# Spring Cloud Ribbon

Understanding and Using Ribbon,
The client side load balancer

# Objectives

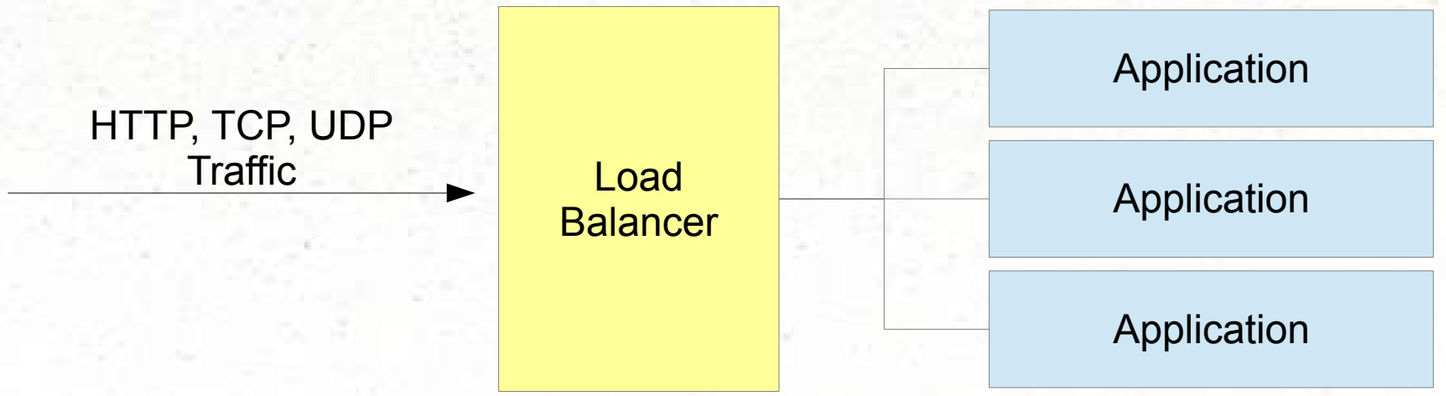
- At the end of this module, you will be able to
  - Understand the purpose of Client-Side Load Balancing
  - Use Spring Cloud Ribbon to implement Client-Side Load Balancing

# Module Outline

- **Client Side Load Balancing**
- Spring Cloud Netflix Ribbon

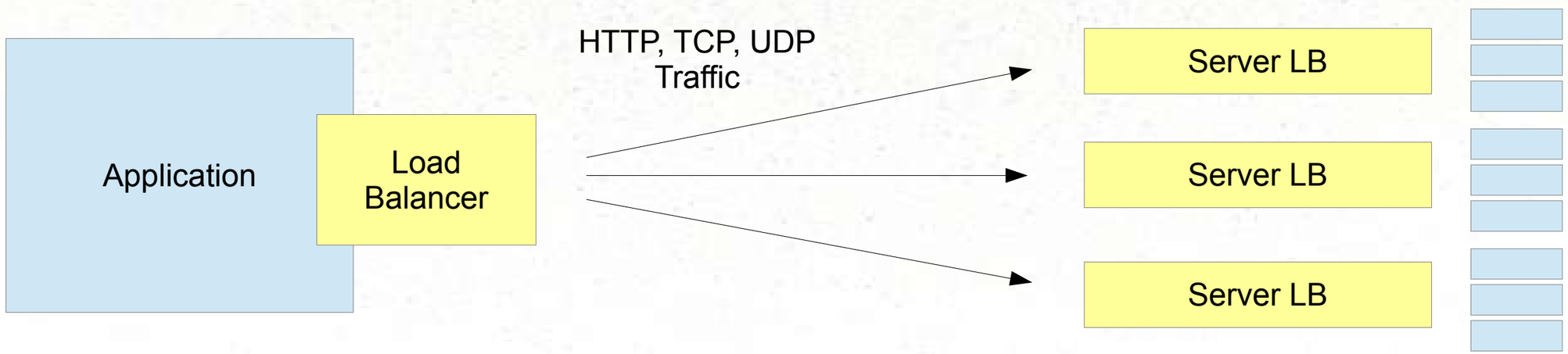
# What is a Load Balancer?

- Traditional load balancers are server-side components
  - Distribute incoming traffic among several servers
  - Software (Apache, Nginx, HA Proxy) or Hardware (F5, NSX, BigIP)



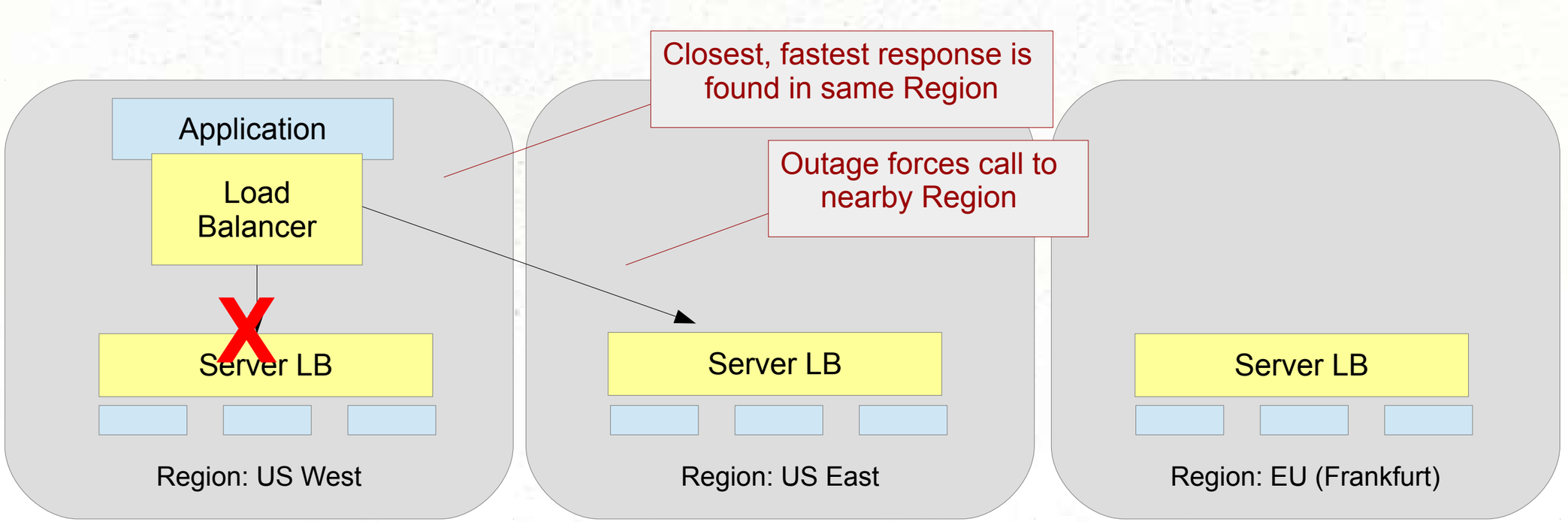
# Client-Side Load Balancer

- Client-Side Load Balancer selects which server to call
  - Based on some criteria
  - Part of client software
  - Server can still employ its own load balancer



# Why?

- Not all servers are the same
  - Some may be unavailable (faults)
  - Some may be slower than others (performance)
  - Some may be further away than others (regions)



# Module Outline

- Client Side Load Balancing
- **Spring Cloud Netflix Ribbon**

# Spring Cloud Netflix Ribbon

- Ribbon – Another part of the Netflix OSS family
  - Client side load balancer
  - Automatically integrates with service discovery (Eureka)
  - Built in failure resiliency (Hystrix)
  - Caching / Batching
  - Multiple protocols (HTTP, TCP, UDP)
- Spring Cloud provides an easy API Wrapper for using Ribbon.

# Key Ribbon Concepts

- List of Servers
- Filtered List of Servers
- Load Balancer
- Ping

# List of Servers

- Determines what the list of possible servers are (for a given service (client))
  - Static – Populated via configuration
  - Dynamic – Populated via Service Discovery (Eureka)
- Spring Cloud default – Use Eureka when present on the classpath.

Example of "Static" server lists

"stores" and "products" - Examples of client-ids

application.yml

```
stores:
  ribbon:
    listOfServers: store1.com,store2.com
products:
  ribbon:
    listOfServers: productServer1.com, productServer2.com
```

# Filtered List of Servers

- Criteria by which you wish to limit the total list
- Spring Cloud default – Filter servers in the same *zone*

# Ping

- Used to test if the server is up or down
- Spring Cloud default – delegate to Eureka to determine if server is up or down

# Load Balancer

- The Load Balancer is the actual component that routes the calls to the servers in the filtered list
- Several strategies available, but they usually defer to a Rule component to make the actual decisions
- Spring Cloud's Default: ZoneAwareLoadBalancer

# Rule

- The Rule is the single module of intelligence that makes the decisions on whether to call or not.
- Spring Cloud's Default: ZoneAvoidanceRule

# Using Ribbon with Spring Cloud – part 1

- Use the Spring Cloud Starter parent as a Parent POM:

```
<parent>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-parent</artifactId>
   <version>Angel.SR4</version>
</parent>
```

- ...OR use a Dependency management section:

```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-starter-parent</artifactId>
         <version>Angel.SR4</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

...exactly the same options
as a spring cloud config client
or a spring cloud eureka client.

# Using Ribbon with Spring Cloud – part 2

- Include dependency:

```
<dependencies>
   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-ribbon</artifactId>
   </dependency>
</dependencies>
```

# Using Ribbon with Spring Cloud – part 3

- Low-level technique:
  - Access LoadBalancer, use directly:

```
public class MyClass {
    @Autowired LoadBalancerClient loadBalancer;

    public void doStuff() {
        ServiceInstance instance = loadBalancer.choose("subject");
        URI subjectUri = URI.create(String.format("http://%s:%s",
            instance.getHost(), instance.getPort()));
        // ... do something with the URI
    }
}
```

"subject" - An example of a "client-id"

Instance selected by the load balancer

# RestTemplate as Ribbon Client

- Spring Cloud automatically provides a RestTemplate
  - Assuming LoadBalancerClient Bean is present, and RestTemplate on classpath.
- To use:
  - @Autowire the RestTemplate, qualify with @LoadBalanced if multiple
  - Specify service to call using syntax “http://<service-name>

```
public class MyClass {
    @Autowired@LoadBalanced RestTemplate restTemplate;

    public void doStuff() {
        String subject = restTemplate.getForObject("http://subject");
    }
}
```

Optional if there is only one RestTemplate in the context

“subject” - example of service name.
“http://” - syntactical requirement only

Load-balanced RestTemplate automatically uses Ribbon to select client.

# API Reference

- Previous example used Ribbon API directly
  - Not desirable – couples code to Ribbon
- Upcoming examples will show declarative approach
  - Feign, Hystrix.

# Customizing

- Previously we described the defaults.  What if you want to change them?
- Declare a separate config with replacement bean.

```
@Configuration
@RibbonClient(name="subject", configuration=SubjectConfig.class)
public class MainConfig {
}
```

```
@Configuration
public class SubjectConfig {
    @Bean
    public IPing ribbonPing(IClientConfig config) {
        return new PingUrl();
    }
}
```

Replaces the “Ping” strategy
Employed when calling “subject” clients

Note: Do NOT component scan SubjectConfig!

# What Customizing Choices are available

- Quite a Few!
  - Recommend looking at the JavaDoc or GitHub Code
    - http://netflix.github.io/ribbon/ribbon-core-javadoc/index.html?com/netflix/loadbalancer/package-summary.html

# Summary

- Client-Side Load Balancing augments regular load balancing by allowing the client to select a server based on some criteria.
- Spring Cloud Ribbon is an easy-to-use implementation of client side load balancing.

# Exercise – Using Ribbon

Using Ribbon Clients

Instructions: Student Files, Lab 5