

## 1 1998 Paper 12 Question 1

For a large-scale distributed system or application:

- (a) Describe alternative approaches to creating unique names. Include a discussion of the information conveyed by a name.

A name is a unique identifier paired with information about the node. The information paired with the node can be arbitrary but should be immutable and generally relevant. For example information about the hardware or software the node is running or a “priority”. If we assume a fail-stop model then the time the node was created would also be reasonable. Critically, the information in the nodes name can only be provided by the node which named that node. So if a node named itself then it may not know its time of creation or its own priority.

There are a three main ways to generate a unique identifier:

- Delegated by by a naming service such as a router or leader (RAFT).

This unique identifier can simply be a counter. If given by a “superior” then it should use a unique “superior ID” as a prefix.

- A function of something known to be unique.

For example the MAC address or the ID of the user who is working on the node paired with the wall time at which they logged in.

- A random number

This is not guaranteed to be unique; but can be designed such that the probability of uniqueness is astonishingly high. For most systems, a 64-bit number would be sufficient and could even be paired with the current wall time to bring the probability of uniqueness even higher and stop the probability of ever having a collision in a given second increasing.

For example, if we used a random 128-bit integer paired with the wall time (ms-precision) then the probability of the Internet having a name collision in the next million years is  $\approx 1.4 \times 10^{-14}$ . This assumes that the seed to the random number generator is *not* the wall-time.

- (b) Discuss name to location binding under the headings

- (i) availability of the binding service

To get high availability of the binding service, the binding service should be local. In this case either a random number or a function of an existing unique identifier is preferable. If names are given by a leader then the node may have to wait to be given a name.

- (ii) consistency of the naming data

To ensure data put in the name is consistent, the data should all come from the same source: a coordinator. In this case it is preferable to use strategy one – where a leader or superior delegates names.

- (iii) mobility of named objects

If a leader delegates a name, then it may not be unique on other networks. This is therefore unusable. Additionally, if network-specific information (such as time of joining or priority) is included in the name then the named object is not portable. Names generated under strategy 2 or strategy 3 are globally unique.



<https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y1998p12q1.pdf>



## 2 1999 Paper 9 Question 1

Describe one of the following architectures for supporting distributed programming:

- Either**            Remote Procedure Call  
  
**or**                Object Request Brokerage



<https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y1999p9q1.pdf>

RPC is a method of providing location transparency which involves wrapping a function call to another node inside a “stub” which has the same signature and arguments as the function. From the users perspective, they make a local function call to the stub.

When the stub is called, instead of performing the function, it marshalls the arguments into a transmittable format (ie JSON) then sends them across the network to the node on which the function is to be performed. The target node then unmarshalls the data and performs the function. Dependent on the exact semantics, a response or return value may be sent or the original sender may be polled.

Many emerging applications require timely responses to events. Discuss the following approaches to achieving this:

(a) polling

Polling is the process of regularly checking if an event has happened. In the context of distributed systems, this would mean sending a message to the node on which the RPC was performed every  $\tau$ s. This is a very inefficient strategy and will greatly increase network traffic.

(b) synchronous call back

In synchronous call-back, the thread which made the RPC blocks until the call-back returns at which point it is immediately woken. This is analagous to a blocking system call in an Operating System.

(c) asynchronous notification

In asynchronous notification, the thread which makes the RPC can continue executing and will receive an asynchronous notification when the RPC completes. This is analagous to an asynchronous IO operation in an Operating System.

