Computer Communications 36 (2013) 750–757

Contents lists available at SciVerse ScienceDirect

Computer Communications

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

Evaluating per-application storage management in content-centric networks

Giovanna Carofiglio [a], Massimo Gallo [b], Luca Muscariello [b], Diego Perino [a,*]

[a] Bell Labs, Alcatel-Lucent, Nozay, France
[b] Orange Labs, France Telecom, Issy Les Moulineaux, France

ARTICLE INFO

Article history:
Available online 16 February 2013

Keywords:
ICN caching
Resource management
Service differentiation

ABSTRACT

Content-centric networking proposals have recently emerged to redesign the Internet architecture around named data rather than host addresses. Such designs advocate the usage of widely distributed in-network storage, with direct impact on end-user performance and network provider costs.

In this paper, we investigate the role of storage management schemes designed to deal with traffic of different applications. First, we show the impact on user performance, service provider and network cost of a static per-application storage allocation using measured traffic traces. Then, we analyze the performance of this static partitioning scheme by means of simulations with synthetic traffic traces. Finally, we evaluate two mechanisms for dynamic storage management, namely strict priority and weighted fair allocation, designed to overcome static partitioning limitations in presence of content time-to-live and of dynamic traffic patterns.

1. Introduction

The exponential growth of the amount of content in the Internet observed in the last years, has pushed content and service providers to deploy dedicated infrastructures for content delivery (e.g. CDNs, HTTP proxies). Such architectures are typically designed for one specific application (e.g. VoD) in order to improve and provide performance guarantee to users' and to optimize different metrics across the network, as bandwidth utilization and server load.

To reduce costs and enhance control, many network operators have recently started to build their own content delivery infrastructures instead of delegating this task to CDN vendors (e.g. Akamai, Limelight, CDNetworks). CDNs, traditionally conceived as separate infrastructures, are so evolving towards in-network solutions where storage capabilities can be distributed deep in the network from the backhaul to the home gateway. Classical CDNs cannot deploy their technology that close to customers without ad hoc partnerships with carriers.

Recent content-centric proposals [1], like CCN/NDN, DONA, NetInf, PURSUIT, go a step forward, making the content a first class network entity. Accordingly, content is identified, addressed and retrieved by its name independently from its location. Therefore, every router can store the data packets it forwards, in order to serve future requests for the same packets. To this aim, router built-in buffer memories are exploited, possibly enhanced with additional memory modules [2]. Storage becomes then available in all network nodes, and it can further improve the performance experienced by end-users while reducing the transport cost for network providers [3].

Content-centric architectures are naturally designed to carry multiple services over a single network infrastructure, whereas CDNs are typically tailored to deliver one specific service and dimensioned according to the requirements of the given application. This is a compelling feature as a service dedicated infrastructure is expensive in terms of deployment and management, and does not scale with the growing number of services.[1] In addition, a common content-centric infrastructure represents a more suitable solution for network operators as it can result in simplified management, enhanced control and better resource allocation. On the other hand, in a content-centric network, a non trivial task is to decide how resources (processing, bandwidth and storage) should be allocated and managed among applications.

In this paper, we focus on per-application storage management schemes designed to predict end-user performance and control providers' costs for all running services. Per-application management allows to allocate more resources to applications with strict performance requirements, to the detriment of others applications that would not suffer of a user experience degradation, with a

* Corresponding author. Tel.: +33 1 3077 6164.
E-mail address: diego.perino@alcatel-lucent.com (D. Perino).

[1] The term "application" refers to a set of services deployed by different providers and characterized by same performance requirements, document size, content popularity and traffic patterns, e.g., HTTP Web, live streaming, VoD, P2P file sharing.

0140-3664/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2013.01.011

potential increase of the overall network cost. For instance, per application management can be used to provide more resources to HTML pages, as the user experience mostly depends on their download latency rather than the download latency of their depended objects, as images or videos [4]. Also, per-application management allows running management algorithms tailored for specific applications, that are not suitable for other content, over appropriate packet only. For instance, per-application management can be used to run ad hoc management schemes over media streaming packets [5].

In our preliminary work [6], we have focused on in-network storage management and explored its potential to accommodate the traffic of different application by means of experiments under fairly realistic network conditions. In this paper, we extensively evaluate by means of packet-level simulations the storage management schemes we have introduced in [6]. First, we show the impact of *static storage partitioning* (SP) among different applications on user performance, service provider and overall network cost, and deeply analyze the interplay of different parameters using measured and synthetic traffic traces (Section 3). In the second part of the paper, we extend our study to *dynamic storage management* techniques to account for content Time-To-Live (TTL), i.e. limited temporal validity, and we show how this approach allows to overcome the limitations of a static storage allocation in terms of resource under-utilization (Section 4). To this end, we evaluate the *priority* (PSM) and *weighted fair* (WFSM) storage management schemes: these solutions address two common service differentiation objectives, namely strict resource prioritization and proportional fairness respectively, while remaining simple to implement with no additional complexity in terms of re-configuration.

2. Content-centric networks

In this paper we focus on the *content-centric networking (CCN)* proposal by PARC [7], and currently investigated within the NDN (Named Data Networking) NSF project. While our analysis is carried out for the CCN network architecture, most of the results generalize to other content-centric proposals and to telco CDNs based on the following communication principles.

Let us now briefly recall how a content-centric network operates. In CCN, content items are identified through a unique name in the standard form of a URI (Universal Resource Identifier) and are split into self-identified packets. Naming structure must be hierarchical (for scalability) and a name may also include application and time information (like time to live) depending on the nature of the content itself (data, video, voice or other). This information can be specified by the content provider directly, or enforced by a network operator. Once a content item is published into a network repository, a user can retrieve it by emitting per-packet requests through a protocol control unit called *interest*. The CCN transport paradigm is receiver-driven as no data packet is sent unless requested by the user. Without loss of generality we assume a simple transport protocol with fixed unitary window. Indeed, our results also apply to other transport designs because of the timescale separation between transport and caching dynamics [8].

Every network node is equipped with a local cache (*content store*, CS), where data packets are transparently stored to serve future requests. The CS implements replacement policies like FIFO or Least Recently Used (LRU). Upon reception of an interest from an input interface, every node forwards it to the interface indicated by the *Forwarding Information Base* (FIB) only if the given packet is missing in its local CS. Otherwise, the request is satisfied by a CS hit and the data packet is delivered to the user, following the reverse path of the interest.

Whenever an interest is forwarded to a given interface, an entry is created in the *Pending Interests Table* (PIT). The PIT keeps track of pending interests and interface (s) from which the request is received, in order to send back data along the reverse path. Once the packet is eventually sent down to the user, the related PIT entry is removed and the data packet is stored and forwarded by all network nodes along the path. The PIT additionally allows to filter multiple interests on the same data packet, usually coming from different users requesting the same content. In fact, in this case, if a PIT entry for the given packet request already exists, the interest is not forwarded. This mechanism reduces the amount of forwarded requests and has an impact on both transport and storage dynamics.

3. Protecting application performance by storage partitioning

Storage management in CCN routers becomes a fundamental issue in presence of traffic belonging to multiple applications competing for the available resources. Indeed, user performance and provider costs are affected by the average packet hit probability at different network nodes, i.e. the probability to find a requested packet at a given content store [6,3].

In this section we evaluate the impact of storage partitioning among different applications by means of simulations. *Storage partitioning* (SP) [6] refers to a static per-application storage allocation, where each application is statically assigned a fraction of the content store memory space. As previously mentioned, the application information can be assumed to be directly included or easily inferred from the packet name. Therefore, packets can then be easily recognized and inserted in the corresponding partition, where the replacement policy is independently applied. As a baseline for our evaluation we consider the *Shared storage management* (SH) scheme, that does not discriminate among applications, and where the replacement policy is independently applied over all packets. In our evaluation we assume LRU is used as packet replacement policy.

In 3.1, we analyze the impact of SP on CS dynamics with measured traffic traces. Then, we deeply investigate the role of different system parameters on users' performance and transport cost using synthetic traffic traces over different topologies. Simulations are run using an event-driven simulator implementing CCN link forwarding, packet-level caching and transport protocols [9]; we assume URI-like names and name-based shortest path routing protocols. We focus on steady state average system dynamics, neglecting the initial transient period required to fill up content stores.

3.1. Storage partitioning in a real traffic scenario

We now present an example of storage partitioning (SP) in a real traffic scenario. We use as input for our simulator a real traffic trace collected within a commercial network of a major European network operator. The trace is based on passive, packet-level observations collected over six hours at a central office (BRAS Broadband Remote Access Server) that aggregates the traffic of

Table 1
Hit probability (p_{hit}) at access nodes of the aggregation network topology for the measured traffic trace for Shared storage (SH) and Static Partitioning (SP). Content Store (CS) size is 1 GB.

	p_{hit} SH [%]	p_{hit} SP [%]
HTTP Web (A1)	43.0	47.6
Web media streaming (A2)	26.5	39.7
UGC Streaming (A3)	17.6	14.9
Unknown (A4)	12.5	11.8
All content items	17.7	17.1