Designing Structured Peer-to-Peer Architectures for Mashups
Osama Al-Haj Hassan, Ashraf Odeh, Adi Maaita, and Aymen Abu-Errub

Abstract—Mashup platforms are popular because they provide end-users with a powerful tool to query the Web. The majority of these platforms are designed using centralized architectures or loosely coupled distributed architectures. The lack of structure in these platforms makes searching for mashups an expensive task. In this paper we present two structured peer-to-peer architectures for mashup platforms, namely, MashChord and MashCAN. These platforms increase the efficiency of searching for mashups. In addition, our design makes sure a mashup is hosted at several peers which strengthens the reliability of the system. Moreover, execution offloading feature is added to our design to support the functionality of each peer in the system.

Index Terms—mashups, peer-to-peer, Web-2.0, structured, search, Chord, CAN.

I. INTRODUCTION

Today’s Web focuses on topics such as semantic web [1], [2] and social networks [3]. In these types of applications, one of the most important features is personalization which is the hype of Web 2.0. One of Web 2.0 applications that offers personalization is mashup platforms. Mashup platforms empower end-users with a useful tool to search the Web in a personalized manner. They provide end-users with an interface to design data mashups. A mashup is basically a tree hierarchy that starts by fetching data from several data sources across the Web and proceeds by refining this data by using operators such as filter and truncate and ends by dispatching final result to end-user. One of the most popular mashup platforms is Yahoo Pipes [4]. An example of a mashup is shown in Figure 1 which fetches data from sport feeds, then filters the data based on containment of the term ‘Rafa Nadal’.

Mashup platforms support personalization because they enable each end-user to create his own mashups. This is unlike web services which are designed for groups of people. Therefore, the number of mashups that a mashup platform hosts is expected to be very high. As a consequence, scalability issues arise for mashup platforms which requires careful attention.

Two existing architectures for mashup platforms are loosely distributed architecture and centralized architecture. Both architectures suffer from few drawbacks. In a loosely distributed architecture, peers of the network provide capability of designing and executing mashups. End-user uses this capability to design mashups of his own need. The advantage of using such an architecture is distributing load between network peers which increases the scalability of the system. However, the way communication takes place in loosely distributed architecture is by exchanging messages using flooding or some sort of random walks. This is an inefficient way of communication because if a user at one end of the network is looking for a mashup that exists at far end of the network, then messages have to be exchanged between neighbors gradually until the whole network is covered and the mashup is found. This is huge amount of messages needed for one search attempt. The problem exacerbates if the network is facing high traffic of search attempts.

Centralized architectures consist of one server which has the capability of designing and executing mashups. So, end-users use their machines to connect to this server, design and execute their mashups. The positive points of using this architecture is simplicity and a direct communication between client and server which results in a cheap search process. But, the system might fail if something wrong happens to the centralized server. In addition, the server might not be able to handle large number of users connecting to it. As a consequence, this architecture suffers from low reliability and scalability.

To overcome these problems, we extend our work in [5] by providing MashChord and MashCAN which are two designs for mashup platforms based on two popular peer-to-peer architectures, namely, Chord and CAN. Adding structure to a mashup platform increases its reliability and scalability. In addition, we add mashup execution offloading as a mean of relieving load on each peer in the network.

II. LITERATURE REVIEW

Our system represents mashup platforms over structured peer-to-peer topology. Therefore, our literature review will discuss the two aspects of mashup platforms and structured peer-to-peer networks.

A. Mashup Platforms

Mashup platforms are becoming very popular Web 2.0 applications. They have been investigated in literature. One famous mashup platform is Yahoo Pipes [4]. It is a platform that enables end-user to build mashups by providing a set of operators such as fetch, filter, and sort operators. The mashups built using Yahoo Pipes extract data from several types of data sources such as RSS and Atom feeds. Mash-Maker [6] is a mashup platform that enables end-users to extract data sources and populate them in a visual manner. Marmite [7] is a tool that helps end-users to aggregate several data sources and direct the end result to other files. This tool is implemented as a Firefox plug-in. MARIO [8] enables