Design Documentation

Distributed Global State Monitoring

Andrew Azores
Evan Holtrop
Jazz Kersell
Darren Kitamura

1. Introduction
    1. Purpose
        - The Purpose of this project is to provide a proof of concept for a paper provided by the supervising professor. We will also be implementing the algorithm over a distributed network.
    2. Description
        - The Purpose of this project is to provide a proof of concept for a paper provided by the supervising professor. We will also be implementing the algorithm over a distributed network.
    3. Scope
        - The scope of this project is to create an easy to use interface that will allow out professor to expand. The end product should allow our professor to develop other applications, and give him a better basis to perform demos. We will also provide him with the working Drone demo that is explained above. To do this we will need to implement his Monitoring algorithm, as well as develop a distributed network among devices to send the data required by the algorithm.
2. Design Principles
    1. Distributed Network
        - Distributed networks is the idea that a network exists between devices with no server. This way devices communicate directly with each other instead of communicating with a server.
        - The benefits of a distributed network mostly have to do with points of failure, in a distributed network is a user drops then the network still exists. While in a Centralized network if the server disconnects all users are disconnected from each other.
    2. Threads
        - Everything that can be put to a separate thread has been done so. Our entire application runs asynchronously from itself, from the Service to the Server each module runs on its own.
    3. Modularity
        - Each module can run on its own without the need for another. The monitor module never defines the type of any objects that the user wishes to send across. Same with the networking module. The application layer is completely unaware of what happens on the top layer.
3. Development Details
    1. Language of Implementation
        - Due to our end goal being a demo of phones flying drones we were heavily limited to Java/Swift for our development languages.
        - Our reasoning for using Swift over Objective-C is the increased flexibility Swift offers in combination with the ability to avoid long complicated work-around techniques to create Objective-C equivalent programs of the Java counterparts. Apple has also advertised at their developer's conference that Swift can perform up to 40% faster on similar tasks compared to using Objective-C as a push to create a modern language for their mobile devices. With the above in mind it made complete sense to create this project in Swift
        -
    2. Supported Frameworks/API
        - Lombok

- Volley
- NanoHTTP
- NSD
- Gson

3. Time line of Critical Components

Component	Date
Global Monitor	January 19th 2015
Distributed Network	January 19th 2015
Drone Pilot	??
IOS Implementation	January 19th 2015

4. Component Overview
   1. Network Module
      1. Device Info
         - Used to serialize device information. Data tracked is ip, port, and a DeviceLocation structure.
         - This structure is used as a container to send data across the network
         - Function: toString - converts itself to Json.
      2. Device Location
         - Used to keep track of location data of a given device, data tracked is latitude, longitude, altitude, barometer pressure, speed, bearing, GPS accuracy, gravity and linear acceleration.
         - This structure is used as a container to send data across the network
      3. NetworkPeerIdentifier
         - Used to keep track of all other peers on the network. Each device contains a list of these so that they know of all peers.
      4. Payload Object
         - A wrapper for the object you wish to send over the network. Other meta data is kept here for communication purposes.
         - Upon testing we found that the capstoneService can filter data as needed. We will need to test with a large number of devices to find if this is truly redundant.
      5. Capstone Service
         - a service that manages NSD registration. It also keeps track of all device information requires (barometer, gravity, ip, port) it also manages receiving tokens, sending Events to the global monitor, requesting data from peers.
         - Most calls in this Service provide functionality to (un)package, send, and receive data to and from both local and remote recipients, either other CapstoneService instances or Monitor instances. Functionality to retrieve and share data from local sensors is also exposed.
      6. Capstone Server
         - This class manages all communication over the distributed network. In our case the

network will be distributed over NSD (Android) and Bonjour (IOS).

2. Monitor Module
    1. Event
        - An event is what is sent to the monitor to register that something has changed in the underlying application. An event consists of an Event Type (Send, Receive, Internal), a Network Peer Identifier, a valuation, and a vector clock.
    2. Token
        - this class is used to represent the computation slicing tokens
    3. Vector Clock
        - Represents the logical time of each process. The logical time of a process is represented by an integer which counts the events that occur in that process.
        - Function: Compare – compares two vector clocks and returns the required comparison depending on the outcome. Equal if both clicks are the same, Bigger if this has a more recent event, Smaller if clock has a more recent event, Concurrent if each has some more recent event
        - Function: Merge – Merge two vector clocks so that the merged clock contains the most recent occurring events in each process represented by the vector clocks. This returns a merged click such that the new clock contains the most recent events. Note: will throw an exception if clocks are of different size
    4. Valuation
        - The values of the variables being monitored in the process.
    5. Process State
        - Represents the state of the process. The state consists of a Valuation and a Vector Clock.
    6. Automation
        - The automaton is a finite state machine which represents the LTL3 property which is being monitored. It consists of:
            - Automaton States
            - Automaton Transitions
        - It is important to note that the automaton does not track it's current state. This is done by the GlobalView since depending on what information is known about the global state, the automaton may need to be in a different state.
    7. Automation State
        - represents a state that the automaton can enter
        - type represents whether the state is an accepting, rejecting, or undecided state
    8. Automation Transition
        - Moves the automaton from one transition to another state.
        - Has a list of conjuncts which represent the predicate that labels the transition. There are only conjunctive predicates. Any disjunctive predicates are delt with by splitting them into multiple transitions.
    9. Conjunct
        - Represents a Boolean expression that is used to evaluate the predicate labelling a transition.
        - The expression is represented by a Boolean Expression Tree
    10. Boolean Expression Tree
        - Represents simple Boolean expressions (IE, x > 2, y == 5) and uses Valuations to evaluate these expressions and return the result.

        11. Global View
            - Tracks what is known about the global state of the system, and uses tokens to retrieve information from other processes to update it's knowledge of the global state.

5. System Architecture
    - Application -> Distributed Network -> Global Monitoring Algorithm
    - When an application runs into a situation where it needs information from another device it sends a message to the Global Monitor. The global monitor will find the information necessary from the device needed and update the global state with the new information.
    - The distributed network is a layer in between the Application and the Global Monitor. The network is a link layer between the two that manages all IP/port connections.
    - In our case the application layer will be the drone pilot program. Due to the fact that we do not have drones design of this module has been postponed.
6. Communication Protocol
    - Communication is handled by using Json. All data is serialized into Structures described above. The structure then be converted into Json and are then sent over the network. The receiving device unpacks the Json and interprets the data as needed.
7. Monitor Algorithm Overview
    - The monitor algorithm verifies that specified properties hold throughout the system. It receives events from the process it is monitoring and records the information from that event. When information from another process is needed the monitor sends a token to that process to request the information. The receiving monitor then processes the token, adding to it the requested information and sends it back to where it originated. This mechanism propagates information about the state of the system throughout the processes, and allows them to collectively determine whether the systems satisfies a given property.
8. Future Changes
    1. Drone Implementation
        - the Android and IOS application will be changed to support USB connection with the drones provided by the professor.
        - The current due date of the drone pilot module is unknown, we currently do not have the drones so development has not started.