1. With RPC, the code to execute the function is on a different computer to the code which calls the function. This means that parameters and return values need to be transmitted over a network rather than just being stored in registers or shared memory. This means that there can be network failures or delays while executing the RPC.
2. If the client sends a timestamp or even a random identifier with the request, the server could keep a log of the requests it has already fulfilled. It could cache the results of the function and if it receives a request with an ID in the log, it could simply send back the cached result.

Alternatively, the server could be set up such that every RPC is idempotent, so it can safely execute repeat requests. This eliminates the need for a persistent log or cache.

|  |  |  |
| --- | --- | --- |
| Message | Timestamp at send | Timestamp at receive |
| m1 | 1 | 2 |
| m2 | 2 | 3 |
| m3 | 3 | 4 |
| m4 | 3 | 5 |
| m5 | 4 | 5 |
| m6 | 6 | 7 |
| m7 | 7 | 8 |
| m8 | 9 | 10 |
| m9 | 8 | 11 |

|  |  |  |
| --- | --- | --- |
| Message | Timestamp at send | Timestamp at receive |
| m1 | [1,0,0,0] | [1,1,0,0] |
| m2 | [2,0,0,0] | [2,0,1,0] |
| m3 | [3,0,0,0] | [3,0,0,1] |
| m4 | [1,2,0,0] | [3,2,0,2] |
| m5 | [1,3,0,0] | [2,3,2,0] |
| m6 | [2,3,3,0] | [3,3,3,3] |
| m7 | [2,3,4,0] | [4,3,4,0] |
| m8 | [5,3,4,0] | [5,4,4,0] |
| m9 | [2,3,5,0] | [5,5,5,0] |

|  |  |  |  |
| --- | --- | --- | --- |
| Events | | Lamport | Vector |
| send(m2) | send(m3) | Y | Y |
| send(m3) | send(m5) | Y | N |
| send(m5) | send(m9) | Y | Y |

* 1. For process scheduling, monotonic clocks are most appropriate because they are good at measuring elapsed time within one node.
  2. For I/O, monotonic clocks are also the most appropriate because the I/O devices likely don’t have their own internal timer, and so there is no need to worry about synchronising clocks across nodes.
  3. For distributed filesystem consistency, logical clocks are most appropriate because it is important to know which updates came in which order, but not necessarily the date/time at which they occurred.
  4. For cryptographic certificate validity, time-of-day clocks with NTP are most appropriate because certificates can be set to expire at a given date and time, and the nodes need to know whether that is in the future or the past.
  5. For concurrent database updates, logical clocks are most appropriate for the same reason as c.
  6. An operation whose effect does not change no matter how many times it is invoked. For example, an idempotent RPC does not need to be deduplicated on the server.
  7. The underlying system hiding the location of a resource. For example, an RPC uses location transparency to hopefully ensure that the programmer performs the RPC just like a local function call.
  8. Converting an abstract data object into a well-structures sequence of bytes which can be reconstructed into the data object. For example, data can be marshalled in order to be transferred over a network and then unmarshalled by the recipient.
  9. A name which itself conveys no information and can only be used to compare with similar names. For example, a random identifier for a database record is a pure name.
  10. A name which does convey some information and commits the system to maintaining the context in which it can be resolved. For example, an email address is an impure name because it contains information such at the hostname of the email server.
  11. Performing multiple synchronous operations at once. For example, instead of making a separate RPC each passing one identifier and returning one attribute (e.g., when looking up records in a database), rather send a single RPC request with an array of identifiers which returns an array of attributes. This masks network latency.
  12. Software which lies in between applications and the network connection. For example, a broadcast algorithm.
  13. The difference between two clocks at a point in time.
  14. The gradual increase in clock error over time.

1. See Q14 above