May 24, 2016

Context-based Cache Filling from Peer Caches

Christopher John Scott Dougall

Context-based Cache Filling from Peer Caches

ABSTRACT

A network cache serves as a local repository of items such as video, audio, images, web pages, files, etc. within a network, e.g., a local area network (LAN). This disclosure describes techniques to fill a network cache with digital assets such as videos, web pages, data files, etc. Assets are selected for the cache based on popularity. One indicator of popularity is the recent demand for different assets. Another indicator of popularity is geographic location of the internet cache. Different assets are downloaded to a cache from different peer caches within a LAN.

KEYWORDS

- Internet cache
- Content delivery
- Peer-based cache filling
- Cold-start problem

BACKGROUND

Digital assets such as videos, images, data files, etc. are available over the Internet. Different assets have different popularity levels and experience different levels of demand. For example, if a video on a video sharing website goes viral, there is high demand from users. When a large number of users request an asset, such requests can result in repeated downloads of the same data over the Internet which can affect backhaul network performance. Caches are used to serve the digital asset locally, e.g., from within a LAN, to reduce load on the backhaul network.

As new or different content gains popularity, content of network caches needs to be updated. Also, as new caches are introduced into a LAN, such caches need to be configured to store digital assets. Disk replication technology that mirrors contents of one disk to another may not be suitable for cache filling for a variety of reasons. For example, replicating a disk might take a substantial amount of time, during which asset popularity changes and renders the replicated cache content less useful. Also, disk replication may not support prioritization of contents as a disk is replicated. Further, downloading items for the cache from a network may require substantial time, e.g., in days or weeks, during which the caches may not be able to provide digital assets locally.

DESCRIPTION

This disclosure describes an example environment with caches in a local area network that are coupled to an external network such as the Internet, and to a manifest server that is located outside the LAN. Further, techniques to fill the caches with popular assets using contents of peer caches are described.

Example environment

Fig. 1 shows an example environment in which techniques of the present disclosure can be implemented. A local-area network (LAN) 106 is coupled via backhaul 104 to the Internet 102. Within the LAN 106 are several network elements 108 that are connected by interconnections 110. Network elements 108 can be any device such as routers, switches, wireless base-stations, computers, phones, network attached storage, etc. that can be coupled to a network. As shown in Fig. 1, LAN 106 includes network caches 112a-d, coupled to other caches within the LAN 106 by interconnections 114. Manifest servers 116 are accessible from the LAN 106 via backhaul 104.

Manifest servers

Manifest servers 116 are coupled to LAN 106 via backhaul 104. A manifest server may include a processor, storage devices, and is coupled to the Internet via a network interface. The manifest server stores a manifest of recently demanded digital assets, along with a popularity metric. For example, the manifest can be implemented as a table 130 stored by the server, as shown in Fig. 3. Table 130 includes a unique identifier for each digital asset, universal resource locators (URL) corresponding to the asset, attributes such as size, most recent time of use, etc. of the asset, and a popularity metric.

The popularity metric is computed based on factors such as demand for the asset and a time, e.g., a timestamp, associated with the demand. In some examples, popularity metric may be specific to locations, e.g., a digital asset may have a particular popularity metric in New York, and a different value in Houston. In some implementations, the popularity metric can also be stored with reference to a type of locations, e.g., an educational video may be popular in schools in Cleveland and may have a high popularity metric with reference to schools in general, but may not be popular at airports and may have a low popularity metric with reference to airports. Popularity metrics are determined with user consent. For example, a certain asset is determined as popular at a location based on user requests from the location, when users consent to use of their request data to determine popularity. Request data from users who do

not provide consent is not used to determine the popularity metric. Further, request data may be anonymized, aggregated, or otherwise processed before use to compute the popularity metric, e.g., to ensure user privacy.

Network caches

Each of the caches 112 is a storage device that is coupled to a network. A cache can include storage elements such as RAM, hard disks, solid-state storage, tape storage, etc. Storage capacity of a cache may be relatively large, e.g., 4 terabytes, 8 terabytes, etc. Further, the cache can include a processor and a network interface. LAN 106 can include any number of caches. Caches can be added to or removed from LAN 106, e.g., by coupling or decoupling a cache device from the LAN 106.

In some implementations, a cache maintains a table that includes identifiers corresponding to each digital asset that is stored by the cache. In some examples, the table can also include a popularity metric corresponding to the respective asset, as described above with reference to the manifest server. In operation, the cache can respond to queries for stored items such as digital assets. For example, the cache can indicate whether it has stored a particular asset or can provide a requested asset.

Further, in some examples, the cache can be configured with location information. The location information may be set in a factory, e.g., by a manufacturer or seller of the cache, or may be configurable by a user, e.g., at insertion of the cache in a LAN. The location information can be provided in any form, e.g., as a latitude/longitude, as a zip code, as a street address, a city name, etc. The location information can also indicate a usage environment of the internet cache. For example, a geographical context for the cache can be school, hospital, coffee shop, train station, etc.

Operation

When a digital asset that is unavailable locally, e.g., within caches 112a-d, is requested by a user of LAN 106, the asset is downloaded over backhaul 104. As an asset experiences more requests, the popularity increases, and the popularity metric is revised based in part on the number of requests. Caches 112a-d store a local copy of popular assets. For example, the local copy is stored, e.g., based on a predicted use of the asset, before a request for the asset is received. In this manner, requests for the asset from users that access the asset via LAN 106 do not trigger Internet downloads over the backhaul. Instead, such assets may be served locally by any of the caches 112a-d.

Assets that are popular are cached locally by the caches 112a-d. Further, in some implementations, caches 112a-d store popular assets based on their location and/or geographic context.

For example, if the caches are located within the LAN of a school, then assets relating to education may be downloaded. If the caches are located within the LAN of a train station, then assets such as transportation timetables, maps, etc. may be preferentially downloaded to the caches.

One example use context is when a cache 112d is newly introduced into the network. The newly introduced cache initially contains no data, or data that is out of date. Such a newly introduced cache may not be useful until it acquires digital assets. This is referred to as cold-start problem. Thus, filling the cache with popular digital assets is performed upon introduction of the cache into the LAN. Filling the cache with popular assets can optimize a hit rate, e.g., a probability that a requested asset is available within the LAN.

Cache filling

Fig. 2 shows example communications exchanged among a requesting cache, a manifest server, peer caches, and external network, to fill the requesting cache. The requesting cache requests a manifest from the manifest server. In some implementations, the manifest server determines a location for the cache. For example, the location may be included in the request, or may be determined by the manifest server. The manifest server determines an appropriate manifest by selecting a portion of the table that includes items that are suitable for the cache. For example, the manifest server selects items that have a high popularity metric, items that are popular at the location of the cache, etc. The manifest server generates the manifest that includes the selected items and transmits it to the cache. In some examples, the the manifest server generates a manifest in response to each request, such that each generated manifest is specific to the requesting cache.

The requesting cache requests assets from peer caches in the LAN, e.g. caches 112a-c. In some implementations, the requesting cache requests items based on the popularity metric in the manifest, e.g., in decreasing order of popularity. Further, the requesting cache downloads different digital assets, or parts thereof, from different peer caches, which can have a load-balancing effect. In some examples, if certain assets identified in the manifest are unavailable from peer caches, the requesting cache downloads such items from an external network.

While Fig. 2 shows the manifest being provided by the manifest server, in some implementations, the requesting cache can request a local manifest maintained by one or more peer caches. In these implementations, the requesting cache requests the manifest from the one or more peer caches and determine assets to download based on the received manifest.

When two or more caches are introduced a LAN, each of the caches receive a manifest. The newly-introduced caches 112c and 112d download popular content from peer caches. In this example, each of the newly introduced caches can initially download different items from the peer caches, such that the newly introduced caches can provide downloaded assets to each other. This can reduce overheads and balance network load.

Examples of use

In one example, a school has a LAN with no caches. A new cache is introduced into the school LAN, e.g. to service demand for internet services. The cache is shipped to the school preconfigured with its geographical location/ context. Once the cache is connected to the LAN, it obtains the manifest from a manifest server, as described earlier. The cache then downloads the popular digital assets based on the obtained manifest. In this example, the manifest server determines based on the geographical location and context of the cache that content such as educational videos, audiobooks, etc. is suitable for caching in the school network. Accordingly, the manifest server provides a manifest that includes items that list such assets. Further, the manifest server excludes otherwise popular items, such as movies, that may not be suitable for the cache based on the location and context.

In another example, a hospital has a LAN with five caches. The hospital installs two additional caches. The new caches transmit a request to the manifest server. The manifest server provides a manifest for the caches which includes the most popular assets for the hospital context. For example, the most popular assets for caching in this context may be medical-informational videos, drug information, etc. In this example, each of the new caches gets a custom manifest that is different from that for the other new cache. The new caches download from the five caches, that are peer caches digital assets identified in their respective manifests. For example, the digital assets may be downloaded in a descending order of popularity so that the new caches can immediately begin to serve downloaded assets. Each of the newly introduced caches also download available assets from other newly introduced caches.

The techniques described in this disclosure enable filling a network cache in a local area network. Popular content is identified and retrieved from peer caches. By downloading assets from peer caches rather than from an external network, network caches may come online faster, e.g., start servicing requests for digital assets. Further, downloading from peer caches can also achieve a load balancing effect. Also, by prioritizing downloads of popular assets, the caches are more relevant to users within the LAN.

FIGURES

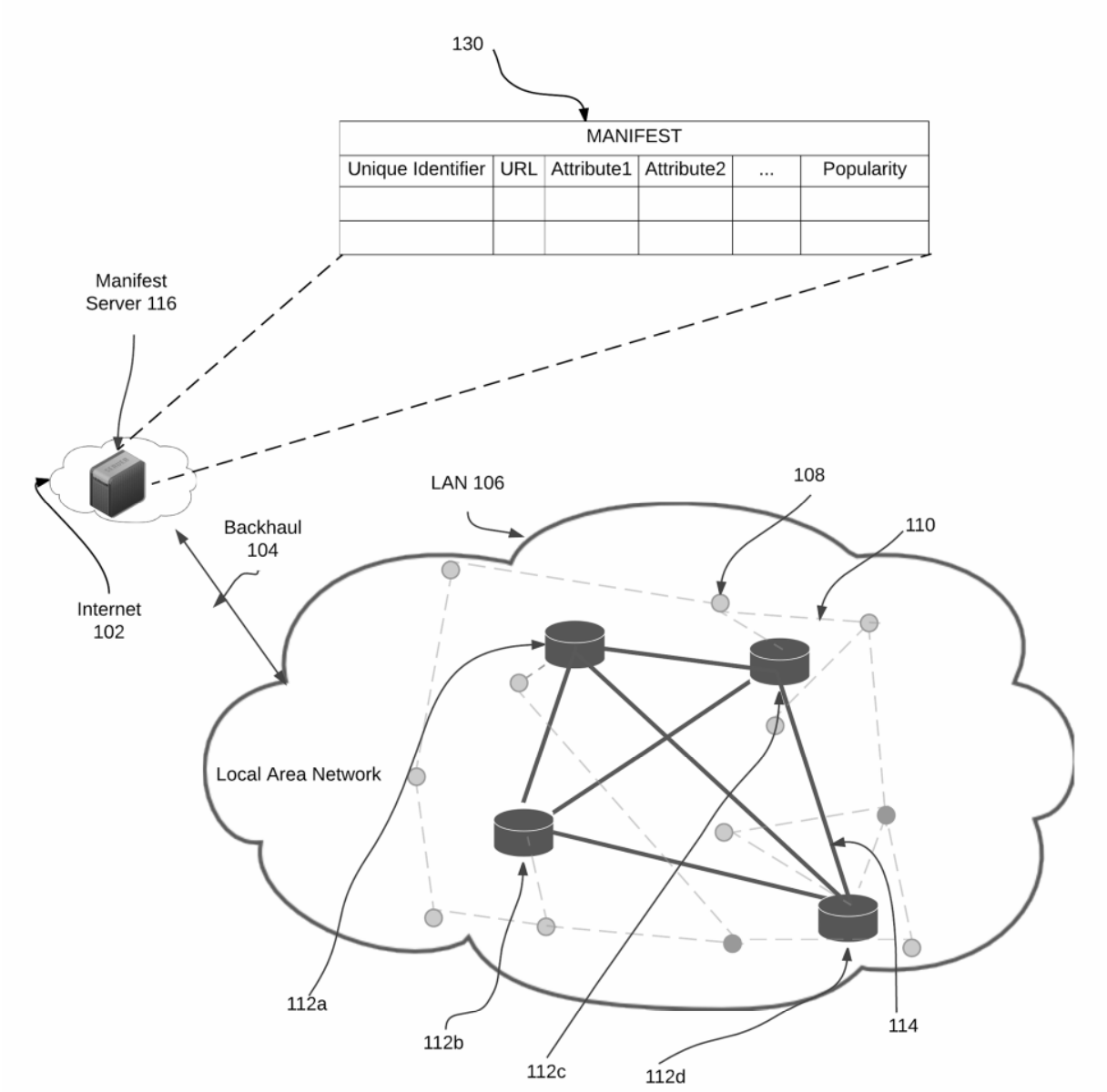

Fig. 1

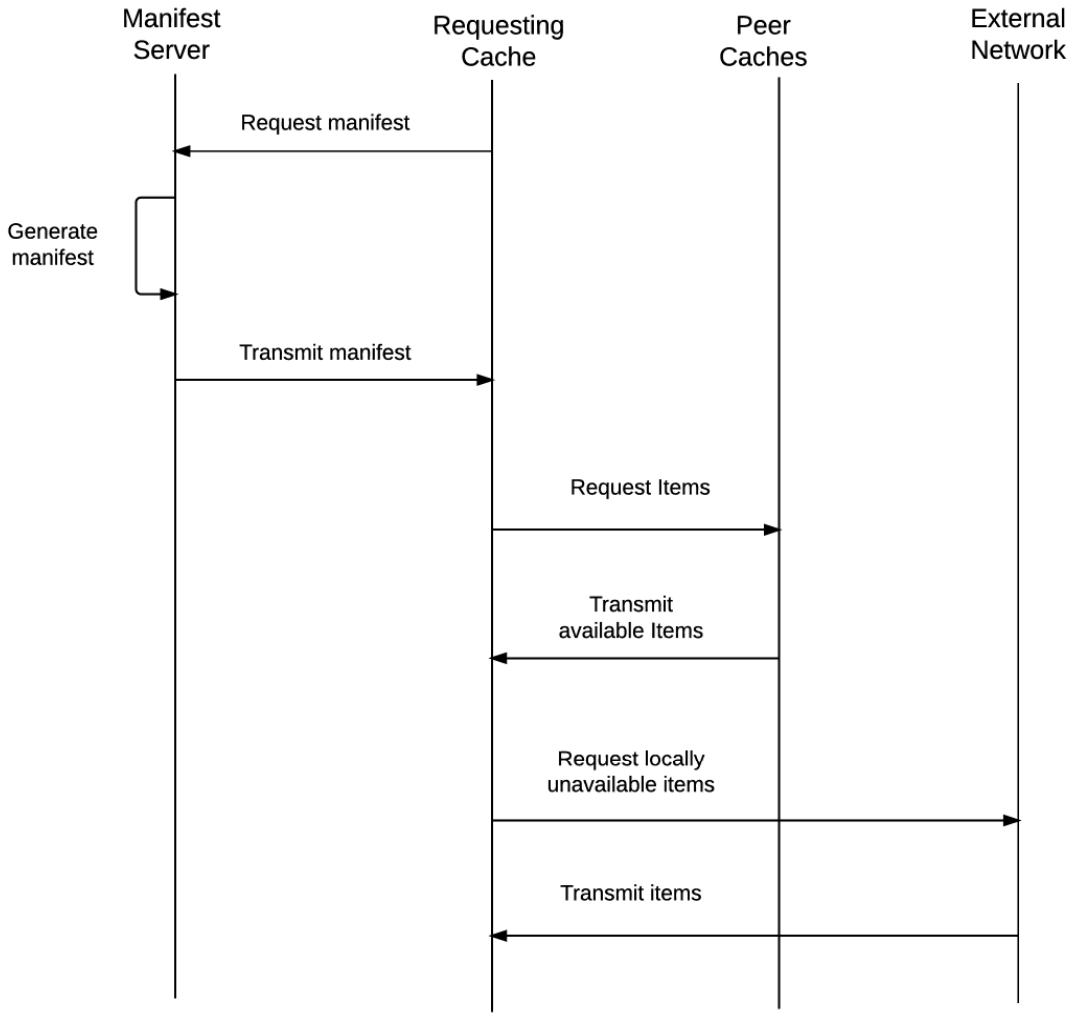
Manifest Server
Requesting Cache
Peer Caches
External Network
Request manifest
Generate manifest
Transmit manifest
Request Items
Transmit available Items
Request locally unavailable items
Transmit items

Fig. 2