❖ Remote Procedure Call (RPC)

RPC is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network details. A procedure call is also called a function call or subroutine call. RPC uses the client server model.

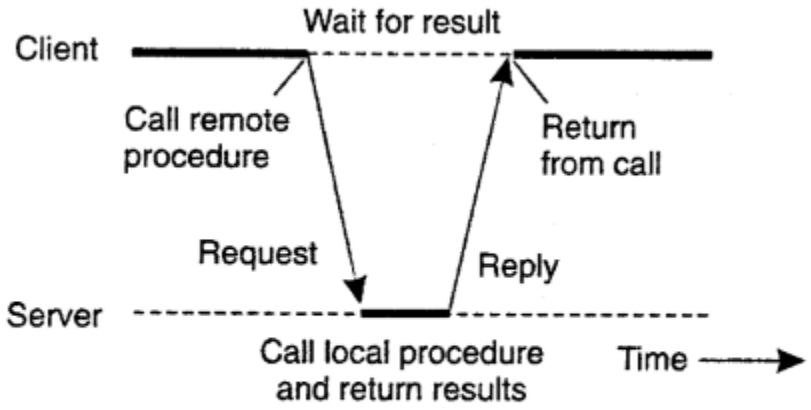


Figure 4-6. Principle of RPC between a client and server program.

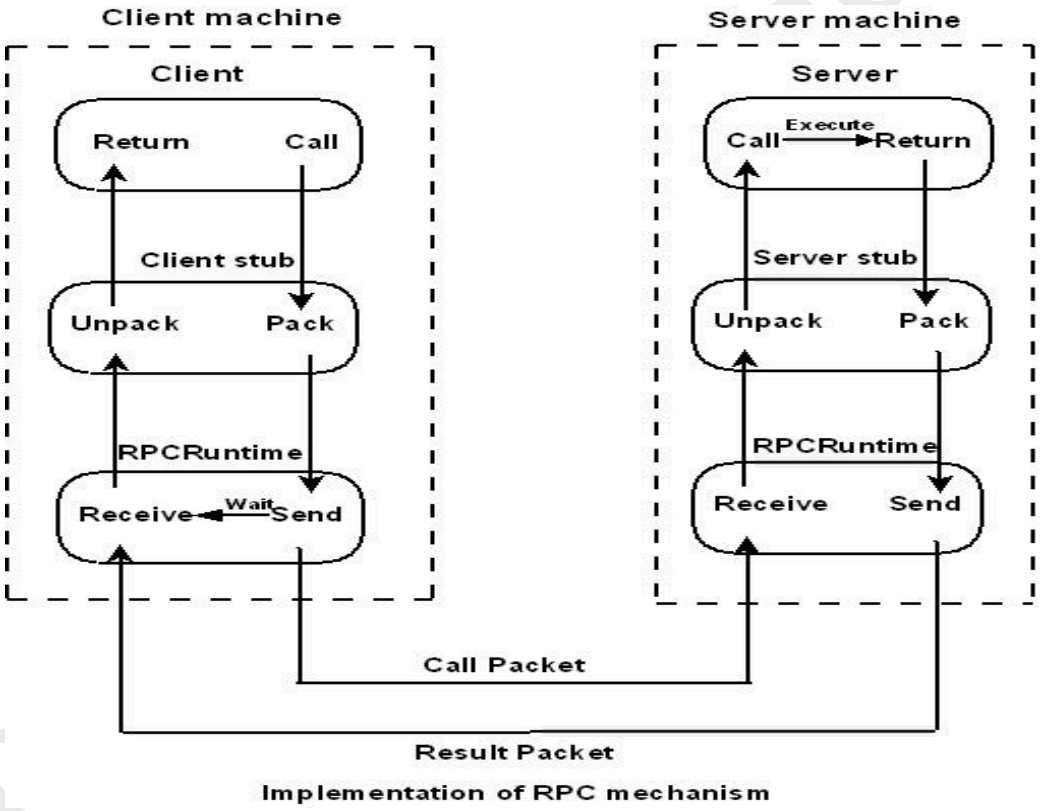
It includes five elements:

- The client
- The client stub
- The RPC runtime
- The server stub
- The server

1. Client

A Client is a user process which initiates a RPC

The client makes a normal call that will invoke a corresponding procedure in the client stub.



Implementation of RPC mechanism

2. Client Stub

Client stub is responsible for the following two tasks:

On receipt of a call request from the client, it packs specifications of the target procedure and arguments into a message and asks the local RPC Runtime to send it to the server stub.

On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

3. RPC Runtime

Transmission of messages between Client and the server machine across the network is handled by RPC Runtime.

It performs Retransmission, Acknowledgement, Routing and Encryption.

RPC Runtime on Client machine receives messages containing result of procedure execution from server and sends it client stub as well as the RPC Runtime on server machine receives the same message from server stub and passes it to client machine.

4. Server Stub

Server stub is similar to client stub and is responsible for the following two tasks:

On receipt of a call request message from the local RPC Runtime, it unpacks and makes a normal call to invoke the required procedure in the server.

On receipt of the result of procedure execution from the server, it packs the result into a message and then asks the local RPC Runtime to send it to the client stub.

5. Server

When a call request is received from the server stub, the server executes the required procedure and returns the result to the server stub.

Steps of RPC

1. Client procedure calls client stub in normal way.
2. Client stub builds a message, calls local OS.
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Parameter Passing

In RPC, the parameters are passed in two ways - Pass by value and Pass by reference.

In pass by value, the actual parameters and their data types are copied in to the stack and passed to the called procedure.

In pass by reference, the pointer to the data is passed instead of value to the called procedure at the server side.

It is very difficult to implement as the server needs to keep the track of the pointer to the data at the client's address space.

Packing parameters into a message is called parameter marshaling. As a very simple example, consider a remote procedure, add(i, j), that takes two integer parameters i and j and returns their arithmetic sum as a result. The call to add, is shown in the left-hand portion (in the client process) in Fig. 4-7. The client stub takes its two parameters and puts them in a message as indicated. It also puts the name or number of the procedure to be called in the message because the server might support several different calls, and it has to be told which one is required.

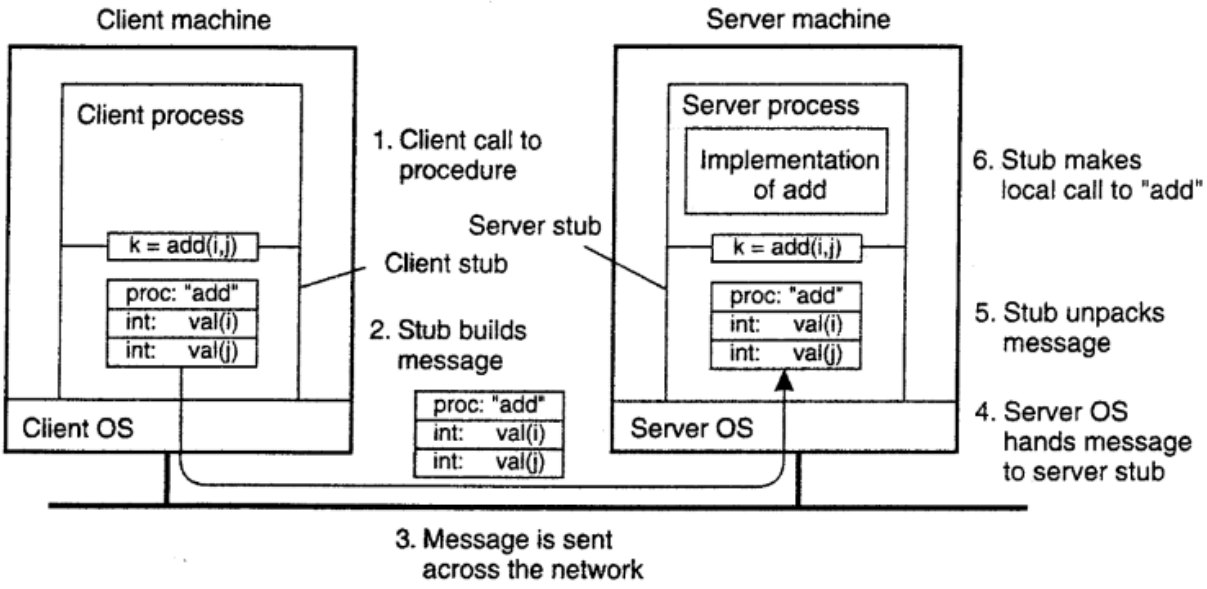


Figure 4-7. The steps involved in a doing a remote computation through RPC.

When the message arrives at the server, the stub examines the message to see which procedure is needed and then makes the appropriate call. If the server also supports other remote procedures, the server stub might have a switch statement in it to select the procedure to be called, depending on the first field of the message. The actual call from the stub to the server looks like the original client call, except that the parameters are variables initialized from the incoming message.

When the server has finished, the server stub gains control again. It takes the result sent back by the server and packs it into a message. This message is sent back to the client stub. which unpacks it to extract the result and returns the value to the waiting client procedure.

Asynchronous RPC

As in conventional procedure calls, when a client calls a remote procedure, the client will block until a reply is returned. This strict request-reply behavior is unnecessary when there is no result to return, and only leads to blocking the client while it could have proceeded and done useful work just after requesting the remote procedure to be called. Examples of where there is often no need to wait for a reply include: transferring money from one account to another, adding entries into a database, starting remote services, batch processing, and so on.

To support such situations, RPC systems may provide facilities for what are called asynchronous RPCs, by which a client immediately continues after issuing the RPC request. With asynchronous RPCs, the server immediately sends a reply back to the client the moment the RPC request is received, after which it calls the requested procedure. The reply acts as an acknowledgment to the client that the server is going to process the RPC. The client will continue without further blocking as soon as it has received the server's acknowledgment. Fig. 4-1O(b)shows how client and server interact in the case of asynchronous RPCs. For comparison, Fig. 4-10(a) shows the normal request-reply behavior.

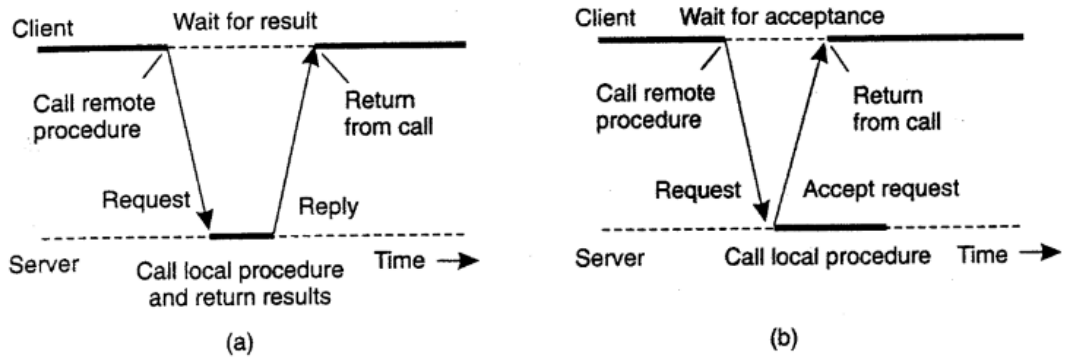


Figure 4-10. (a) The interaction between client and server in a traditional RPC. (b) The interaction using asynchronous RPC.

Asynchronous RPCs can also be useful when a reply will be returned but the client is not prepared to wait for it and do nothing in the meantime. For example, a client may want to prefetch the network addresses of a set of hosts that it expects to contact soon. While a naming service is collecting those addresses, the client may want to do other things. In such cases, it makes sense to organize the communication between the client and server through two asynchronous RPCs, as shown in Fig. 4-11.

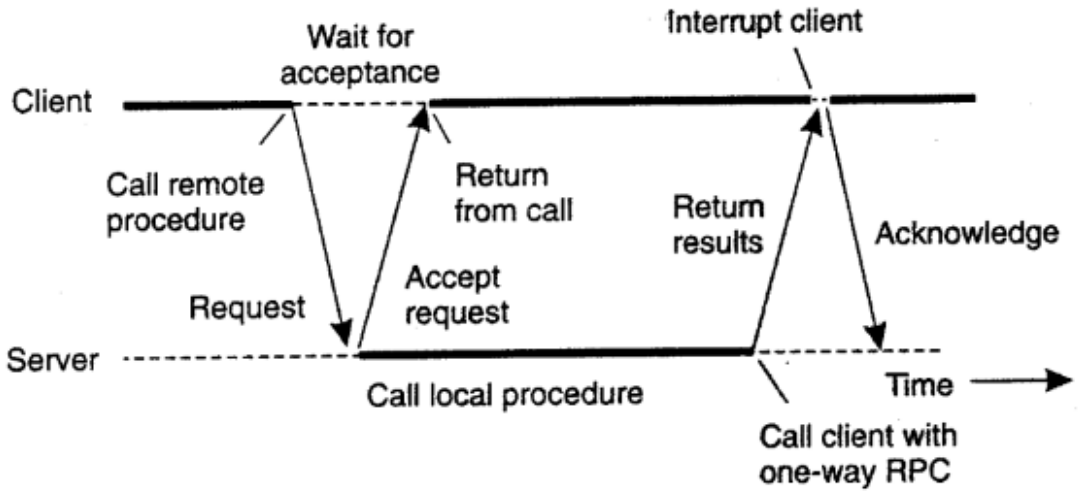


Figure 4-11. A client and server interacting through two asynchronous RPCs.

The client first calls the server to hand over a list of hostnames that should be looked up, and continues when the server has acknowledged the receipt of that list. The second call is done by the server, who calls the client to hand over the addresses it found. Combining two asynchronous RPCs is sometimes also referred to as a deferred synchronous RPC.

It should be noted that variants of asynchronous RPCs exist in which the client continues executing immediately after sending the request to the server. In other words, the client does not wait for an acknowledgment of the server's acceptance of the request. We refer to such RPCs as one-way RPCs.

❖ Resource Reservation Protocol in Real-time Systems

Resource Reservation Protocol (RSVP) is used in real-time systems for an efficient quality band transmission to a particular receiver. It is generally used by the receiver side for the fast delivery of the transmission packets from the sender to the receiver. Features of Resource Reservation Protocol:

- RSVP is receiver initiated. Receiver node in the real time system initiates the protocol.
- RSVP is simplex(unidirectional). The receiver node just receives the packets and does not want to send any data.
- Quality of Service is provided by RSVP protocol.
- Admission Control is used in RSVP at each hop in the network topology.
- Classification, buffer management, and scheduling is managed efficiently by RSVP
- It dynamically adapts to the change in route for the efficient message transfer.
- It is actually not a routing protocol. It depends upon other routing protocols.
- RSVP operates at the Transport layer of the OSI model and is used to reserve resources for a specific flow of data between a sender and receiver.
- It can be used for both unicast and multicast communication.
- RSVP uses soft state mechanism, which means that it periodically refreshes the state information for a particular flow of data.

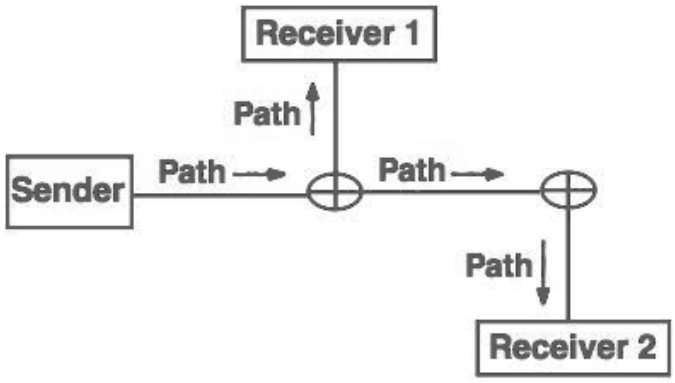
- It supports different types of traffic, including constant bit rate, variable bit rate, and on-demand traffic.
- RSVP can be used in conjunction with Differentiated Services (DiffServ) to provide Quality of Service (QoS) in a network.
- It uses a signaling message called RSVP PATH message to request for resources and RSVP RESV message to reserve resources for a particular flow.
- RSVP can support multiple QoS levels for a particular flow of data.
- It is widely used in multimedia applications such as video conferencing, streaming, and online gaming.

Flowspec is used in RSVP to determine the different parameters like bandwidth, link strength, congestion, etc for smooth and collisionless communication and data transfer. Filterspec is used in RSVP for filtering of the packets. It is used to route the packets according to its destination type as a fixed or shared receiver.

Types of messages used in RSVP connection establishment:

- Reservation Message (resv): The receiver sends the Reservation Message (resv) to the sender, which specifies all the required resources and parameters for the reservation to establish.
- Path Message (path): Upon receiving the reservation message from the receiver, the sender records all the necessary resources to be reserved and records the path. The sender multicasts a Path Message (path) to all the receivers, which specifies the routing details of the packet. It also contains all the necessary specifications about the reservation to be made for the receiver.

Path Messages Transfer:

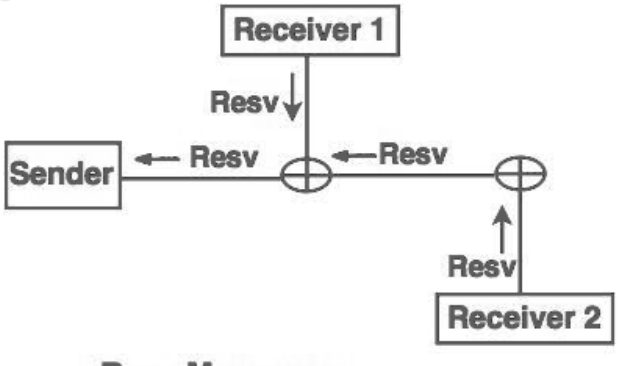


Path messages

The receivers in a flow make the reservation in RSVP, but the receivers do not know the path traveled by the packets before the reservation. The path is required for reservation To solve this problem the RSVP uses the path messages.

A path message travels from the sender and reaches to all receivers by multicasting and path message stores the necessary information for the receivers.

Reservation Messages Transfer:



Resv Messages

After receiving a path message, the receiver sends a Resv message. The Resv message travels to the sender and makes a resource reservation on the routers which support RSVP.

The data sent by the sender in Resource Reservation Protocol is encrypted to prevent the data sent to the receiver from breach. Error reporting is done at the sender side in Resource Reservation Protocol for making the necessary changes in the communication strategy. In case of failure in RSVP, the admission state of the hop is sent to the requester for handling the necessary packets. In RARP the routers record the necessary forward and reverse routing state. The router may also make the necessary changes in the path message sent by the sender to the receiver to indicate the actual resource availability.

Advantages of Resource Reservation Protocol (RRP) in Real-Time Systems:

- Predictable Resource Allocation: RRP allows for the reservation of system resources in advance, ensuring predictable and guaranteed resource availability for real-time tasks. This enables the system to meet timing constraints and ensure timely execution of critical tasks.
- Deterministic Behavior: RRP provides deterministic behavior in real-time systems by allocating fixed resources to specific tasks. This determinism allows for precise timing analysis and ensures that tasks can meet their deadlines with high accuracy.
- Quality of Service Guarantees: RRP enables the system to provide quality of service guarantees by reserving resources based on the requirements of real-time tasks. It ensures that tasks receive the necessary resources and can operate within specified timing constraints, leading to reliable system behavior.
- Resource Optimization: RRP allows for efficient resource utilization by reserving resources only when they are needed. This prevents resource wastage and maximizes resource availability for other tasks, leading to overall improved system performance.

Disadvantages of Resource Reservation Protocol (RRP) in Real-Time Systems:

- Increased Resource Overhead: Implementing RRP requires additional resources for reservation and management, including memory for storing reservation information, computational power for scheduling and resource allocation decisions, and communication overhead for maintaining the reservation protocol. This can increase the overall resource overhead of the system.
- Limited Flexibility: RRP relies on static resource reservations, which may limit the flexibility and adaptability of the system. Changes in task requirements or dynamic workload variations may require manual reconfiguration of reservations, leading to inflexibility in resource allocation.
- Scalability Challenges: RRP may face scalability challenges in large-scale real-time systems with a high number of tasks and complex resource dependencies. As the number of tasks and resource requirements increase, managing and coordinating the reservations can become more challenging and may impact system performance.
- Reservation Conflicts and Deadlock Risks: In systems where multiple tasks contend for the same resources, conflicts, and deadlocks may arise if the reservation protocol is not carefully designed and managed. Resolving conflicts and avoiding deadlocks requires careful coordination and scheduling decisions, which can add complexity to the system design.