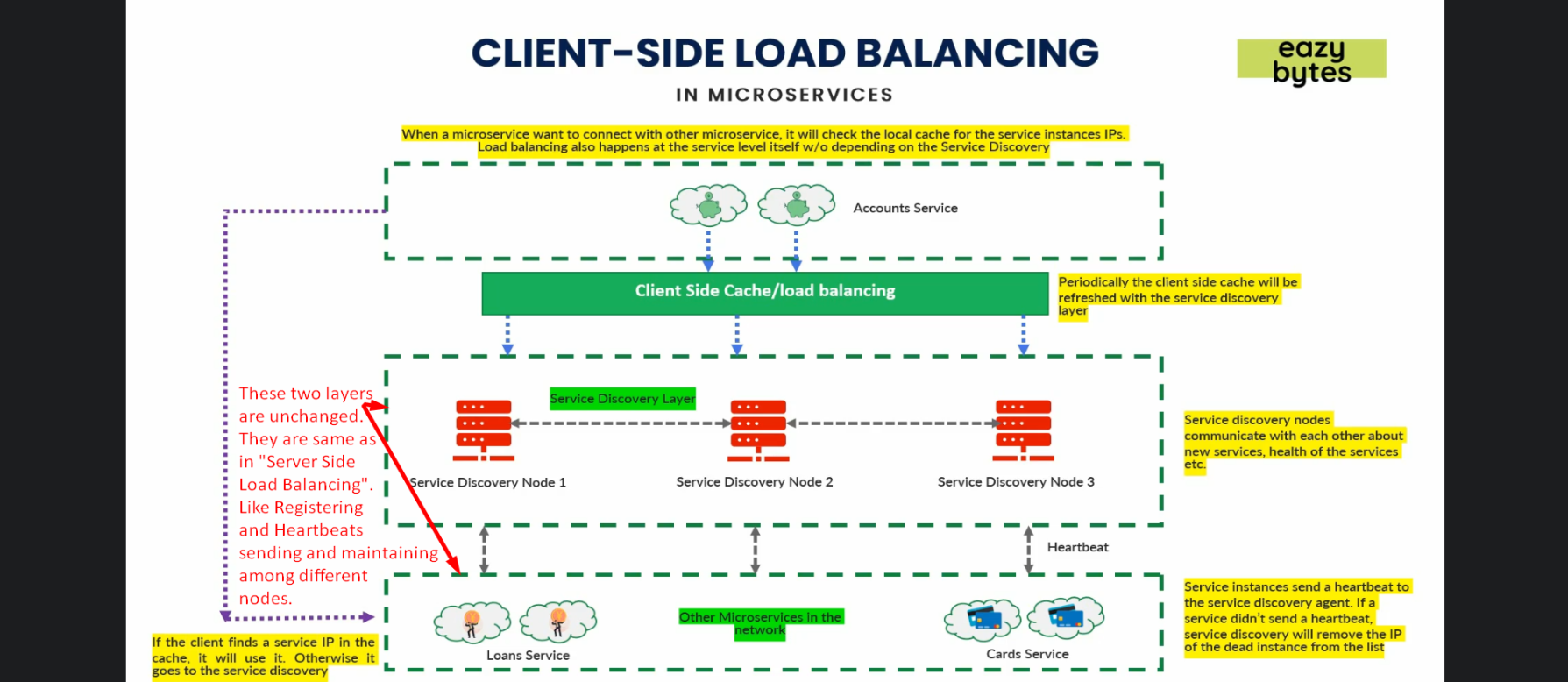
1. By default we were using Server side load balancing but we can improve this architecture further by using client side load balancing.
2. **Agenda**:
   1. How client side load balancing works.
3. 
   1. In this architecture, there is no difference in how
      1. A microservice gets itself registered with “Service Discovery Layer”.
      2. How nodes in this layer maintain the heartbeats from microservices.
   2. The only difference is that we write some logic b/w the client app and service discovery layer that caches the mappings (logical names to real IPs).
4. Let’s discuss the “Client Side Cache/Load Balancing”.
   1. Let’s suppose that the very first time, accounts service very first time is trying to connect with loans service.
   2. As we know it doesn’t know the direct URL of loan service, so it will reach out to the service discovery layer.
   3. Service Discovery Layer will give all the following details
      1. How many instances of loans microservice.
      2. Their direct IPs.
   4. Client side load balancer will cache all this info so that in future it doesn’t need to reach out to Service Discovery Layer.
   5. **Advantages**:
      1. Now “**Service Discovery Layer**” doesn’t have a load of requests.
      2. No need for so many nodes/agents inside “**Service Discovery Layer**”.
      3. Now, account microservice doesn’t have to make calls to “**Service Discovery Layer**”.
   6. **Question**: What if an instance whose details cached is deleted/ killed later on?
      1. Client side load balancer does two things.
         1. 1st: Client side load balancer will periodically make request to the microservice to refresh the cached details.
         2. 2nd: During the interval (timeframe), when we make request and get an exception, it will not throw exception immediately rather it assuming that cached details needs to be refreshed, connects with “Service Discovery Layer” and refreshes the cache.