Computer Networks 93 (2015) 518–530

Contents lists available at ScienceDirect

Computer Networks

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

Scaling persistent connections for cloud services

Wenjie Lin [a,*], Puneet Sharma [b], Sarbajit Chatterjee [c], Deepti Sharma [c], David Lee [b], Subu Iyer [d], Ajay Gupta [c]

[a] The Ohio State University, 395 Dreese Lab, 2015 Neil Ave, Columbus, OH 43210, USA
[b] Hewlett Packard Labs
[c] Hewlett Packard Enterprise
[d] HP Inc.

ARTICLE INFO

Article history:
Received 4 March 2015
Revised 17 August 2015
Accepted 6 October 2015
Available online 20 October 2015

Keywords:
Persistent connections
Scalability
Clouds
Internet connected devices
IoT

ABSTRACT

The unprecedented growth of cloud applications for connected devices is being hampered by scalability and associated costs for maintaining persistent connectivity between the cloud services and massive numbers of clients. In this paper we present the design and implementation of *Connection Parking*, a scalable and cost-effective cloud service delivery mechanism to millions of cloud-connected devices. In particular we focus on push delivery mechanisms where it is not possible for the cloud service to initiate connection establishment with the device. Our approach reduces the state resource requirements on the cloud servers without significant adverse impact on service delivery delay. We have prototyped and evaluated our solution with an operational stack of a cloud delivery service that has been designed, commercially deployed and is in production to support millions of connected devices. Our initial results demonstrate that Connection Parking can improve the scalability of cloud services with persistent connections by 3X under different operational scenarios. Our approach can be deployed in a transparent manner and does not require any modifications to either the cloud stack or the client device.

1. Introduction

Cloud applications have seen tremendous growth of online users and connected devices. From smart phones, tablets, laptops, PCs, and other devices, users are persistently connected with the cloud to receive a multitude of services. While these connections are idle most of the time, they require timely service delivery once a user-interested event occurs. How to manage this large number of persistent connections is a challenge for cloud applications. Besides the scalability, there are several other issues that make many current popular techniques inappropriate for cloud service delivery over persistent connections. In particular we focus on the scenarios where IoT devices are located behind NAT and firewalls and it is not possible for the cloud service to directly initiate connection establishment with the devices. The functional issues we consider in this paper include NAT/Firewall traversal, secure service delivery and transparent deployment. Additionally, there are performance requirements of low latency for timely service delivery while keeping the infrastructure costs low and proportional to the level of service activity.

This massive number of persistent connections between the cloud-connected devices and cloud service stacks during the service lifespan imposes a challenge to the current connection technologies. There are three main considerations that make this problem challenging. First is the scalability requirement to be able to match and support the growth to

* Corresponding author. Tel.: +1 6142705040.
E-mail addresses: lin.820@osu.edu (W. Lin), puneet.sharma@hpe.com (P. Sharma), sarbajit.chatterjee@hpe.com (S. Chatterjee), deeptis@hpe.com (D. Sharma), david.lee10@hpe.com (D. Lee), Subu.Iyer@hp.com (S. Iyer), ajay.gupta@hp.com (A. Gupta).

http://dx.doi.org/10.1016/j.comnet.2015.10.004
1389-1286/

such massive numbers of cloud-connected devices. Second, the connection between the devices and the cloud stack has to be persistent to allow for real-time status exchanges such as service delivery in a timely manner. Third is the requirement that the infrastructure costs grow in accordance with the number of *active* devices and not with the total number of connected devices. This is significant as most of the devices are often idle and their activity utilization can be as low as 1%. It is an unaffordable and unacceptable waste if a cloud server allocates connection resources for every connection including those with an extremely low utilization. While the service-level agreements (SLAs) for timely communication of activity/service-events warrant persistent connections, the low activity demands tearing down inactive connections immediately for cost-effective resource utilization.

While these three challenging requirements have existed individually in various scenarios, the cloud connectivity for IoT devices and other mobile devices requires all three of these challenges to be met simultaneously. It is not scalable to maintain such a massive number of persistent connections for this purpose due to the server resource requirements of maintaining buffers and state variables for each connection; we need a large number of servers to maintain the persistent connections and this cost is unacceptable. Further, due to the extremely low utilization of connections it is a big waste to maintain all the persistent connections during the complete service period. Consequently, commonly used persistent connection based application protocols, such as HTTP, FTP and Telnet, are not suitable. When the application devices have dynamic IP addresses or are behind NATs and firewalls, the connection establishment has to be initiated by the devices. Consequently, UDP-based application protocols do not work either. A new design approach is needed for such a massive number of persistent server-device connections where both scalability and low latency connectivity are key design requirements. In this paper we propose a novel persistent connection management scheme that can *park* millions of persistent connections to save server resource consumption by lightweight connection keep-alive management. Our novel approach called *Connection Parking* acts as a transparent proxy between the devices and cloud services to park inactive persistent connections, thus freeing up valuable resources on expensive servers and improving scalability manifold. It is co-located transparently in the path between the devices and server, just like other appliances such as load-balancer. After the initial connection setup, the parking proxy *parks* the connection by half-closing it from the server side. Thus the server can release precious connection management resources and be able to support a significantly larger number of devices. The job of maintaining the connection by sending keep-alive messages to the device is offloaded to the Connection Parking proxy, thus keeping the device side of connection half-open. We have implemented and evaluated our Connection Parking system with a production cloud service stack that is in operation with support for millions of cloud-connected devices.

This paper is organized as follows: The problem we address is described in Section 2. We then introduce our *Connection Parking* scheme in Section 3, with further detailed explanations of parking TCP connections and parking secure connections in Sections 4 and 5, respectively. Section 6 describes the implementation of Connection Parking proxy and its evaluation results are shown in Section 7. We discuss some other interesting issues and related work in Sections 8 and 9, and finally conclude the paper in Section 10.

2. Cloud service persistent connections: need and requirements

Before we describe our *Connection Parking* scheme, let's take a look at a motivating example of the need for massive persistent connections in cloud services and the operational characteristics of such persistent connections.

2.1. A motivating example

Envision a cloud service that has millions of users. It delivers user-interested content such as videos, images, software updates, advertisements, and even some control data to the users' Internet connected devices (e.g., laptops, smart phones, tablets, printers, and thermostats) as shown in Fig. 1. These devices are usually behind NATs, such that the cloud server cannot directly initiate a connection with them for service delivery. Hence a device initiated persistent TCP connection is usually maintained between each of the devices and the cloud server. Above the transport layer, the feature-rich server stack might also include SSL/TLS for securing the connection, XMPP for session management, web application server, and data storage.

As the number of users reaches the million scale, handling the large number of persistent connections between the server and the cloud connected devices becomes

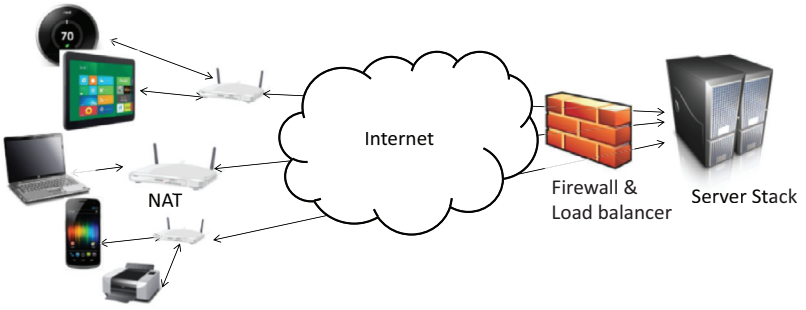

Fig. 1. An example of cloud service.