Last edited by **graysonc** 1 day ago

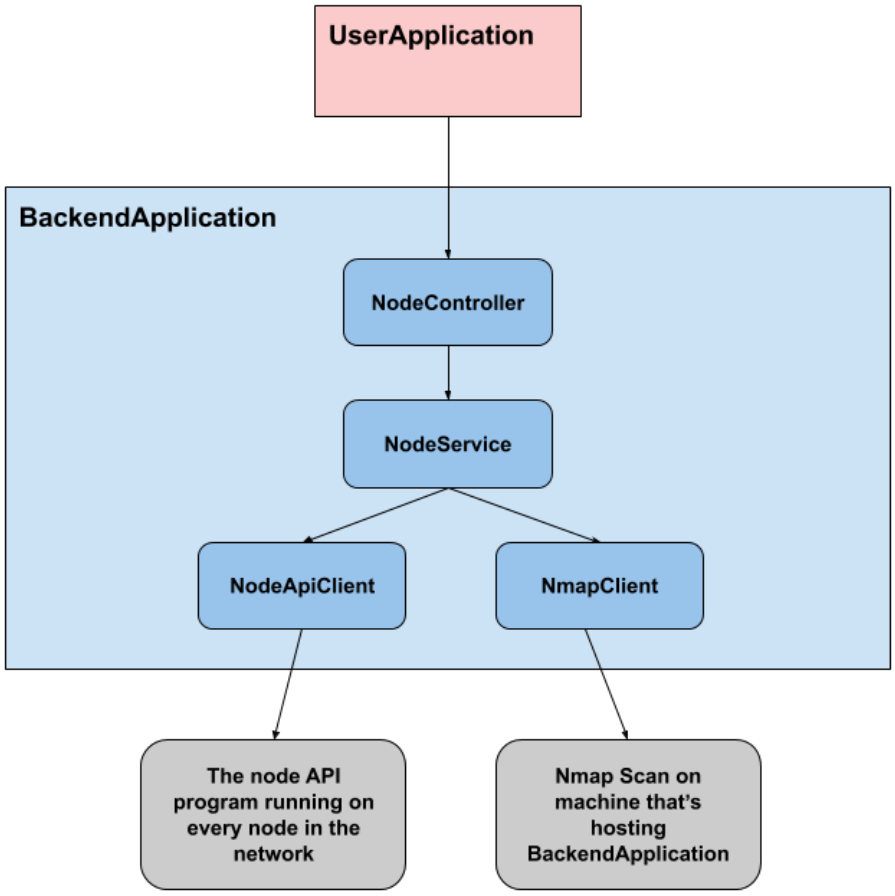
Backend Design

Overview

This application serves as a facade between the User Application and the mesh network. It handles all accessing and mutation of information among the network and hides the complexity of these operations behind a simple REST API, on which the User Application can call.

The only exception to this description is with video streaming, in which case the User Application communicates directly with the Camera Node. All other operations go through this Backend Application.

Component Breakdown



Package Structure & Source Code Breakdown

- BackendApplication
  - network_manager
    - config
      - NetworkManagerConfiguration

NetworkManagerConfiguration

This is the configuration class for the `network_manager` module. It provides all the Java beans for dependency injection. If you add a new component (e.g., a controller or service) to this application, then you should add a `@Bean` method that provides an instance of that component.

NodeController

`NodeController` defines the REST API request mappings to be used by the User Application. It receives HTTP requests and then calls on `NodeService` to perform the desired operation.

This component consists of two files: `NodeController.java` and `NodeControllerImpl.java`. This seemed like a good design pattern when I made these classes, but it's not necessary for the interface to be separate from the implementation. I regret doing it like this.

NmapClientException

This is a generic exception that is thrown by only `NmapClient`. I added these exceptions so that each of the services would have its own unique exception.

NodeApiClientException

This is a generic exception that is thrown by only `NodeAPIClient`.

NodeServiceException

This is a generic exception that is thrown by only `NodeService`.

NodeModel

This is just the node data model. It models the properties of a network node, which are configured by modifying `NodeCommon/network_settings.cfg`. The primary key is `ipAddress`.

NodeType

This is just an `Enum` representing the different types of nodes. These types are `MASTER`, `RELAY`, and `CAMERA`.

NmapClient

This component encapsulates the usage of `nmap`. Its only purpose is to perform an `nmap` scan of the mesh network and parse the results to extract the IPv4 addresses.

NodeApiClient

This component handles all interactions with the Node API (the Python REST service run on every node in the mesh network). It has a set of methods that mirrors the methods defined in the Node API. When the Backend Application wishes to call a Node API method on a node in the mesh network, it will call the corresponding method in `NodeApiClient`, which will then make the HTTP request to the node directly.

NodeService

After `NodeController` receives a request, it calls on `NodeService` to carry out the desired operation by delegating the work to `NmapClient` and `NodeApiClient`. These operations are `getAllNodes`, `getNodeByIpAddress`, and `updateNode`.

BackendApplication.java

This is the program entry point for the application.