Building a distributed, fault-tolerant, offline web

Arthur Marques Felix Grund Paul Cernek

October 6, 2017

Abstract

paper instructions: https://www.cs.ubc.ca/ bestchai/teaching/cs527_2017w1/proposals.html

1 Introduction

2 Background

2.1 Flyweb

FlyWeb[1] is a web API developed by the Mozilla Firefox community[2] which enables web applications to connect and communicate to each other over a local-area network. To do so, FlyWeb relies on a zero-configuration network and its mDNS/DNS-SD protocols [2, 1]. With FlyWeb, a web page may be seen as a device, which hosts a server. This device advertises itself to other client devices in the local network and, eventually, these other clients will discover and connect to the advertised server. Thus, connected clients/servers enable cross-device communication.

2.2 Zero-configuration Networks

Zero-configuration networking is a combination of protocols that aims to automatically discover computers or peripherals in a network without any central servers or human administration. To do so, Zero-configuration networks have two major components that provide *(i)* the automatic assignment of IP addresses and host naming (mDNS), and *(ii)* service discovery (DNS-SD).

Roughly, when a device enters the local network, he assigns and IP/name to himself and then multicast that to the local network, resolving any name conflict that may occur in such process. IP assignment considers the link-local domain address, which draws addresses from the IPv4 169.254/16 prefix and, once an IP is selected, a host name with the suffix ".local" is mapped to that

[1]https://flyweb.github.io/spec/
[2]https://wiki.mozilla.org/FlyWeb

IP [2]. As devices are mapped to IPs/host names, their available services are discovered using a a combination of DNS PTR, SRV, and TXT records [1], thus their services can be requested by other devices.

2.3 Replication

(work in progress...)

Data replication is one of the major design approaches to achieve reliability and fault-tolerance in distributed systems: information is shared on redundant replicas such that any replica can become the new master if the current master replica fails. While enabling the system artifacts of fault-tolerance, reliability and availability, replication comes at a cost of performance: depending on the required operations in the system for replication, system performance can suffer significant bottlenecks. Different models of replication have been proposed to trade consistency for performance which resulted in different levels of consistency as a design choice for the target system.

In order to support replication in the client-server paradigm present in network applications, the State Machine Replication model was proposed in the 1980s in [3] and later refined in [4]. It is based on the concept of distributed consensus in regard to reliably reaching a stable state of the system in the presence of failures. [3] introduced the strategy of *active* replication (also called *primary-backup* or *master-slave* scheme) where requests to the master replica are processed to all other replicas. Given the same initial state and request sequence, all replicas will produce the same response sequence and reach the same final state.

In the passive scheme (also called multi-primary or multi-master scheme), requests are first processed on the master replica and the resulting state is distributed to other replicas.

3 Proposed Approach

4 Evaluation

5 Timeline

References

[1] S. Cheshire and M. Krochmal. Dns-based service discovery. RFC 6763, RFC Editor, February 2013.

[2] S. Cheshire and M. Krochmal. Multicast dns. RFC 6762, RFC Editor, February 2013.

[3] L. Lamport. Using time instead of timeout for fault-tolerant distributed systems. *ACM Transactions on Programming Languages and Systems*, 6/2:254–280, April 1984.

[4] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, Dec. 1990.