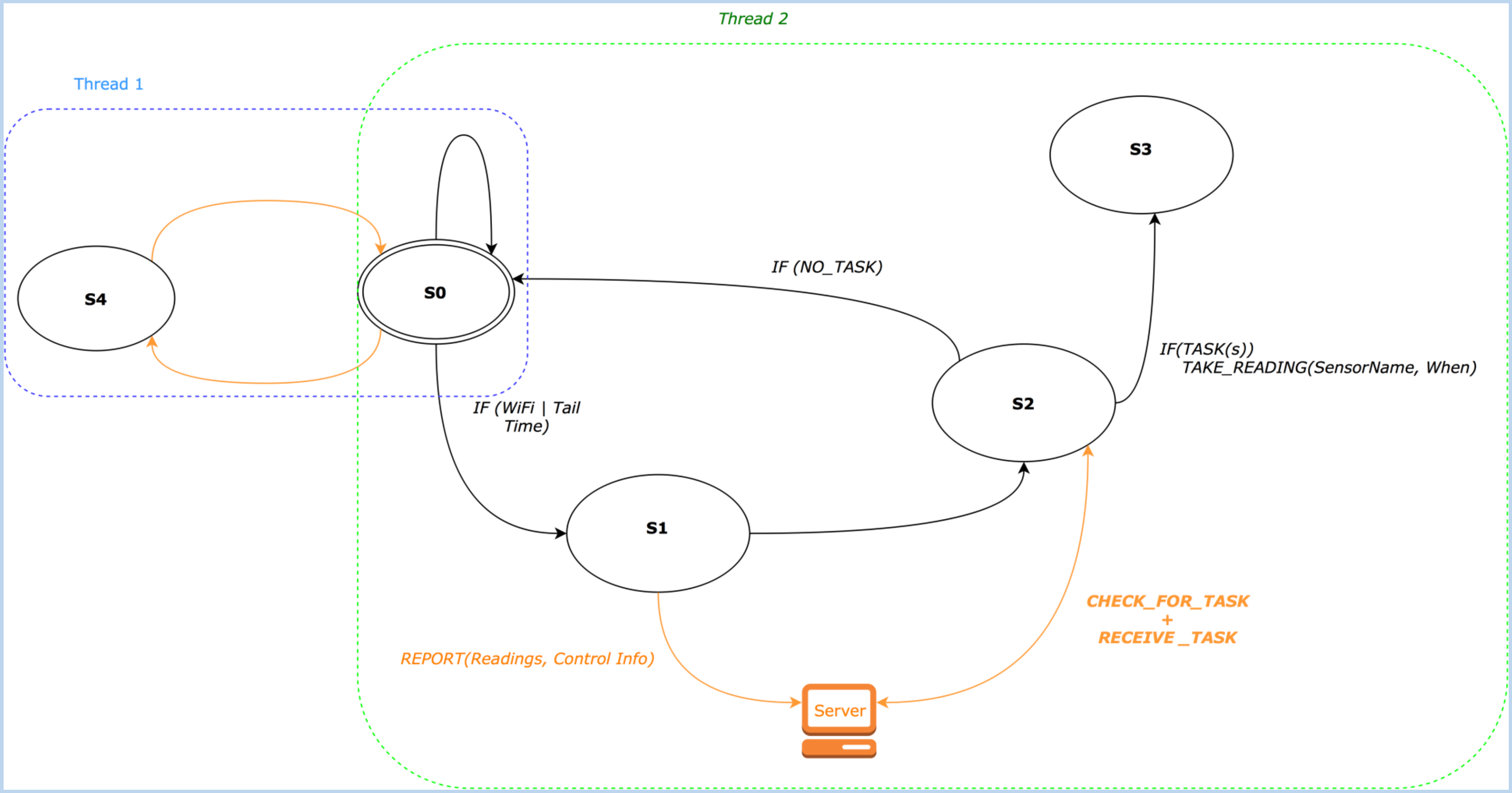
Crowd Sensing Client

The Client is responsible for taking sensor readings on the device and sending the recorded readings to the crowd sensing server. The state-diagram representation of the client program looks like:


Thread 2
Thread 1
S3
S4
S0
IF (NO_TASK)
IF(TASK(s)) TAKE_READING(SensorName, When)
S2
IF (WiFi | Tail Time)
S1
CHECK_FOR_TASK + RECEIVE_TASK
REPORT(Readings, Control Info)
Server

S0

Dormant state. Application is silently running in the background as a service. (Periodically transitioning to S4)

S1

Reporting state. S0 -> S1 transition occurs when the device is connected to Wi-Fi or during radio tail time. Application checks local log files and sends data over to the server in form of JSON strings. Local log files are flushed after upload completes.

S2

S1 -> S2 transition is automatic. In S2 the program waits for the server to send a list of readings to take. (task list is in the form of JSON strings). The list of tasks received tasks can be represented as:

```
      {  [Barometer, 7:00 pm, 1 hr] , […], […]  }
```

The task list is in the form

```
      {  [<Sensor Name>, <Reading deadline>, <deadline threshold>], […], […]  }
```

The deadline threshold is the time period within which the sensor reading can be taken depending on sensor availability. E.g. A 7:00 pm deadline with a 1hr threshold expects the sensor reading to be taken between 6pm to 7pm.

S3

If a list of tasks is received in S2, the program begins to take readings for each task in S3. Otherwise, S3 is skipped and a S2->S0 transition takes place.

Temporarily, a reading is taken as soon as the task is received from the server. *Ideally*, readings would be taken in a piggyback fashion depending on device usage. If the required sensor was not used before the deadline and within the threshold, then, the sensor would be invoked by our application to meet the deadline. All other times, we expect the sensor to be invoked by the user and our application utilizes this reading instead of invoking a sensor by itself.

S4

S4 is an isolated state from all states except S0. As the application silently runs in the background in S0, periodically, a method will be invoked in a separate thread to log control info. (Includes log of sensor activity throughout day/week/month). S4 continually appends to (or creates) the control info log file. (Note: All log files are flushed in S1 once the reporting is complete).

Pseudocode

```
S0
{
     if( isTailTime || activeWiFi)
          S1();
     Thread t = new Thread( {
                              While(1)
                                   Thread.sleep(ONE_HOUR);
                                   S4();
                         })
     t.start();
}


S1
{
     File ControlLog, ReadingLog;
     Merge_To_JSON(ControlLog, ReadingLog) -> JSON_string;
     Send(ServerSocket, JSON_string);
     Bool PendingTask <- read(ServerSocket);
     If(PendingTask)
          S2();
     Else
          S0();
}
```

```
S2
{
      Read(serverSocket) -> task_list_str;
      JSON_to_List(task_list_str) -> TaskList;

      For( task : TaskList )
            S3( task.SensorName, task.Deadline, task.Threshold);
}

S3 (SensorName, Deadline, Threshold)
{
      If(TimeNow < Deadline || TimeNow within Threshold)
            TakeReading(SensorName) -> Reading;
            LogToFile(ReadingLog, Reading);
            Return;
      // Reading was not reported to server.
}

S4
{
      LogToFile(ControlLog, Current_battery);
      LogToFile(ControlLog, IMEI);
      LogToFile(ControlLog, SignalStrength);

      For( sensor : ALL_SENSORS)
            AppendToLog(sensor.activeUsage)
}
```