Computer Communications 41 (2014) 55–66

Contents lists available at ScienceDirect

Computer Communications

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

User-defined consistency sensitive cache invalidation strategies for wireless data access

Sunho Lim [a,*], Yumin Lee [a], Jongpil Cheon [b], Manki Min [c], Wei Wang [c]

[a] T^{2}WISTOR: TTU Wireless Mobile Networking Laboratory, Dept. of Computer Science, Texas Tech University, Lubbock, TX 79409, United States
[b] Dept. of Educational Psychology and Leadership, Texas Tech University, Lubbock, TX 79409, United States
[c] Dept. of Electrical Engineering and Computer Science, South Dakota State University, Brookings, SD 57007, United States

ARTICLE INFO

Article history:
Received 23 May 2013
Received in revised form 21 December 2013
Accepted 26 December 2013
Available online 18 January 2014

Keywords:
Cache invalidation
Lazy request
Multiple data transmission
Opportunistic data access
User-defined consistency

ABSTRACT

In order to fulfill users' insatiable interests in accessing Internet services and information wirelessly, one of the key optimization techniques is caching frequently accessed data items in a local cache. A strong consistency is implicitly assumed in most caching schemes but it may cause a long query delay. In this paper, we propose a consistency-sensitive cache invalidation scheme, called ConSens, based on the existing Invalidation Report (IR) and Updated IR (UIR) based cache invalidation frameworks. In the ConSens scheme, each user is able to set its own consistency level with a server independently. This user-defined cache consistency can support diverse consistency requirements of applications. We also propose both lazy request and opportunistic data access techniques to effectively balance the data accessibility and query delay. In addition, we enhance the IR-based cache invalidation mechanism and propose a multiple data transmission scheme, called MDT, to further reduce the query delay. Extensive performance evaluation studies show that the proposed strategies can effectively balance the data accessibility, reduce the query delay, and significantly increase the number of opportunistic accesses according to the user-defined consistencies.

1. Introduction

Recent technological advances in high-speed wireless network, mobility support, and portability enable mobile users to access the Internet wirelessly. For example, smart phones have been receiving a tremendous attention and are becoming popular. 115.8 million users in the U.S. already own a smart phone in 2012 and it is expected to increase upto 192.4 million by 2016 [1]. 47.7% of all mobile subscribers in the U.S. are smart phone users in 2012 and it is expected to increase upto 74.1% by 2016 [2]. Approximately seven out of ten American adults access the Internet wirelessly anyhow. Typically, the explosive rise in the number of application software running on smart phones, called apps, has fueled mobile users to easily and frequently access the Internet wirelessly anytime and anywhere. According to Apple AppStore, more than 800K apps are active and more than 50 billion downloads have been made as of May 2013 [3].

In order to fulfill mobile users' rapidly growing interests in accessing Internet services and information wirelessly, one of the key optimization techniques is caching frequently accessed data items in a local cache. Caching techniques can relieve the network traffic and improve information accessibility and availability. Thus, the communication cost in terms of wireless bandwidth, battery energy, and latency can be reduced significantly. A great deal of research effort has been devoted in developing various caching strategies in wireless and/or mobile environments. Here, we use a mobile user to refer to a wireless mobile device or a person who carries it. Thus, we use the terms mobile users and mobile nodes (later in short, nodes) interchangeably.

Most caching strategies implicitly assume a strong consistency between the original data item and its cached copy [4–12]. When a node generates a query, the query should be answered by the recently updated data item stored either at a server or a local cache. However, ensuring a strong consistency requires a non-negligible long query delay to confirm that both the source data item and its cached copy are consistent. For example, update-sensitive weather information such as tornado, hurricane, or tsunami requires a real-time update operation. News, stock prices, and traffic information are also update-sensitive. A cached copy must be updated before answering a query when the source is updated at the server. However, ensuring a strong consistency is not always a prompt and

* Corresponding author.
E-mail addresses: sunho.lim@ttu.edu (S. Lim), yumin.lee@ttu.edu (Y. Lee), jongpil.cheon@ttu.edu (J. Cheon), manki.min@sdstate.edu (M. Min), wei.wang@sdstate.edu (W. Wang).

critical requirement. This is because a requirement of data consistency (later in short, consistency) can be different depending on the update sensitivity of information. For example, maps, video clips, e-flyers, and weather information are not update-sensitive. They may not need to reflect the current update of the source stored at the server in an urgent manner. Occasional inconsistencies between the source data item and its cached copy would be acceptable. In the Bing™ [13], the weather information encapsulated in a Web browser toolbar is not always latest updated (e.g., 45F as of 25 min or 1 h ago). Some mobile apps are not urgent to update their contents, such as mobile education game, travel, or magazine.

Under the consideration of diverse information and different update sensitivities, we propose a *consistency-sensitive cache invalidation* scheme, called *ConSens*. In the ConSens scheme, each node is able to set its own consistency level, called *target consistency*, with the server independently. This user-defined cache consistency is implemented by a minor modification of the existing Invalidation Report (IR) and Updated IR (UIR) based cache invalidation frameworks [4,5]. We also propose a *multiple data transmission* (MDT) scheme to enhance the fundamental IR-based cache invalidation mechanism [4].

Most cache invalidation strategies [4–12] often assume that (i) the validity of cached data items is determined by a strong consistency and/or (ii) the entire nodes have the same consistency with the server. However, the proposed approach is quite different from the conventional cache invalidation strategies, and our contributions are summarized in four-fold:

- First, we propose a consistency sensitive cache invalidation scheme, in which each node can flexibly and independently set its own consistency with the server. A consistency condition is developed to decide the current consistency based on the number of invalid cached data items accessed during a consistency window.
- Second, both *opportunistic data access* and *lazy request* techniques are also developed to improve the data accessibility but reduce the query latency. Depending on the consistency condition, each node can judiciously access the cached data items even without sending a request message to the server and waiting for the incoming IR (or UIR) from the server.
- Third, we also propose a multiple data transmission (MDT) scheme to further reduce the query delay in the IR-based cache invalidation mechanism. In the proposed scheme, the server repeatedly broadcasts a set of requested data items separately. Then each node that has sent a *request* message is able to individually answer a query as soon as receiving the requested data item.
- Fourth, we integrate the proposed techniques with the existing IR- and UIR-based cache invalidation frameworks [4,5]. We modify two major IR- and UIR-based cache invalidation schemes to implement the proposed consistency-sensitive data access and multiple data transmission techniques, ConSens-IR, ConSens-UIR, ConSens-IR-MDT, and ConSens-UIR-MDT.

We conduct extensive simulation-based studies of the proposed schemes to observe the impact of query interval, update interval, and target consistency on the communication performance. The simulation results indicate that the proposed schemes not only can increase the data accessibility but also can reduce the query latency significantly.

The rest of paper is organized as follows. The prior work is analyzed and the proposed strategies are presented in Sections 2 and 3, respectively. Section 4 is devoted to performance evaluation and comparison. Finally, we conclude the paper with future research directions in Section 5.

2. Related work

Most cache invalidation strategies deploy one or combination of the following design aspects: (i) Whether to maintain cache status or not (e.g., stateful or stateless); (ii) Who initiates cache validity (e.g., push, pull, or hybrid); and (iii) Level of cache consistency (e.g., strong, weak, or probabilistic). In this paper, we focus on the consistency dependent cache invalidation techniques.

2.1. Stateful or stateless

A server may or may not maintain any cache status of connected or disconnected nodes. In a stateful approach [6–9], the server keeps track of which node caches which data item (s). Whenever a data item is updated, the server broadcasts an IR to inform nodes of the cache update. In [8], a server-based poll-each-read (SB-PER) is proposed under the assumption of available global access and update information in the server. However, obtaining global update information is practically hard, if it is not impossible. For example, data items such as News, stock prices, or traffic information are not updated on a regular basis but are updated by an event, which is not pre-scheduled. In this paper, we do not consider the data items characterized by a scheduled update. In a stateless approach [4,5,10], however, the server is not aware of any cache status and thus, it periodically broadcasts an IR (or UIR) containing the limited amount of prior update histories. Here, the query delay is affected by the IR (or UIR) size and broadcast interval.

2.2. Push, pull, or hybrid

The push, pull, and hybrid techniques specify who initiates cache invalidation operation. In the push approach [4–10], the server initiates the cache invalidation, for example, by broadcasting an IR. In the pull approach [14], however, whenever a query is generated which can be answered by a cached data item, a *request* message is sent to the server for verifying the validity of the cached data item before it can be used for answering the query. In [8], an on-demand approach is proposed for fast moving vehicles in a multi-cell environment, where a query is forwarded to the server either for validating the cached data item or receiving the queried data item. In particular, when a vehicle handoffs, it sends the *ids* of the entire valid cached data items to the server for validity check and receives an *invalidation* message from the server accordingly. Several push- and pull-based schemes [11] and their combined schemes [15,16] have been proposed under the consideration of tradeoffs between query delay and communication overhead.

2.3. Strong, weak, or probabilistic

Various IR-based cache invalidation strategies have been widely used in wireless and/or mobile environments [4–12], where a server broadcasts an IR periodically (i.e., every L second, where L is a broadcast interval). A node can answer a query with a cached copy after validating it with the incoming IR. This ensures a strong consistency but the long query delay (i.e., $\frac{L}{2}$ in average) is unavoidable. Although a set of updated IRs (UIRs) [5] is broadcasted between two successive IRs to further reduce the query delay, non-negligible query delay (i.e., $\frac{L}{2 \cdot m}$ in average, where m is a number of UIRs) is still expected. The strong consistency ensures that only recently updated data item is accessed for answering a query, but the consistency maintenance cost in terms of wireless bandwidth, battery energy, communication overhead, and query delay increases.

A time-to-live (TTL) based approach have been deployed in Web and/or mobile environments [17,18]. In [18], an estimated TTL