, time spent in other parts of the system must have been minimized.

The beginning of 2000s was the time objects and object oriented programming languages have become mainstream. As a result, there was an opportunist storm to try objects in everywhere and data stores had their own lion's share. Thus, an object database as a back-end for web application was a meaningful decision at that time. However, object databases never took off.

ZODB (Zope Object Database) was a viewable option in 2002 for Python, main programming language of Indico. Since ZODB is an object database, it enables directly manipulating and transparently storing objects. This capability of data store eliminates a layer between objects in programming language and data in store which in turn, saves a lot of time, money and complexity in application.

Fast forward to 2014, ZODB is far from having medium sized community. What is saved before by using ZODB is now being paid to solve limitations of ZODB. Finally, net gain turned into negative which signified to replace ZODB with more mainstream, powerful and scalable data store which will support next decade of Indico.

1.3 Approach - How change?

Firstly, limitations of ZODB should be extensively studied because most of the time, leaving aging production system behind and writing it from scratch results in failure. It's developed for more than 10 years, leaving ZODB means discarding this accumulated knowledge. Moreover, unless problems are made ridiculously clear, clean solutions can't be found.

We will present these reasons and the ways how we tried to solve these limitations. After we become certain about that refactoring would be more cost-effective, next step is to define criterias for the replacement technology in terms of where ZODB has failed and also keeping the future of Indico in mind.

In the golden era of data store, started by Google and Amazon to implement their extreme requirements, there are zillions of candidates, even barely knowing each of them is a really difficult task. Goal is to collect and classify candidates that conforms to criterias as much as possible.

After first elimination and classification, next round is to get familiar with each one via implementing a very small subset of Indico. Implementation is important because making sense of abstract concepts is difficult to understand and to interpolate expectations. Even if feature implemented is small, it makes easier to predict the schedule and complexity.

When know-how of each system is acquired, tipping point is choosing one or subset of candidates because in big applications, each part has different characteristics and requirements which maps into suboptimal solutions by using only one data store.

In either case, transition phase is long and painful so it should be well-planned in terms of bugs, needs of community and release cycle. The final bit is to implement the system conforming to the plan.

1.4 Outline - What is Done and Next?

In the following, we will give every tiny detail from the study of limitations of ZODB to new implementation with new polyglot database system. Moreover, database change involves refactoring huge body of code base so this project also aims to pay years of technical debt and to complete what we wish we had had in the beginning such as modular Indico. In conclusion section, how important steps are taken to modernise Indico code base and to prepare transition of Indico to Python 3 can be seen in addition to getting scalable by new back-end.

Even if important steps are accomplished, that's enough because software is very volatile and requirements and technologies are quickly changing. While keeping basic principles like DRY and KISS, extending data system for specific cases will help for better scalability and easening development of certain features such as statistics or social features.

2

Background and Related Work

In this chapter, we first look at some topological aspects of today's OSN, in the context of their impact on the structure of the underlying P2P infrastructure. We then focus on specific ways of offering such infrastructure, and focus on the SpiderCast protocol. We then survey the related work on building P2POSN, discuss their approaches, and point out their connection to our own work, when applicable. Finally, we present some insights about the global characteristics of the social graphs and P2P network graphs, which further motivate our work.

2.1 Online Social Networks

The inherent characteristics of OSNs, as they were implemented in the currently available client-server architectures, should be considered when designing a P2P platform able to sustain an OSN with a large userbase. They have been comprehensively studied following (a) a purely *topological view* [**?** **?**], focusing, for instance, on the structural position of their nodes, on their level of segmentation or connectivity, on the structural mechanisms that describe their evolution, etc; and (b) a *semantic view* [**?** **?**], which brings together structural characteristics and user data patterns (e.g. what type of information has a tendency to be further shared/retweeted and by what category of users, which are the most popular topics or users). All these characteristics may be used not only to help design the underlying architecture of OSN, but also to enhance P2P systems, recommendation systems, location based systems, trust computation and so on.

In the context of this work, we will look at the former category of studies, because we are mainly interested to see how the topology of the P2P platform impacts the performance at the OSN level (e.g. a user may not want to keep links to any other users and/or to have no links to his friends). For example, the first OSN is considered to be created by the users who exchanged email messages, as a social graph was formed in this way. Today, popular OSNs, such as Facebook [**?**], Twitter [**?**], LinkedIn [**?**] and Flicker [**?**], rely on such graphs to perform basic operations like sharing, organization or data location [**?**].

It was observed that, for any OSN, the social graph is formed from three different types of groups (i.e. sets of nodes) [**?**]: (a) *zero-degree nodes* - usually not having an active participation within the network, (b) a *giant connected component* - densely connected social network core, containing a large number of users and having at least one path between any pair of nodes (typically containing high-degree and highly active

nodes), and (c) *isolated communities* - small sets of nodes who interact only with each other. Note that in time the giant connected component tends to gather the vast majority of users [**?** **?**], up to 99% of nodes (as the isolated communities merge with it [**?**]). This property holds for all the networks used for our system evaluation, being also preserved with high probability by the obtained overlay, as we will see in Chapter 4. We will further analyze all these characteristics and more, making also a short comparison with P2P networks characteristics, in Section 2.4.

2.2 P2P Networks

This thesis argues about the feasibility of a P2P system that aims to minimize as much as possible the users' tradeoffs when moving away from a social network service based on the client/server paradigm. Thus, when architecting a P2P-based OSN, we also need to deal with P2P-specific aspects, such as data availability, updates, network topology, search, data locality, etc. For this work, we mainly focus on obtaining data availability, scalability, and inexpensive communication. In this section we will point out only aspects related with this goal.

The P2P model was made popular mostly by file-sharing and content distribution applications such as BitTorrent[**?**], eDonkey[**?**], Napster[**?**], Gnutella[**?**], etc. Apart from this, it is also used for real time communication between users. The nodes in a P2P network form overlays which can be classified as structured and unstructured. Even if unstructured overlays are often relatively simple, their search operations tend to be inefficient. On the other hand, structured P2P overlays usually use distributed hash tables (DHT) to perform directed searches, which makes lookups more efficient in locating data [**?** **?**]. Thus, for search performance, a structured overlay may be preferred. However, they tend to be more expensive to maintain especially under high churn rate, as their topology impose more constraints (which makes one think of the potentially large control message overhead and degraded routing performance). Unstructured overlays are arguably more resistant to churn, but it is commonly believed that they impose greater message overhead (as they generally use either flooding or random walks)[**?** **?**]. On the other hand, when the overlay network exhibits the clustering and short path lengths of a small world network, it ensures that even message flooding approaches work well (with a low time-to-live for messages, resulting in a low message overhead) [**?** **?**].

Thus, this motivates the following question: how to propagate updates while maintaining an overlay which is scalable, churn-resistant, small diameter and with a low maintenance cost for connections. Basically, as we presume that nodes in a P2P social network will primary store the data they publish on, we need a solution for building an overlay network that allows efficient topic-based event routing in a highly dynamic environment. In this regard, the SpiderCast protocol supports publish/subscribe communication in a decentralized environment [**?**]. This protocol aims to build an overlay that is both topic-connected and has a low per-topic diameter, while requiring each node to maintain a low average number of connections. In our P2P social network case, the topics are the same with the nodes' profiles, hence the number of topics is equal with the number of nodes and each node interest contains only his friends. Thus, this protocol fits our needs since we can see the P2POSN as a publish/subscribe system in which each user subscribes to his friends. In addition, we can efficiently use the principles behind SpiderCast protocol to reduce the average node degree by manipulating the common interests (friends) between

nodes for constructing a low-degree overlay in which all the nodes with a common friend form a connected component. Since the clustering coefficient is known to be high in social networks [**?** **?**] we expect to have a large number of duplicate paths, a good premise for SpiderCast to be successful in this environment. Furthermore, the experiments on SpiderCast show that the average node degree decreases with the number of nodes [**?**]. When keeping the number of nodes constant, the increase of the number of topics moderately increases the average node degree, demonstrating also a good scalability with the number of topics.

In our context, we should keep in mind that the availability of any specific file is not guaranteed in a P2P network. We will see that this can become a problem for a P2P social network, as for today's OSNs this is a basic requirement. Thus, we need a cost efficient solution able to ensure data accessibility with a high probability. Towards obtaining high availability for distributed application in an environment prone to a high churn rate, a solution for self-healing active replication on top of a structured P2P overlay is proposed in [**?**]. Yet, each data item is associated with a unique key and its replicas are placed on the closest nodes to its key. This may not be the best approach for us as we might need to consider, for replication, node attributes such as: users diurnal behavior, number of friends, access patterns, interaction patterns, or we should provide a mean of computing the key so as to consider these attributes when computing the distance between keys, and is challenging to consider all of them at once.

2.3 P2P Social Networks

A P2P network can be seen as a social network where users are the nodes linked according to their friendship relations; the similarities between the two have been previously explored in [**?** **?**]. Such networks become as large as hundreds of millions of nodes. The authors try to identify a pertinent set of opportunities and challenges for a P2P environment in order to support the deployment of a social network. Since this identification was made, services like Diaspora [**?**], considered a Facebook's young competitor and Identi.ca [**?**], a solution to decentralize Twitter, have started to appear.

Solutions for P2P social networking have come also from the research community. Gossple, which links users based on their similarities, is a network of anonymous social acquaintances that uses a gossip protocol for updates and communication [**?**]. Gossple provides a way to meaningfully associate and exchange information between users' profiles, without requiring peers to have a previous relation. This is an interesting approach as one of the reasons for belonging to a particular social network may be to meet people. Thus, a user may seek to meet people with similar interests, passions, background, problems, professions and hobbies, shifting the key focus on finding like-minded people. Basically, the nodes periodically gossip data about their interest profiles and based on this interest compute the distances between them and other nodes and create links. However, our work differs in a number of aspects as we do not look into preserving users' anonymity and to creating links based on a defined interest metric.

An approach closely related with ours is PeerSoN, which aims to overcome the OSNs limitations by looking to find solutions to the privacy issues (i.e. encryption, access control and decentralization) and permitting direct exchange of data between users even without Internet connectivity, while preserving OSNs features (e.g. data availability and accessibility) [**?**]. For ensuring that the private information is visible only by the right

Network	Average path length	Type (OSN or P2P)
Facebook[?]	4.8	OSN
Flicker[?]	5.67	OSN
Twitter[?]	4.12	OSN
Maze friends network[?]	7.69	P2P
Maze download network[?]	9.17	P2P

Table 2.1: Average path lengths for OSNs and P2P networks

users, PeerSoN does not assume any kind of trust relationship between peers, but provides access control by using encryption and key management. While similar solutions could be adapted to the work presented in this thesis, we believe that we first need to define the design of a scalable P2POSN underlying infrastructure.

LifeSocial.KOM is a P2P-based platform for secure online social networks which claims to provide the functionality of common online social networks in a totally distributed and secure manner [?]. It uses FreePastry for interconnecting the participating nodes and PAST for reliable, replicated data storage. Alas, it was tested at a very small-scale only.

In [?] a privacy aware decentralized OSN called Porkut is introduced. The idea behind it is to exploit trust relationships in the social network for decentralized storage of OSN profiles and to address availability and accessibility issues by considering users' geographical locations and online time statistics, while using k-anonymity techniques for ensuring a privacy preserving indexing. Yet, they store the data in plain text relying on social pressure and monitoring for data integrity and privacy. A solution, for providing reliable storage, is to offer proper incentives to nodes, based on existing social relationships, in order to make them to cooperate for storing data and to remain in the system [?]. However, this approach is suitable mainly for data intensive applications that do not require global visibility or global sharing.

There are also customized solutions such as SCOPE which is a prototype for spontaneous P2P social networking [?]. It provides the distributed database and lookup services deployed on DHT technology so as to support social networking in local areas. Finally, the Maze P2P network integrates both a friends' social network and a download network. The similarity between nodes' interest is revealed by the social network relationship within the Maze network [?]. Thus, Maze is improves the content search in the download network by using the "small world" property [?] of the friends' social network as it enables small search path length.

2.4 Brief Analysis of Social and P2P Networks Graphs

When trying to determine if a shift from a client/server to a P2P infrastructure is feasible, all the characteristics of today's OSNs such as clustering coefficient, path length, access patterns, interaction patterns, etc., need to be considered so as to make a realistic feasibility study.

Social graphs exhibits the "small world" property which practically measures to what extent a network can spread information quickly and widely [?]. It is obtained with a small average path length and a high clustering coefficient. We observe that while ONSs have their average path length lower than that computed for world population [6.6] [?

], the average path length for P2P networks is higher. The average path length values for some of the most popular social networks and some P2P networks are presented in Table 2.1. The values for average path lengths and clustering coefficients obtained for all these graphs (so as for any "small world" network) compared with those obtained for a corresponding random network, with the same number of nodes and average degree per node, have an average path length close to the random network, while the clustering coefficient is much larger [**?**]. The data in these papers suggests that even if for P2P networks the path length is slightly bigger and the clustering coefficient slightly lower, they seem to have similar topological structure. More, with redundancy and some solution for information availability P2P social networks can reach "six degrees of separation" and in a P2P network we can take advantage of this short average path length. For example, flooding with relatively small time to live would cover most of the graph. In addition, if this is correlated with a large clustering coefficient, interest based clusters of peer can be identified. These features can influence the performance for operations such as information dissemination, friends search, etc., in a P2P social network.

Some very interesting findings show that in unstructured Gnutella-like P2P overlays: a) the long-lived peers form a densely connected and stable core, which leads to stable and high connectivity among peers even in the presence of high churn and b) the network dynamics lead to a core overlay that exhibits onion-like biased connectivity (peers form "layers" depending on their uptime - peers with shorter uptime form a layer by forming connections with each other and with peers with higher uptime) [**?**]. This is quite similar with how the OSNs evolves in time: old users have a tendency of having larger friendbase and to move closer to the social network core and to also form a densely connected core [**?**].

Considering only the "static" social graph does not suffice. A proof of the fact that social links are not valid indicators of real users' interactions, for the biggest OSN Facebook, is given in [**?**]. There is a high skew distribution of users interaction, for instance less than 1% of Facebook users interact with more than 50% of their friends. Consequently, the social links should include data about users' interactions. For 50% of users, 70% of interaction comes from 7% of friends and 100% of interaction comes from 20% of users. Basically, this also mean that the interaction graph can eliminate useless links in the "static" social graph, as the results correspond with theoretical limits on human social cognition[1] [**?**]. These invalid links are the reason why, for instance, the OSNs exhibit lower values for path length than those obtain for world population. 99% of Facebook users interact only with less than 150 friends. While the interaction links weight (i.e. the number of interactions between to users) could be used to further enhance our P2POSN system design by giving priority to stronger links (i.e. links with higher weight), this work focus on keeping all node's friends profiles accessible, regardless the interaction patterns.

However, while users' visible interaction links provide clues on how to design a social platform, the majority of user interactions in OSNs are latent interactions (e.g. browsing a user profile) [**?**]. This study is made on Renren, the largest OSN in China, on 42 million users, 1.66 billion social links and detailed histories of profile visits over a period of 90 days for more than 61,000 users. It analysis the latent interaction graph which basically models users' browsing behavior. In addition, it shows that the latent interaction graph has a clustering coefficient (0.03) lower than those obtained for social graph (0.18) and

[1] This limit is given by Dunbar's number (150) which represents the maximum number of stable social relations that one can maintain.

visible interaction graph (0.05), while in terms of connectivity it falls between them [**?**]. These characteristics could be relevant for us mainly for data location and replication. They allow the weighting of all the social links, in order to give priority to some particular links as well. Yet, this extension is beyond the scope of our work.