Mobile Code

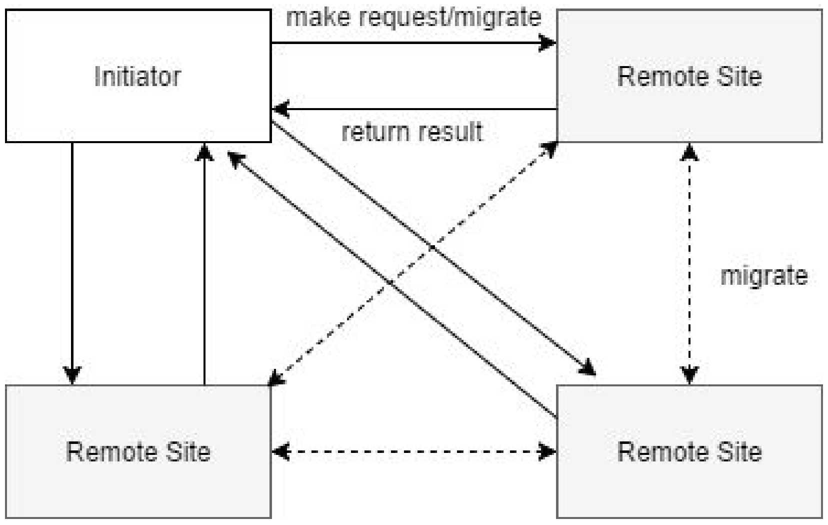
Group: APP
Betty Chen - l235chen
Hau Chen - h222chen
Phillip Jie - pyfjie
Sultaan Shah - seshah

What is the Mobile Code architectural style?

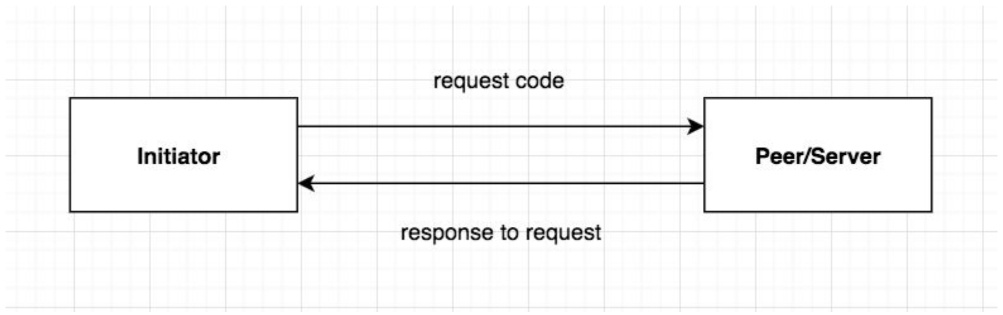
Mobile code is an architectural style that provides the ability to transmit code across a network to be interpreted by a remote machine. This may be necessary due to a lack of local resources, such as processing power or data, or due to large data sets that are remotely located. There are three components to the Mobile code architecture; code, which performs a particular computation; resources, which represent data and devices used during the computation; and computational components, which are active executors capable to carry out the computations. The connectors are the network protocols and processes for packaging data. The three design paradigms of mobile code are code-on-demand, remote evaluation and mobile agent.

Topologies

- The topology of the mobile code architectural style is the network topology. The connectors are the network communication protocols.



Code on Demand

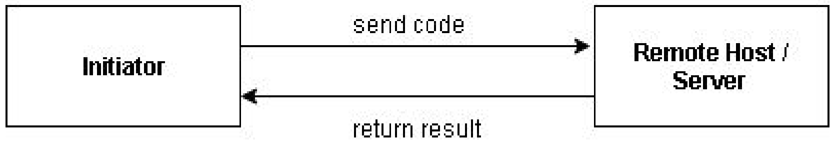


The initiator has the resources to interpret the code but lacks the code itself. A request is sent via the network to a separate server for the code. The server then responds to the initiator either by providing the code or declining the request.

Example:

You want to make an apple pie. You have at home both the required ingredients and an oven, but lack the proper recipe. However, you know that your friend Gordon Ramsay has the right recipe and he has already lent it to many friends. So, you call Gordon asking “Can you tell me your recipe to make an apple pie?” Gordon tells you the recipe and you prepare the apple pie at home.

Remote Evaluation

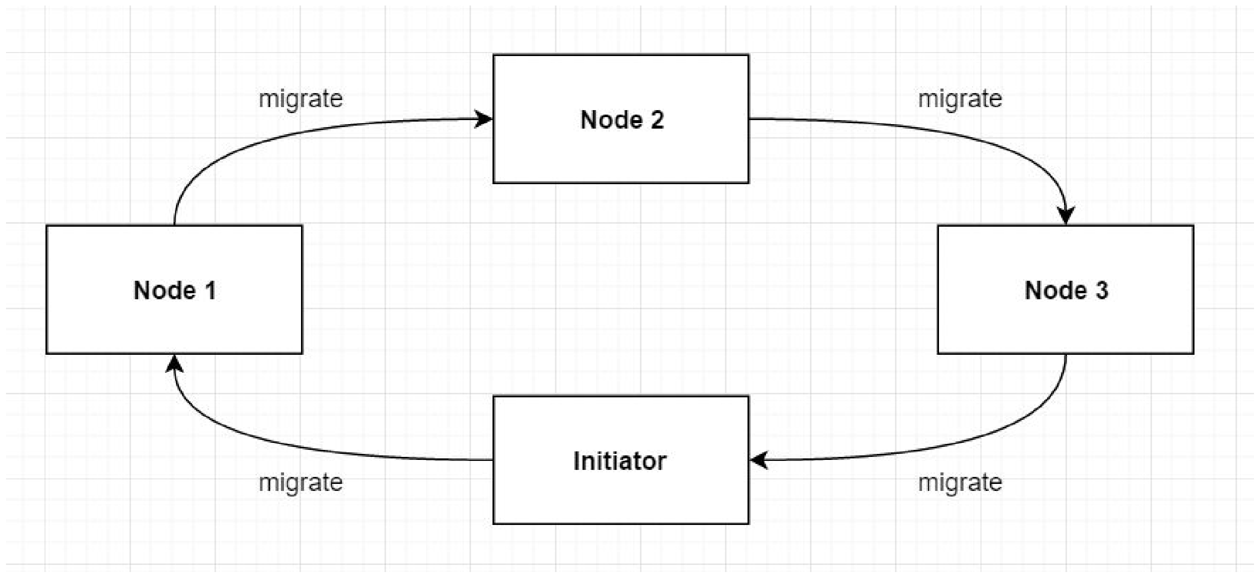


The initiator has the code, but doesn’t know how to interpret or execute it. Thus, the code is sent to a remote host that knows how to interpret it. Once the code has been interpreted or executed, the remote host will send the results back to the initiator.

Example:

You want to make an apple pie. You know the recipe but you have at home neither the required ingredients nor an oven. Your friend Gordon Ramsay has both at his place, yet he doesn’t know how to make an apple pie. You know that Gordon is happy to try new recipes, therefore you phone Gordon asking: “Can you make me an apple pie? Here is the recipe: take two apples....” Gordon prepares the apple pie following your recipe and delivers it back to you.

Mobile Agent



The Initiator has the code and might have some of the resources, however the other necessary resources are located elsewhere, a remote host for example. So the Initiator moves to that host with the code and resources. The Initiator may then move to other hosts as necessary. The processing of the code and resources does not need to return to the Initiator's local host.

Example:

You want to make an apple pie. You have the right recipe and ingredients, but you don't have an oven at home. However, you know that your friend Gordon Ramsay has an oven at his place, and that he is very happy to lend it. So, you prepare the apple pie batter and then go to Gordon's home, where he bakes the pie.

	Initiator has code (recipe)	Initiator has required resources (ingredients)	Initiator processes code (uses oven)
Code-on-demand	No	Yes	Yes
Remote evaluation	Yes	No	No
Mobile agent	Yes	No	Yes

Most applicable to specific kinds of problems?

It is suitable for distributed applications, where it requires multiple computers within the network at the same time or needs to processes large amounts of distributed data.

Engender specific kinds of change resilience?

Due to the dynamic adaptability of this architecture, if new steps are introduced to the system, the code or resources should be able to be retrieved with the new changes applied without affecting the architecture, as long as the code, resources and interpreters are up-to-date and compatible with each other. If the initiator's code has introduced new features, it would need to make sure to get the resources that would meet the demands of this feature. The overall network topology would remain the same, as the code is separate from the architecture itself.

Some specific advantages of using the style include:

- Dynamic Adaptability
  - An application can send an object to another machine, and the object can resume executing inside the application on the remote machine with the same state as it had in the originating application.
- Performance
  - The ability to request the remote execution of code helps increase server flexibility without permanently affecting the size or complexity of the server.

Some specific disadvantages of using the style include:

- Security
  - Code being executed might be malicious, if one node gets infected. Other nodes in the network can be infected as well.
- Network/Transmission Costs
  - To enable intercommunication, a set of rules need to be implemented. For example communication protocols such as HTTP and FTP. Furthermore, communication to more networks can also introduce complexity in moving the data packets from the source to destination.

Support/inhibit specific NFPs?

- Support Scalability
  - As the network grows, it takes advantage of the extra computing power from available hosts to process resources within the network.
- Support Efficiency
  - The programs can be run concurrently on multiple hosts.
  - The separation of data from the processing components allows multiple initiators to interact with the resources.
- Inhibit Security
  - By importing code from outside sources, you place yourself at risk to importing malware to your local host

References:

- Richard N. Taylor, Nenad Medvidovic, and Eric Dashofy. Software Architecture. Foundations, Theory, and Practice, Pages 116 - 119
- Alfonso Fuggetta, Gian Pietro Picco,and Giovanni Vigna. IEEE Transactions on Software Engineering, VOL. 24, NO. 5 : Understanding Code Mobility, Pages 351 - 353