

pattern a named generalization describing the elements and relationships of a solution for a commonly occurring design problem

Four essential parts of a pattern:

- descriptive name
- problem to be addressed
- solution to the problem
- consequences of adopting the pattern

The solution specifies a set of elements (in OO, classes) that will be combined to provide a solution, and the relationships among those elements.

E.g., “Wheeled Vehicle”...

Name: Wheeled Vehicle

Problem: to transport people and cargo on land

Solution:

Elements: wheel, axle, frame, body, power source

Relationships:

An axle connects two wheels.

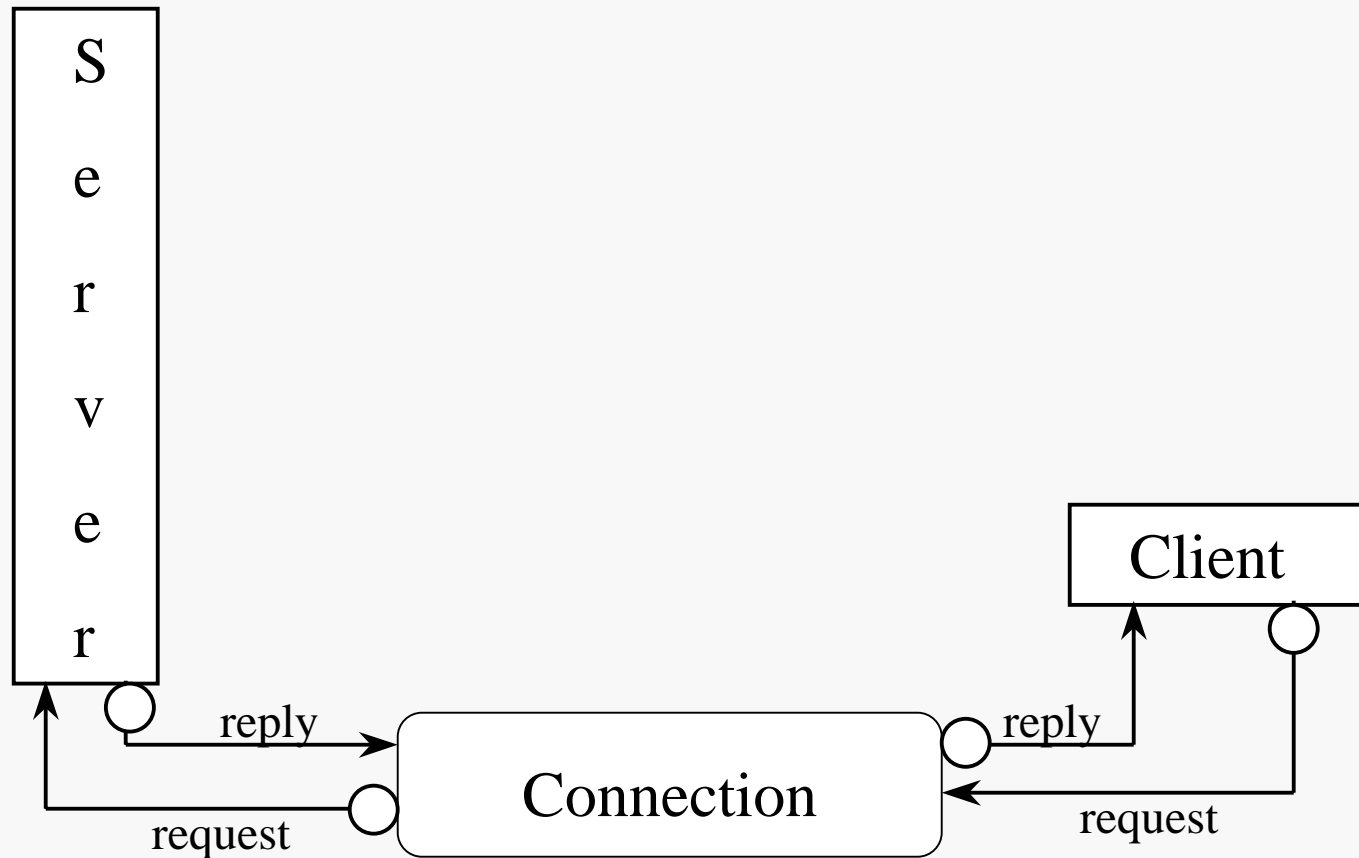
The frame is supported by one or more axles.

The power source causes one or more of the wheels to rotate.

...

Consequences: ??

Problem: to provide a service to multiple clients in a loosely coupled manner*

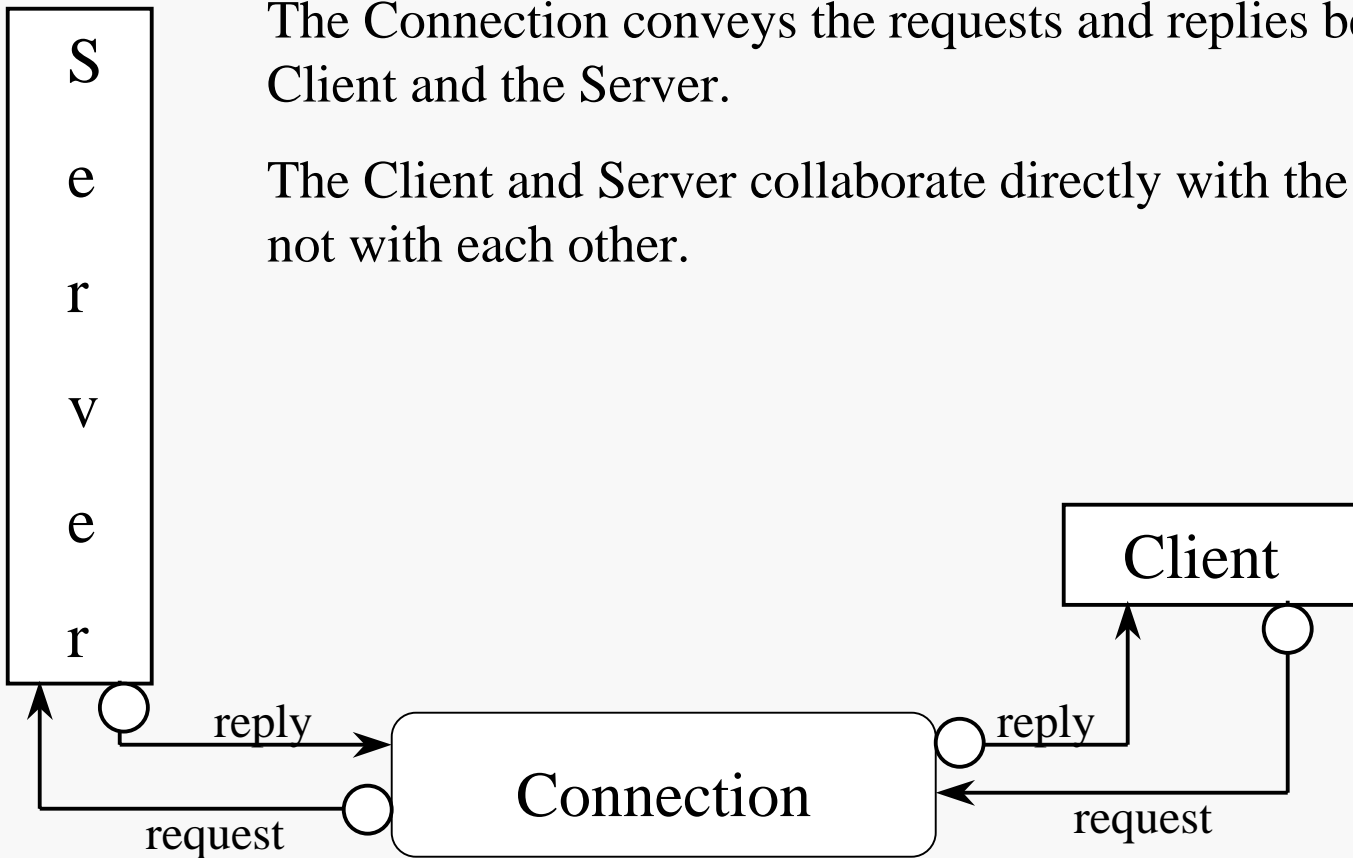


*adapted from Kafura

The Client must generate a request, which is sent to the Server, which then generates a reply to that request.

The Connection conveys the requests and replies between the Client and the Server.

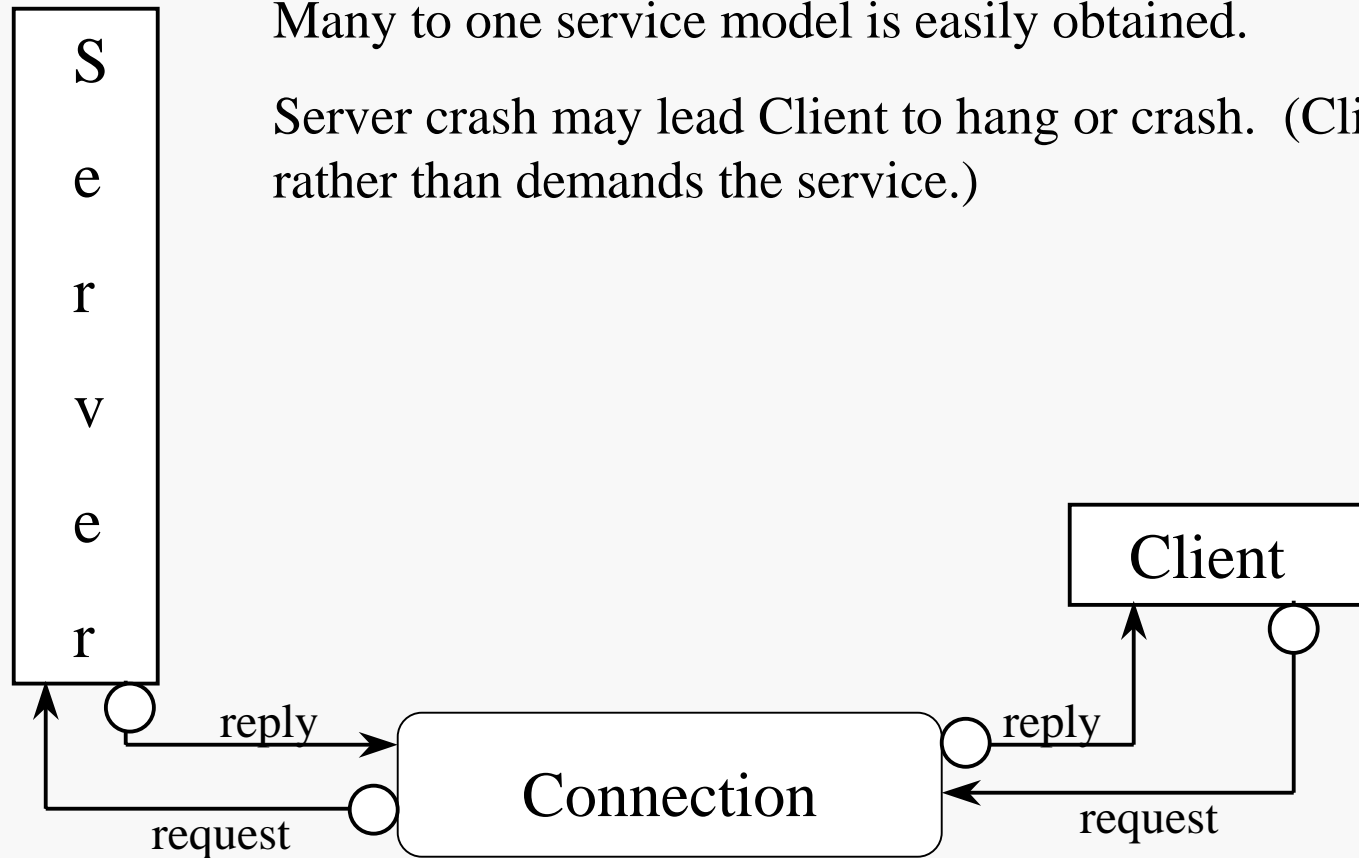
The Client and Server collaborate directly with the Connection, not with each other.



Client and Server are implementation-independent, aside from the message types that are to be passed.

Many to one service model is easily obtained.

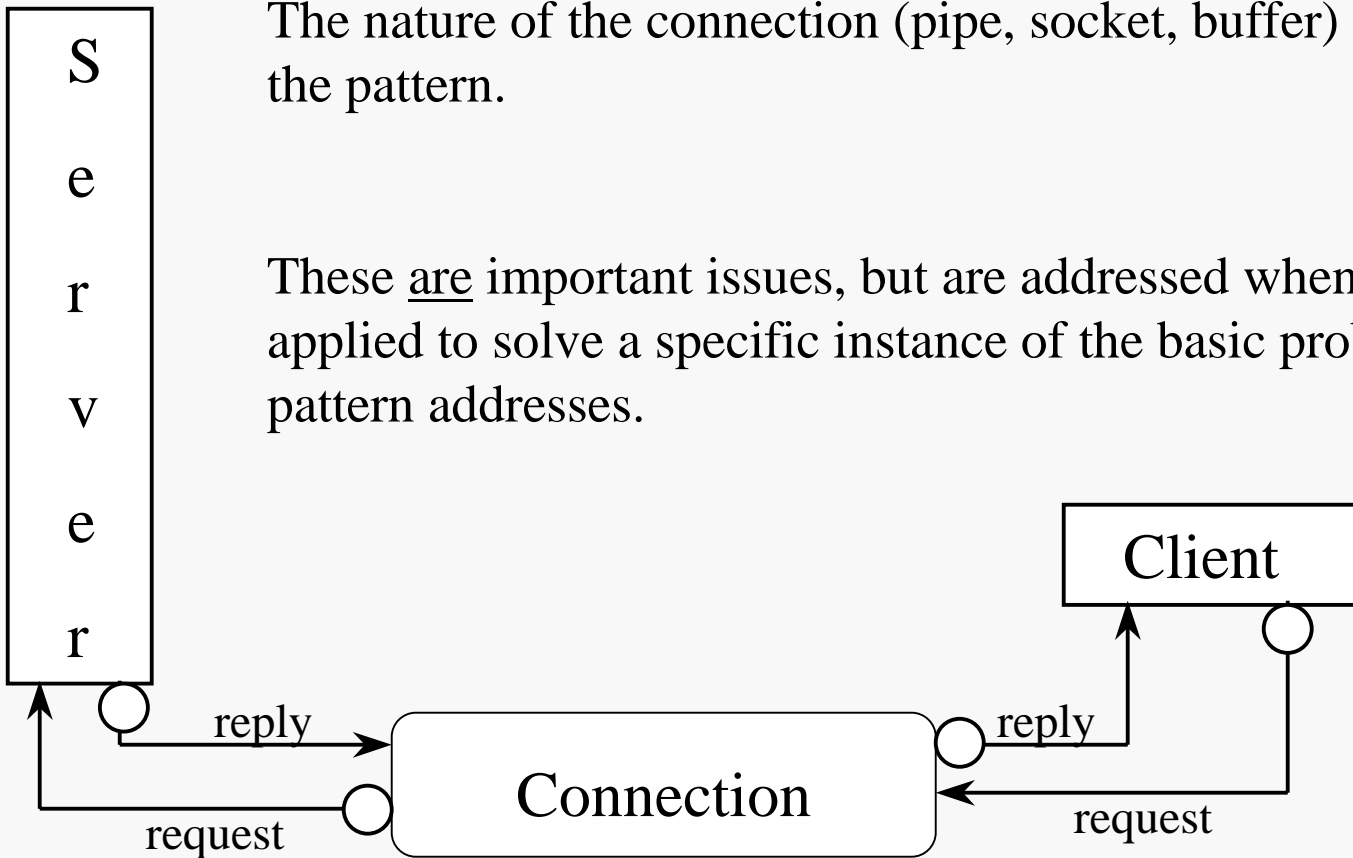
Server crash may lead Client to hang or crash. (Client requests, rather than demands the service.)



The nature of the service that is requested and supplied is irrelevant to the pattern.

The nature of the connection (pipe, socket, buffer) is irrelevant to the pattern.

These are important issues, but are addressed when the pattern is applied to solve a specific instance of the basic problem the pattern addresses.



Bounded Traversal with Conditional Exit

```
Go to first list element.  
If test is satisfied, Quit.  
While not at end of the list:  
    Step to next list element.  
    If test is satisfied, Quit.
```

Specifies the basic logical pattern of a list search.

Doesn't care if list is array, linked, or something else.

Doesn't care what the test is that must be satisfied.

Doesn't care what is to be done next.

Much of successful programming, from design to implementation, hinges upon recognizing the relevance of certain basic, well-understood patterns to the situation at hand.

The ability to do this easily and effectively is what generally separates a competent novice from a wizard.

Pattern-recognition and application is fundamental to many other activities, from sports to mathematics.

Studying an organized library of patterns may, in theory, speed up the process whereby a novice obtains a useful solution.

OTOH, there is no substitute for genuine understanding of the creative process.