New Relic Code Challenge

Using any programming language (taking performance into consideration), write a server ("Application") that opens a socket and restricts input to at most 5 concurrent clients. Clients will connect to the Application and write one or more numbers of 9 digit numbers, each number followed by a server-native newline sequence, and then close the connection. The Application must write a de- duplicated list of these numbers to a log file in no particular order.

Primary Considerations

- The Application should work correctly as defined below in Requirements.
- The overall structure of the Application should be simple.
- The code of the Application should be descriptive and easy to read, and the build method and runtime parameters must be well-described and work.
- The design should be resilient with regard to data loss.
- The Application should be optimized for maximum throughput, weighed along with the other Primary Considerations and the Requirements below.

Requirements

1. The Application must accept input from at most 5 concurrent clients on TCP/IP port 4000.
2. Input lines presented to the Application via its socket must either be composed of exactly nine decimal digits (e.g.: 314159265 or 007007009) immediately followed by a server-native newline sequence; or a termination sequence as detailed in #9, below.
3. Numbers presented to the Application must include leading zeros as necessary to ensure they are each 9 decimal digits.
4. The log file, to be named "numbers.log", must be created anew and/or cleared when the Application starts.
5. Only numbers may be written to the log file. Each number must be followed by a server-native newline sequence.
6. No duplicate numbers may be written to the log file.
7. Any data that does not conform to a valid line of input should be discarded and the client connection terminated immediately and without comment.
8. Every 10 seconds, the Application must print a report to standard output:
    1. The difference since the last report of the count of new unique numbers that have been received.
    2. The difference since the last report of the count of new duplicate numbers that have been received.
    3. The total number of unique numbers received for this run of the Application.
    4. Example text for #8: Received 50 unique numbers, 2 duplicates. Unique total: 567231

9. If any connected client writes a single line with only the word "terminate" followed by a server-native newline sequence, the Application must disconnect all clients and perform a clean shutdown as quickly as possible.
10. Clearly state all of the assumptions you made in completing the Application.

Notes

- You may write tests at your own discretion. Tests are useful to ensure your Application passes Primary Consideration A.
- You may use common libraries in your project such as Apache Commons and Google Guava, particularly if their use helps improve Application simplicity and readability. However the use of large frameworks, such as Akka, is prohibited.
- Your Application may not for any part of its operation use or require the use of external systems, for example Apache Kafka or Redis.
- At your discretion, leading zeroes present in the input may be stripped—or not used—when writing output to the log or console.
- Robust implementations of the Application typically handle more than 2M numbers per 10-second reporting period on a modern MacBook Pro laptop (e.g.: 16 GiB of RAM and a 2.5 GHz Intel i7 processor).
- To test if your application is working as expected, you can try to telnet to it through the port 4000 by executing:
  ```
  > telnet localhost 4000
  ```
  And manually type in the numbers sequentially followed by a newline (enter).