Computer Networks 57 (2013) 3759–3772

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

On the performance improvement of gossip protocols for content-based publish-subscribe through caching

Angelos-Christos G. Anadiotis [a,*], Charalampos Z. Patrikakis [b], Iakovos S. Venieris [a]

[a] School of Electrical and Computer Engineering, National Technical University of Athens, Iroon Politechniou 9, 15773 Athens, Greece
[b] Department of Electronics Engineering, Technological Education Institute of Piraeus, Petrou Ralli & Thivon 250, 12244 Aegaleo, Greece

ARTICLE INFO

Article history:
Received 13 March 2013
Received in revised form 27 August 2013
Accepted 29 August 2013
Available online 12 September 2013

Keywords:
Publish-subscribe
Gossip
Caching

ABSTRACT

Gossip protocols are a common solution for content-based publish-subscribe. However, in spite of the advantages that these protocols present in comparison to other approaches, such as distributed hash tables, they also have inherent scalability issues. In this paper, CCPS (Caching Content-based Publish-Subscribe) is presented as a publish-subscribe protocol designed to address these scalability issues. Through caching, CCPS exploits search query repetition to address load balancing problems. It has been designed having in mind its use in combination with existing publish-subscribe solutions so as to improve their performance. In the paper, the protocol description together with the implementation details of its fundamental operations are provided, while CCPS performance is evaluated against standard gossip protocols.

1. Introduction

Publish-subscribe has been a hot topic in distributed systems quite some time now and, depending on its application, it is categorized [1] as follows:

- Topic-based: topics are usually seen as identifiers of groups of peers that are given as keywords and mostly URLs. Subscription in such systems means that the subscriber becomes a member of a specific group corresponding to the topic.
- Content-based: the subscriptions are expressed using properties that can be even partially matched against the content (publications). This scheme is more flexible than topic based since there is no need for any predefined scheme for the topics.
- Type-based: this scheme is similar to topic-based. The difference is that there is no externally specified, unique name. On the other hand, there is a type which is usually inferred by the nature of the subscription itself.

Systems of all three categories can of course be combined in order to achieve better performance. For example, peers could be first categorized based on topics and then use content-based subscription for the shake of expressiveness. This paper focuses on the content-based publish-subscribe scheme, where subscriptions carry keyword-based queries supporting partial matching of content.

Even though there are several ways to support content-based publish-subscribe, the use of gossip-based approaches is fostered when content matching should be performed in the context of the publishing peer. A common situation that this case applies is when the publisher wants to perform complex operations on the particular content to facilitate its retrieval. For example, the publisher can use rigorous semantic structures and algorithms, focused on the types of content it publishes.

A typical publish-subscribe system for content discovery considers that a subscriber expresses interest on content using certain criteria and receives notifications as soon as such content becomes available. In this work, this asynchronous operation is extended by also supporting notification for already existing content that matches the subscriber's requirements.

[*] Corresponding author. Tel.: +30 2107724323.
E-mail addresses: aca@icbnet.ntua.gr (A.-C.G. Anadiotis), bpatr@teipir.gr (C.Z. Patrikakis), venieris@cs.ntua.gr (I.S. Venieris).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2013.08.030

While the scalability issues of topic- and type-based publish-subscribe systems can be handled by organizing peers into groups based on certain criteria, supporting content-based publish-subscribe in a P2P system can be quite challenging. On the other hand though, the content-based approach is the most flexible one since a subscription can better reflect the users' requirements, while it can easily support emerging search paradigms, such as semantic search.

One way to reduce the overhead in publish-subscribe systems is to decouple the content from the publication. By this, the publication does not have to include the content itself. Instead, it contains the metadata used for matching that content with subscription queries. This is usually performed through proprietary techniques employed by each publish-subscribe protocol. In this paper, the data structures introduced for this reason by the CONVERGENCE FP7 project [2] are considered. CONVERGENCE, adopting and extending the notion of the standard MPEG-21 Digital Item [3], has introduced the Versatile Digital Item (VDI), a data structure used as a container to represent an entity, from digital content to real world objects and principals. Through the use of these data structures, only the necessary matching information is carried by the peers, thus reducing the bandwidth and storage requirements. Nevertheless, the proposed Caching Content-based Publish-Subscribe (CCPS) protocol can support any kind of such structure, with its design being VDI independent; the use of VDIs is provided just as an illustrative example followed by the publish-subscribe implementation in CONVERGENCE.

1.1. Rationale

CCPS is a protocol designed to address the scalability and load balancing issues of gossip protocols used for content-based publish-subscribe, by exploiting search query repetition and introducing a caching mechanism which stores information about subscribers sharing common interests. Such interests are expressed through the subscription queries. The cache is implemented as a Distributed Hash Table (DHT) to guarantee scalability, while its performance is proven to increase with the query repetition probability. The description of the protocol in the paper is accompanied by both a theoretical model, based on the observations of Smyth et al. in [4] and simulations which give proof on the improvement in terms of overhead being almost equal to the caching probability and reaching up to the query repetition probability.

Another contribution of CCPS is in the addressing of the load balancing problem. CCPS proposes a model where the cache contains information about other peers (subscribers) that have already subscribed with the same query. Thus, future subscribers will *share* this information with the existing ones. This sharing is performed using a P2P approach and implementing standard protocols for content sharing, such as BitTorrent. By this, the publishers do not have to send the content each time to every matching subscription, while the subscribers themselves are distributing the overhead for exchanging the subscriptions between them. It should be noted at this point that BitTorrent will be used at the rest of the paper as a representative of a standard and widely used P2P content sharing protocol. CCPS has no dependency of the use of BitTorrent, so any alternative P2P sharing protocol can be used.

Caching has also another important effect in the system performance. The query propagation in the overlay of peers is performed using a gossip protocol, as recommended in the literature [5–7]. However, due to the probabilistic nature of the gossip protocol, peers can receive the same query from different sources, while if there is a match, the same replies are sent to the subscriber by different publishers. Furthermore, the same content may be published by multiple peers (especially in community based systems), which again causes result duplication, and thus, bandwidth and processing power misuse. In CCPS, the subscribers keep only one copy of the published content and, therefore, no matter how many times it has been published, each subscriber will receive only one copy. Moreover, since gossiping rounds are avoided due to the cache, peers neither receive nor reply with the same publication more than once.

The rest of the paper is organized as follows: Section 2 presents other protocols that have been proposed for publish-subscribe. Section 3 includes the description of the proposed protocol both for content discovery and content sharing. In Section 5, the protocol implementation is provided by giving the algorithms used for content publication and subscription. The algorithms have been implemented in a simulator used to perform the experimental evaluation. These results have been used in Section 6 to confirm the corresponding theoretical model also presented in the same section. Finally, in Section 7 conclusions as well as future work on content-based publish-subscribe systems are summarized.

2. Related work

Several protocols focusing on different aspects of publish-subscribe have been suggested, with most of them considering subscriptions being matched against future publications. This work considers an extended scenario, where users expect that their subscriptions are also matched against older publications. This can be seen as a distributed search problem. Moreover, publishers are organized in a peer to peer overlay, while the subscription queries are keyword-based and support partial matching.

DHT based systems, such as CAN [8], Pastry [9], Chord [10] and Kademlia [11], are a common solution in the literature for publish-subscribe operations. However, such systems are following the key-based routing approach, thus failing to provide an effective solution in keyword-based search and especially partial matching which is a common requirement in content-based search/subscribe [12]. On the other hand, content routing (of queries) which is an essential element of keyword based subscription can be achieved either through flooding or random walks. Given though that the latter does not yield a compelling hit rate [12], a combination of them will be considered, in order to achieve higher user satisfaction.