Satellite Communication

A commercial ground station provider wants to sell time slots for satellite communication to space agencies. To specify the prices, the provider wants to identify scheduling hot spots during a day. For this purpose the flight dynamics team has provided a log file containing the time intervals satellites are above the horizon and in principle visible to the ground station. Using this log, find the time range(s) when the maximum number of satellites are visible to the ground station and how many satellites there are during that time range. Usually flight dynamics use the standard "CCSDS ASCII Time Code A" format [1] to represent time stamps. However, for this task they provided time stamps in milliseconds precision and without date. The start and end times of communication periods are inclusive. For instance, if one satellite started communication with a ground station at 12:34:56.789 and another ended its communication at 12:34:56.789, there were 2 satellites visible to the ground station at 12:34:56.789.

Exercise

1) Implement an application that:
    a. Reads the start and end times of satellite visibilities from an input file. The location of the input file must be given as a command line parameter to the application. Each line of the input file contains the start and the end time of the visibility period of exactly one satellite, separated by comma.
    For example:
    12:34:56.789,15:43:21.012

    The log file may not be sorted by start or end time. For testing, the provided input file called "satellites.dat" can be used.
    b. Based on the data from the input file, find the time range(s) when the maximum number of satellites were visible to the ground station and how many satellites there were.
    c. Output the time range(s) and number of satellites found to standard output in the following format: <start time>-<end time>;<number of satellites>.
    For instance:
    12:34:59.001-12:36:42.422;7
2) Document your solution and your major design decisions. The documentation should be in a separate file and it should contain a description of your algorithm as well as the reasons for solving the problem in the way that you did. Furthermore, document any foreseen limitations of your solution. The document could also contain your evaluation of other possible solutions and the reasons why you rejected them over your chosen approach. Keep in mind that the documentation could be in principle handed to the client along with your application.

Further Information:

The solution will be evaluated as though it was part of a production system and will be used to judge your technical capabilities based on your level of experience.

The solution will be judged on the general quality of the code and documentation. In particular it will be judged on whether it produces the correct answer, the algorithm efficiency, appropriate use of standard libraries/APIs, readability and maintainability. It will also be evaluated based on its scalability for possible future extensions such as times being supplied with a much higher precision or over a greater time range.

Unless instructed otherwise, the following languages would be preferred for the solution: Java, C++, C#, Javascript, Python. If the solution is supplied in C++, please ensure that the code is ANSI compliant as in general, solutions are evaluated on a Linux platform.

References

[1] *Time Code Formats*. Blue Book, Issue 4. November 2010. https://public.ccsds.org/Pubs/301x0b4e1.pdf