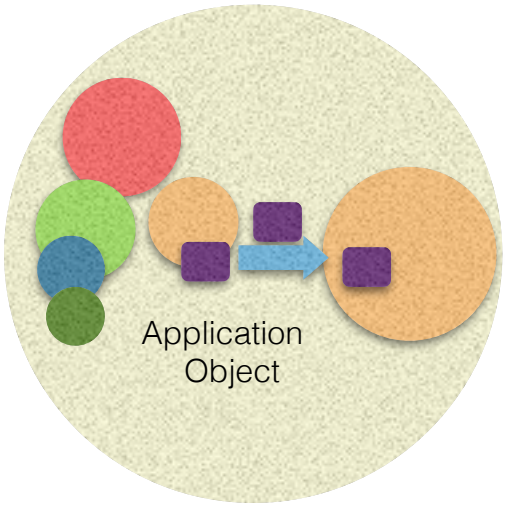
SPACE OS

A Self-Generating Networked Object Operating System

All great developers long to create their own operating system to release them from the bounds of the torturous stuff everyone else wrote. My personal OS is named SPACE. I began the concept and work more that twenty years back (early 90s) when I was running up against the limitations of network distributed applications. I did some work on the problem when I was told to publish or perish so I wrote three patent applications about parts of the system (all three applications were granted US utility patents). I was happy, but I had to pay the bills (yes, I get it, plenty of good OS's out there and I am nut).

A decade later I had time to think between jobs about a good OS (early 2000s) that would resolve issues we were having again with network distributed applications. The problem was still that we had plenty of computers sitting idle with lots of work being done on a few, 80/20 rule applies. Usually there was one server that was a log jam of a bottle neck. Systems pretty much operated like a one-man band instead of a conducted orchestra.



Even an orchestra has its limits. For example, musicians across town cannot see the conductor, or hear the other member of the orchestra so they sit idle, if not bewildered as to what to do next; all that talent producing nothing. Sorry, we did not have much of a working cloud back then, the “cloud” came about in 2012.

Sure, we could us CCTV or Wi-Fi or Mobile to feed audio and video across town but eventfully delays develop due to real world lag. The feedback loop for the conductor would be bad enough without the second violin trying to stay in step with the first violin working remotely.

So now I have come to a fork in the road where I need to decide a new career path. Part of that path is to train using interesting projects. I want to learn a few new languages. Linux being the

major OS. I want to continue my self-generating nodal network ideas of the 90s. So, I am finally starting serious work on SPACE OS.

The concept is this. An OS consisting entirely of objects, that only communicate via messages. SPACE OS Object instances are generated using XML. SPACE OS Classes are written in languages with classes. An OS that is configurable only with XML.

SPACE object instances are ethereal, self-replicating and learning. They can modify their own tries configuration. They can store their tries configurations. They can remember how they solved a problem by storing and then reloading the tries of objects. Object tries can seek out available resources, transfer replication information, create remote tries and begin to process work. The object instances are not aware of where they are, they just run. I guess that is sort of a λ lambda thing.

SPACE OS is an operating system that is built on functional objects. Each object operates on or contains one thing, a function, a collection, a data type

USE AGILE SCUM terms

A functional object has one input and one output.
A container object is passed and received from functional objects.
A container object contains data
A stream object converts objects
A job object contains a set of function objects

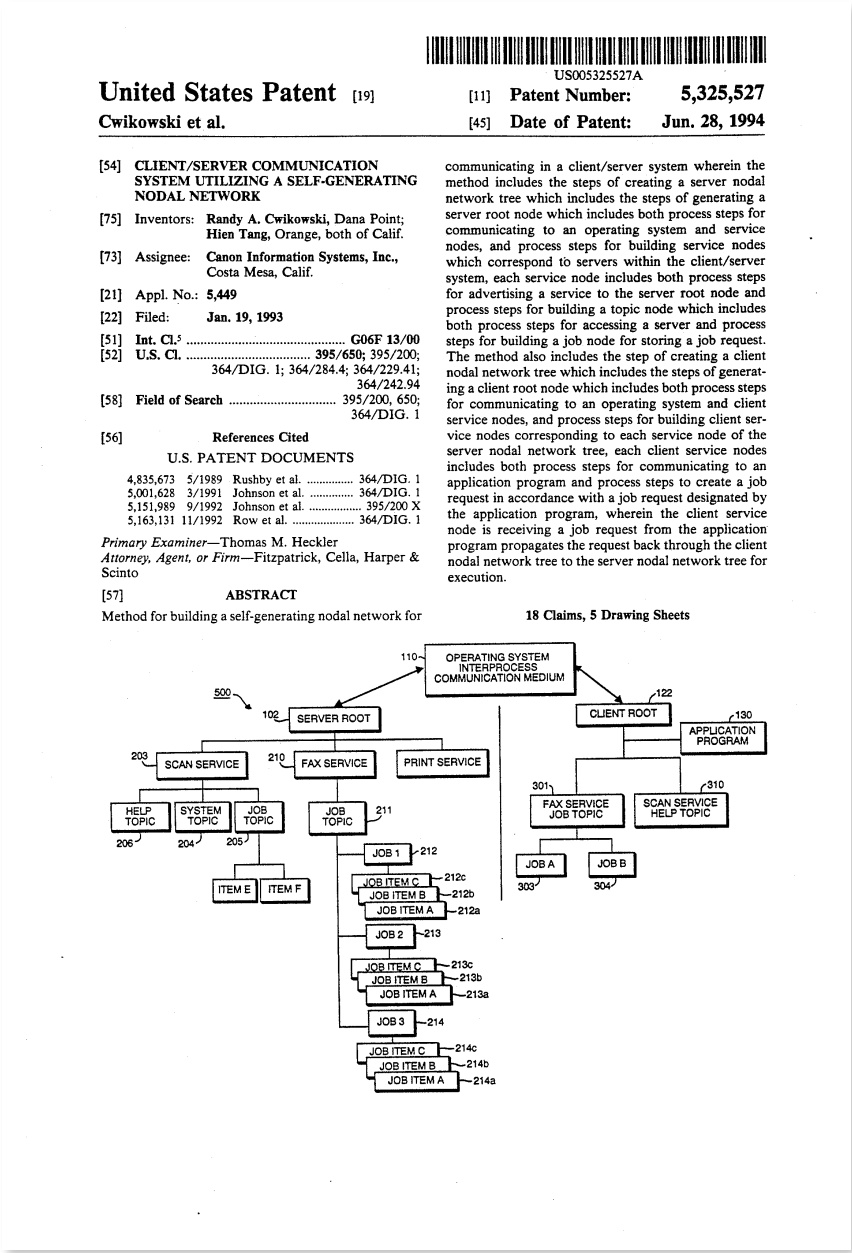
Function object instances can only be contained by one job object.
Several instances of function objects can exist

Process Object
Time Object
Hardware Object
    Display
    Storage
    Network
    Keyboard
    Mouse
Task Object
    A container of Jobs
Job Object
    Contains Functions
    Understands Timers & Schedules

contains jobs
contains threads
contains streams

Thread object
contains functions
executes function objects

Funciton object
does one operation
stateless



US005325527A

United States Patent [19]

Cwikowski et al.

[11] **Patent Number: 5,325,527**

[45] **Date of Patent: Jun. 28, 1994**

[54] **CLIENT/SERVER COMMUNICATION SYSTEM UTILIZING A SELF-GENERATING NODAL NETWORK**

[75] Inventors: **Randy A. Cwikowski**, Dana Point; **Hien Tang**, Orange, both of Calif.

[73] Assignee: **Canon Information Systems, Inc.**, Costa Mesa, Calif.

[21] Appl. No.: **5,449**

[22] Filed: **Jan. 19, 1993**

[51] **Int. Cl.**[5] **G06F 13/00**

[52] **U.S. Cl.** **395/650**; 395/200; 364/DIG. 1; 364/284.4; 364/229.41; 364/242.94

[58] **Field of Search** 395/200, 650; 364/DIG. 1

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,835,673	5/1989	Rushby et al.	364/DIG. 1
5,001,628	3/1991	Johnson et al.	364/DIG. 1
5,151,989	9/1992	Johnson et al.	395/200 X
5,163,131	11/1992	Row et al.	364/DIG. 1

Primary Examiner—Thomas M. Heckler
Attorney, Agent, or Firm—Fitzpatrick, Cella, Harper & Scinto

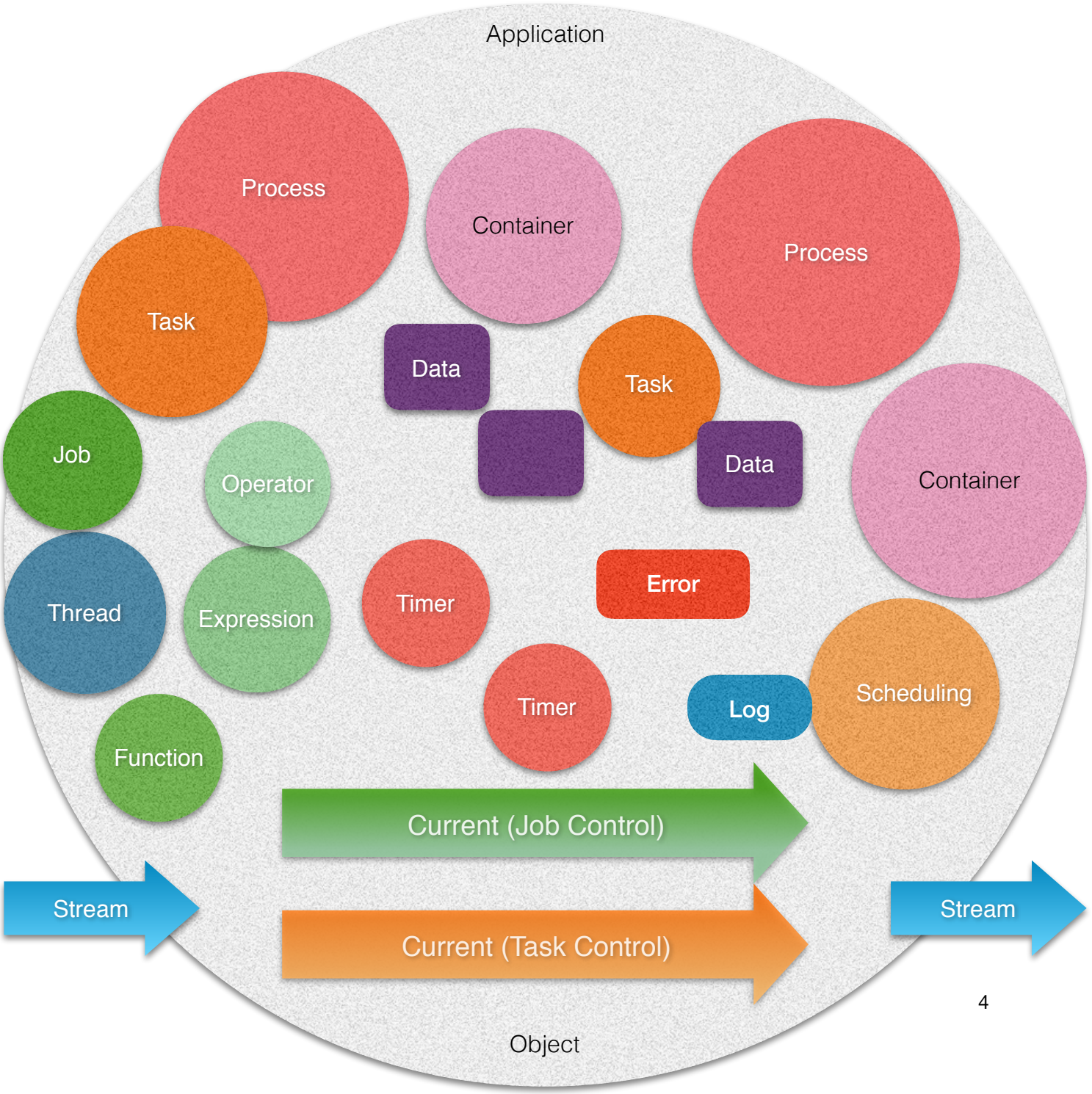
[57] **ABSTRACT**

Method for building a self-generating nodal network for communicating in a client/server system wherein the method includes the steps of creating a server nodal network tree which includes the steps of generating a server root node which includes both process steps for communicating to an operating system and service nodes, and process steps for building service nodes which correspond to servers within the client/server system, each service node includes both process steps for advertising a service to the server root node and process steps for building a topic node which includes both process steps for accessing a server and process steps for building a job node for storing a job request. The method also includes the step of creating a client nodal network tree which includes the steps of generating a client root node which includes both process steps for communicating to an operating system and client service nodes, and process steps for building client service nodes corresponding to each service node of the server nodal network tree, each client service nodes includes both process steps for communicating to an application program and process steps to create a job request in accordance with a job request designated by the application program, wherein the client service node is receiving a job request from the application program propagates the request back through the client nodal network tree to the server nodal network tree for execution.

18 Claims, 5 Drawing Sheets

Stream object (client/server subclasses) (Message queues)
operates on collection objects.
in
out
convert
transfers collection objects

Pipe Object



listens/responds
a File is a stream

Eddy
point of backlog

Container Object
stack

Current, River, Stream, Brook, Workflow, Job flow,
hydrant, spigot,

array
list
hash table

string
record
XML tag
Queues?
Record

Data Object
int
float
char
error

Error Handling

A record can only exist in one place. A container gives a pointer to the record so that others can modify it.

Records should be pulled via a queue when serializing.