



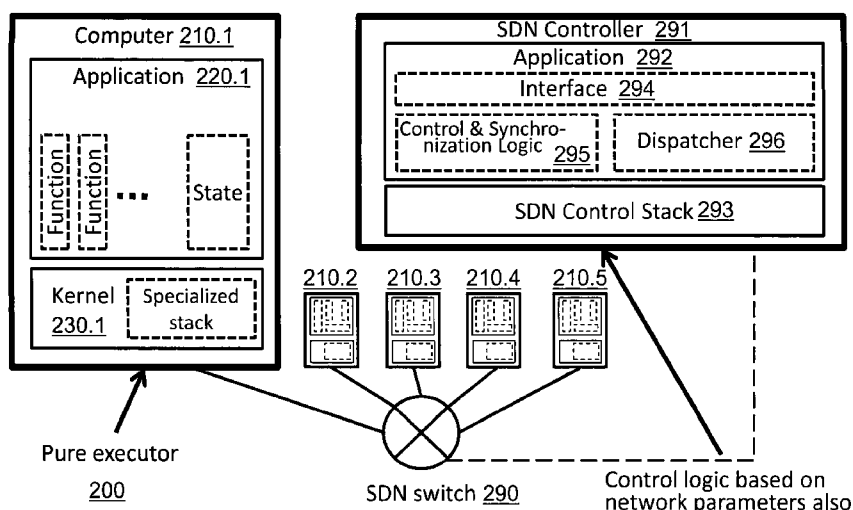
- (51) **International Patent Classification:**
H04L 12/721 (2013.01) *H04L 12/717* (2013.01)
- (21) **International Application Number:**
PCT/EP2015/052877
- (22) **International Filing Date:**
11 February 2015 (11.02.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
US61/939,771 14 February 2014 (14.02.2014) US
- (71) **Applicant:** NEC EUROPE LTD. [DE/DE]; Kurfürsten-Anlage 36, 69115 Heidelberg (DE).
- (72) **Inventors:** BIFULCO, Roberto; Bergheimerstraße 38, 69115 Heidelberg (DE). DUSI, Maurizio; Dossenheimer Landstraße 42, 69121 Heidelberg (DE).
- (74) **Agent:** ULLRICH & NAUMANN; Schneidmühlstraße 21, 69115 Heidelberg (DE).
- (81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) **Title:** SDN-BASED PROCESSING PLATFORM**Fig. 2**

Schematic View of an Embodiment of the Invention

(57) **Abstract:** A switch, or a network of switches, is connected to a plurality of computers. The switch, or the network of switches, includes a network controller configured to convert a program into a set of forwarding rules. The set of forwarding rules includes information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers. The switch, or the network of switches, is configured to forward data which includes a result of a first one of the functions from a first one of the computers for use in a second one of the functions, based on the first set of forwarding rules, such that the second one of the functions is executable by the first or a second one of the computers.

SDN-BASED PROCESSING PLATFORM

CROSS-REFERENCE TO PRIOR APPLICATION

[0001] Priority is claimed to U.S. Provisional Patent Application Serial No. 61/939,771, filed on February 14, 2014, the entire disclosure of which is hereby incorporated by reference herein.

FIELD

[0002] The present invention relates to a software-defined network (SDN) controller system for distributing processing tasks on a network.

BACKGROUND

[0003] Datacenters deploy distributed-processing platforms to execute complex computation tasks on data. Platforms such as Hadoop, Storm, Apache S4 and Blockmon allow tasks to be distributed across machines: complex tasks can be parallelized and computational power can scale out by adding nodes to the computational environment.

[0004] For reliability purposes, distributed-processing platforms must implement fault-tolerant and high-availability techniques. In case of stream computation, an additional requirement is that the application running on top of the platform should be able to process and extract information from the input messages (e.g., packets) on-the-fly.

[0005] A problem with nowadays' solutions to distributed processing of data is that tasks are basically statically assigned to nodes. Either during the design of the specific application to execute, or before deploying the application, the user or the controller embedded in the platform carries out this assignment. And this assignment will not change during the execution of the application, unless some nodes experience failure: the reassignment of nodes is done without taking into consideration the state of the network underneath, but just the state of the node itself.

[0006] However, the reassignment of tasks to machines can be triggered by reasons that are related to the network topology and load: those parameters dynamically change over time, and neither the users nor the controller that comes with the platform have knowledge about it. In the case where the computational environment is composed of virtual machines, at some point in time, it may be desirable to migrate a virtual machine where a given task is running. Reasons for doing this can be several: data locality (moving the execution closer to where the data is), change in network topology, etc.

[0007] Current solutions in datacenters rely on a control and synchronization logic, which is in charge of:

- Finding (new) execution nodes
- Handling communications between nodes
- Instructing dispatchers on the sequence of functions to call
- Managing reliability and state replications

[0008] The above control and synchronization logic is implemented both on the Platform Controller provided with the processing platform, and on the executing nodes, following a schema as provided in Figure 1.

[0009] As illustrated in Figure 1, a distributed-processing network system 100 includes a plurality of computers 110.1-110.5, and a TCP/IP switch/router 190. Each of the computers 110.1-110.5 is connected as a node via a port to the router 190. Each of the computers 110.1-110.5 includes additional components to perform respective computation tasks assigned thereto. For example, computer 110.1 includes a kernel 130.1 with a TCP/IP stack, and a collection of application programs 120.1. Application programs 120.1 include an interface, a control & synchronization logic, a dispatcher, a state memory, and a set of functions.

[0010] Through a common protocol, the Platform Controller (illustrated as the TCP/IP switch/router 190) and the overlay nodes (illustrated as the computers 110.1-110.5) exchange information on how the execution of tasks must proceed. The network underneath, for instance in the form of SDN switches, becomes then a mere executor of the above instructions: the whole computational environment cannot take advantage of the network information to optimize the execution environment.

SUMMARY

[0011] In an embodiment, the present invention provides switch, or a network of switches, connected to a plurality of computers. The switch, or the network of switches, includes a network controller configured to convert a program into a set of forwarding rules. The set of forwarding rules includes information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers. The switch, or the network of switches, is configured to forward data which includes a result of a first one of the functions from a first one of the computers for use in a second one of the functions, based on the first set of forwarding rules, such that the second one of the functions is executable by the first or a second one of the computers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The present invention will be described in even greater detail below based on the exemplary figures. The invention is not limited to the exemplary embodiments. Other features and advantages of various embodiments of the present invention will become apparent by reading the following detailed description with reference to the attached drawings which illustrate the following:

[0013] FIG. 1 illustrates a traditional distributed-processing network.

[0014] FIG. 2 illustrates a distributed-processing network according to an embodiment.

[0015] FIG. 3 illustrates a distributed-processing workflow process according to an embodiment.

[0016] FIG. 4 illustrates a distributed-processing workflow and a forwarding table according to an embodiment.

DETAILED DESCRIPTION

[0017] In an embodiment, the present invention provides a mechanism to enable the network to work as a distributed computing platform, where the processing functions are distributed to stateless edge nodes, while the program execution control flow is enforced by a software-defined network (SDN).

[0018] In the context of a network composed of SDN controller and switches which route information among nodes, the logic of the controller is moved directly into the SDN controller, and the control and synchronization logic is removed from the nodes. In this view, nodes become executor of whatever tasks they are asked to do on each incoming input (e.g., a network packet), and the SDN controller and the switches collaborate to infer the status of the network (e.g., load, topology) and forward incoming input to nodes accordingly. In fact, given a stream-computation environment, network quantities, such as dropping rate, help to infer the current state and the capabilities of a node, and this kind of quantities are already collected by switches and transmitted to the controller in a SDN network environment. As the SDN controller has full knowledge of the network underneath, it can control the load and change the network topology as needed. On the other side, nodes are mere executors and dynamic task migration can be done very efficiently.

[0019] A schema of the proposed solution according to an embodiment is given in Figure 2.

[0020] As illustrated in Figure 2, a distributed-processing network system 200 includes a plurality of nodes 210.1-210.5 which each may be a computer, and a TCP/IP switch/router 290 with a SDN controller 291. Each of the nodes 210.1-210.5 is connected as a node via a port to the router 290. Each of the nodes 210.1-210.5 includes additional components to perform respective computation tasks assigned thereto. For example, node 210.1 includes a kernel 230.1 with a TCP/IP stack, and a collection of application programs 220.1. Application programs 220.1 include a state memory, and a set of functions. The SDN controller 291 include a collection of application programs 292 and a SDN control stack 293. Application programs 292 include interface 294, control & synchronization logic 295, and dispatcher 296.

[0021] The idea is to change the computation paradigm so that each computer/node 210.1-210.5 provides and advertises to the SDN controller 291 a set of functions (capabilities) in the following form:

function_name(stream, [input list]) : [output list]

[0022] A node 210.1-210.5 is therefore modeled with the list of functions it provides, and the Platform Controller (embedded in the SDN controller 291) sees the nodes 210.1-210.5 as a list of functions that are available to be called.

[0023] As illustrated in Figure 3, users submit a program 310 to the controller 291 via user request 320, which in turn installs them as functions on the network 200 underneath according to a given allocation strategy 330.2-330.5, corresponding to specific lists of functions to be installed on corresponding specific nodes 210.2-210.5.

[0024] The controller 291 specifies the flow of the program to be executed into the flow tables of the network elements which interconnect the end-nodes 210.1-210.5 in charge of the computation, and where the operations to execute are written by the network elements directly into the packets flowing into the network.

[0025] The controller 291 converts the program flow 310 into a set of forwarding rules for the network elements. The controller 291 includes control logic that distributes computing functions to be operated by devices along the network which interconnect the computing nodes, rather than by a platform-specific logic.

[0026] Computing nodes 210.1-210.5 in a distributed environment are agnostic to the execution of the overall program and that read the computing function and variables to be executed on a given data directly from the packet itself.

[0027] One particular advantage provided by an embodiment of the invention is that nodes 210.1-210.5 are not aware of the whole program to execute: they just run particular tasks and return the results. To be precise, they are operational-oriented, meaning that they execute an operation on the incoming data, and send out the result of it. The network is instead in charge of making sure to follow the workflow and have all the nodes working cooperatively to execute the full program as specified by the user. Therefore, the workflow of a program becomes precisely a sequence of matching rules and actions which are represented exactly as the forwarding table of a switch, and they are exactly implemented as forwarding rules in the switch's table.

[0028] Figure 4 illustrates a distributed-processing workflow and a forwarding table 410 according to an embodiment.

[0029] As illustrated in Figure 4, the controller 291 converts the program flow 310 into a set of forwarding rules for the network elements, as a forwarding table 410. The forwarding table 410 lists a sequence of matching rules and actions that corresponds to the specific lists of functions installed on corresponding specific nodes 210.2-210.5 connected to ports p2-p5.

[0030] During distributed processing, the user submits data through port p1 to the switch 490 implementing the forwarding table 410. The switch 490 sends and receives data packets to and from the computing nodes 210.2-210.5 to call the specific functions, forward intermediate results, and finally obtain the output on port p6 as the result of program 310.

[0031] Given the proposed schema, the execution environment is not bound to any particular network protocol and can implement its own. In this regard, SDN devices usually are meant to read/write only the packet's headers, in which we can embed the following information: function name, output variables and extra input variables. Such fields can be used for dispatching decisions or to drive the next computation. In the packet's payload the stream of data on which the computation is actually applied can be stored and switches can leave it untouched. Adding an extra header with a "program id" would allow for execution of multiple concurrent programs. The "program id" needs to be taken into account also by the specialized stack, in order to maintain per-program state if necessary.

[0032] In stream computation, streams can be represented by a live packet capture, which requires packets to be adapted to the network-centric execution environment. This means that:

- header rewriting can take place at the network INPUT port;
- if required, rewriting can also happen at the network OUTPUT port in order to restore the original header of the packets; and/or
- If the packet header needs to be preserved, tags can be pushed in front of the header instead of rewriting it.

[0033] At this point in time, the problem of how to carry the information required by the execution environment in the packets is considered with respect to implementation details.

[0034] The SDN Controller can enforce a centralized control over the “speed” of the stream and throttle the amount of packets delivered to the nodes, e.g., the stream can be throttled at the input port (through shaping), or “queue” of nodes can be introduced in the system.

[0035] In regard to the state of a node, task migration may be bound to state migration, because the output of a function may also depend on the current internal state of the function itself. To handle this, programs/functions description can be extended to explicitly describe the presence or absence of a state; the controller is aware of the tasks which require state, and applies a strategy to migrate state among nodes at need by using techniques available known to a person of ordinary skill in the art.

[0036] Embodiments of the invention provide the following features and advantages:

- 1) A network centric execution environment for distributed computing, where a central control logic specifies the flow of the program to be executed into the flow tables of the network elements which interconnect the end-nodes in charge of the computation, and where the operations to execute are written by the network elements directly into the packets flowing into the network;
- 2) A network centric execution environment where a central control logic converts the program flow into a set of forwarding rules for the network elements;
- 3) A control logic for distributed computing operated by devices along the network which interconnect the computing nodes, rather than by a platform-specific logic; and/or
- 4) Computing nodes in a distributed environment which are agnostic to the execution of the overall program and that read the computing function and variables to be executed on a given data directly from the packet itself.

[0037] An embodiment of the invention uses the following:

- 1) A software defined network which connects the computing nodes;
- 2) A network controller able to convert a program flow description into a set of network elements forwarding rules, which encloses both the operations flow and the computing resources assignment;

- 3) A network packet encoding strategy to include the operation and input/output data used by the computing node to advance the computation;
- 4) A node's network stack able to decode/encode the information in the packet and call the corresponding node's computing operation as written in the packet; and/or
- 5) A node able to execute one or more computing operations.

[0038] Common distributed-processing platforms, such as TwitterStorm and Apache S4, rely on the presence of a controller, and interfaces on the nodes to exchange control messages with the controller itself. Operations such as task assignment or dynamic topology change are statically assigned by the controller (or by the user) before executing the distributed application. In case of failure, such topology might change, but no network load or topology information is taken into account during the execution.

[0039] Such information is embedded by routing and forwarding devices in SDN networks, and none of the existing distributed platforms take that into account.

[0040] E.g., unlike current computing platforms, an embodiment of the invention provides that:

- 1) Program Flow is encoded in the network, and the network steers the execution of the program;
- 2) End hosts can be lightweight (no control interface has to be implemented) and are unaware of the program logic: the impact that their failure has on the computation is reduced as they can be easily replaced; and/or
- 3) The functions to be executed on a packet and its additional state variables are encoded into the packet headers.

[0041] Scientific literature presented in the past the concept of enabling the network in being "computation aware". This was achieved by making switches able to execute arbitrary code when processing network flows, creating a so called active network. Two approaches were defined to this purpose, varying in the way the switches were programmed: programmable switches and capsule-based networking. Programmable switches, are network devices that could be instructed to perform flow-specific computation according to the application requirements. With this approach the switches are programmed in advance with the code which is to be executed and the corresponding network flows on which such a code

is executed. In capsule-based active networking, instead, the code that needs to be executed is carried directly in special packets, called capsules. In both cases, differently from our approach, the switch is an executor itself. More precisely, each switch executes a function, while the complete function flows is decided by the endpoints involved in the communication, e.g., programming switches in advance using capsules. In an approach according to an embodiment of the present invention, instead, the network itself enforces the program flow, while the actual functions are implemented still at network end-points.

[0042] Another approach is the process of applying a sequence of services to a given network flow (also referred to as network service chaining). In this case the network operator configures any flow to pass through a fixed set of middleboxes, each one applying a specialized function to the network flows. An example includes a network flow which needs to pass through a firewall and a NAT before accessing an end-point. In contrast to an embodiment of the present invention, in network service chaining, the network is not aware of the function executed on the devices and does not know about the actual required execution flow. The network is involved only in the role of moving packets from one middlebox to the next one, according to the static provided configuration. Hence, service chains are implemented usually as a static routing solution.

[0043] The steering itself happens by configuring static IP routes or exploiting some sort of virtual tunnels, e.g., the ones based MPLS, VLAN, etc..

[0044] An adaptation of the traffic entering the system is provided in order to correctly handle the program execution flow.

[0045] The invention can provide a high impact for the design of high-availability distributed platforms, like NEC's Blockmon and M2M CONNEXIVE platform, and in any system for batch and stream processing, like Hadoop, Spark, TwitterStorm, Apache S4. It simplifies the control operations and allows for network-based optimization in the distributed execution environment.

[0046] In an embodiment, the invention can be characterized by checking the exchange of messages between computing nodes within a topology, and the forwarding rules of the switches and the interaction between the controller and the switches in a SDN environment. Additionally, an embodiment of the invention can be characterized by the distributed platform not basing decision on network load or topology.

[0047] While the invention has been illustrated and described in detail in the drawings and foregoing description, such illustration and description are to be considered illustrative or exemplary and not restrictive. It will be understood that changes and modifications may be made by those of ordinary skill within the scope of the following claims. In particular, the present invention covers further embodiments with any combination of features from different embodiments described above and below. Additionally, statements made herein characterizing the invention refer to an embodiment of the invention and not necessarily all embodiments.

[0048] The terms used in the claims should be construed to have the broadest reasonable interpretation consistent with the foregoing description. For example, the use of the article “a” or “the” in introducing an element should not be interpreted as being exclusive of a plurality of elements. Likewise, the recitation of “or” should be interpreted as being inclusive, such that the recitation of “A or B” is not exclusive of “A and B,” unless it is clear from the context or the foregoing description that only one of A and B is intended. Further, the recitation of “at least one of A, B and C” should be interpreted as one or more of a group of elements consisting of A, B and C, and should not be interpreted as requiring at least one of each of the listed elements A, B and C, regardless of whether A, B and C are related as categories or otherwise. Moreover, the recitation of “A, B and/or C” or “at least one of A, B or C” should be interpreted as including any singular entity from the listed elements, e.g., A, any subset from the listed elements, e.g., A and B, or the entire list of elements A, B and C.

PATENT CLAIMS

What is claimed is:

1. A switch, or a network of switches, connected to a plurality of computers, the switch, or the network of switches, comprising:

a network controller configured to convert a first program into a first set of forwarding rules, the first set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers,

wherein the switch, or the network of switches, is configured to forward data which includes a result of a first one of the functions from a first one of the computers for use in a second one of the functions, based on the first set of forwarding rules, such that the second one of the functions is executable by the first or a second one of the computers.

2. The switch, or the network of switches, of claim 1, wherein the switch, or the network of switches, is configured to encode network packets to include the functions assigned to be executed in the respective ones of the computers and data for processing the functions.

3. The switch, or the network of switches, of claim 1 or claim 2, wherein the network controller is configured to convert a second program into a second set of forwarding rules, the second set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers, and wherein the second set of forwarding rules is executed in parallel to the first set of forwarding rules.

4. The switch, or the network of switches, of claim 3, wherein a program identification is encoded in the network packets to identify the first program and the second program.

5. The switch, or the network of switches, of any of claims 1 to 4, wherein the switch, or the network of switches, is configured to forward different data representing results of the different functions to different ones of the computers in accordance with the forwarding rules such that a final result of the program is obtainable without any of the computers having full knowledge of the program.

6. A method of controlling a switch, or a network of switches, connected to a plurality of computers, the method comprising:

receiving a first program via a user request at a network controller;

converting, by the network controller, the first program into a first set of forwarding rules, the first set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers; and

forwarding, by the switch, or the network of switches, data which includes a result of a first one of the functions from a first one of the computers for use in a second one of the functions, based on the first set of forwarding rules, such that the second one of the functions is executable by the second one of the computers.

7. The method of claim 6, further comprising encoding, by the switch, or the network of switches, network packets to include the functions assigned to be executed in the respective ones of the computers and data for processing the functions.

8. The method of claim 6 or claim 7, further comprising, converting, by the network controller, a second program into a second set of forwarding rules, the second set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers, the second set of forwarding rules being executable in parallel to the first set of forwarding rules.

9. The method of claim 8, wherein a program identification is encoded in the network packets to identify the first program and the second program.

10. The method of any of claims 6 to 9, wherein the switch, or the network of switches, forwards different data representing results of the different functions to different ones of the computers in accordance with the forwarding rules so as to obtain a final result of the program without any of the computers having full knowledge of the program.

11. A system comprising:

a plurality of computers connected in a network; and

a switch, or a network of switches, including a network controller configured to convert a first program into a first set of forwarding rules, the first set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers,

wherein the switch, or the network of switches, is configured to forward data which includes a result of a first one of the functions from a first one of the computers for use in a second one of the functions, based on the first set of forwarding rules, such that the second one of the functions is executable by the first or a second one of the computers.

12. The system of claim 11, wherein the switch is configured to encode network packets to include the functions assigned to be executed in the respective ones of the computers and data for processing the functions.

13. The system of claim 11 or claim 12, wherein the network controller is configured to convert a second program into a second set of forwarding rules, the second set of forwarding rules including information to control data flow between the computers and different functions assigned to be executed in respective ones of the computers, and wherein the second set of forwarding rules is executed in parallel to the first set of forwarding rules.

14. The system of claim 13, wherein a program identification is encoded in the network packets to identify the program and the second program.

15. The system of any of claims 11 to 14, wherein the switch is configured to forward different data representing results of the different functions to different ones of the computers in accordance with the forwarding rules such that a final result of the program is obtainable without any of the computers having full knowledge of the program.

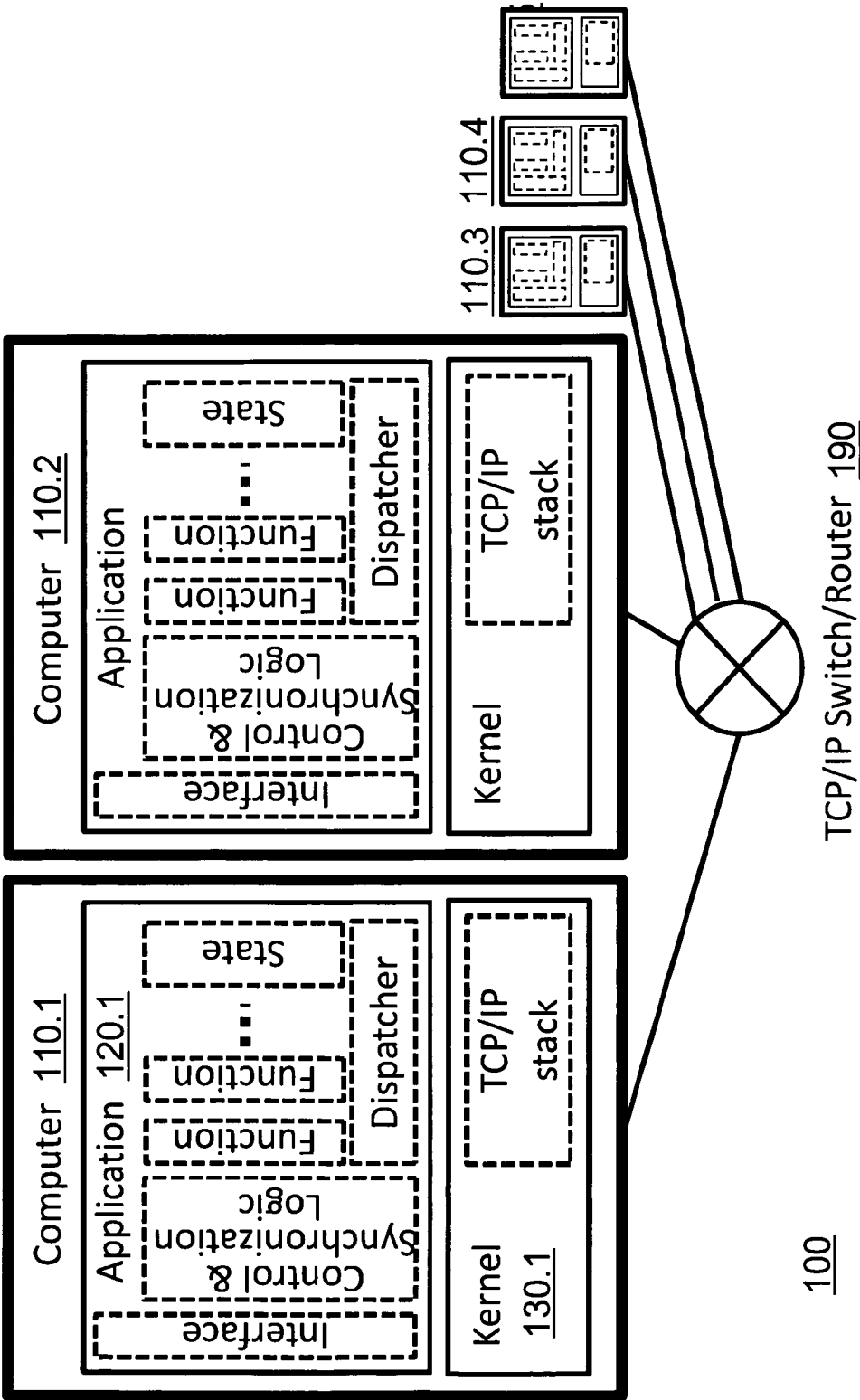


Fig. 1

Control and synchronization logic as implemented in current distributed processing platforms

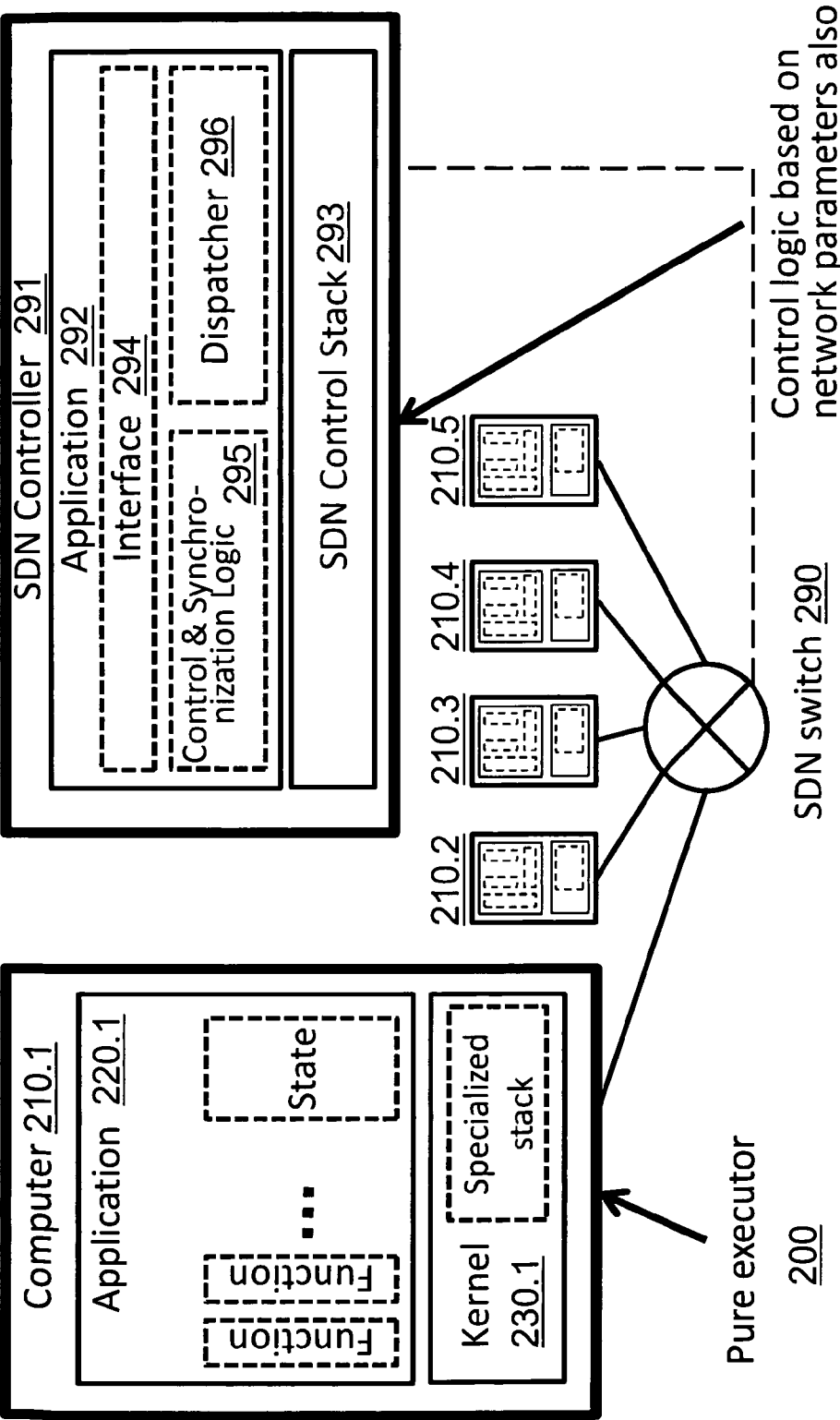


Fig. 2

Schematic View of an Embodiment of the Invention

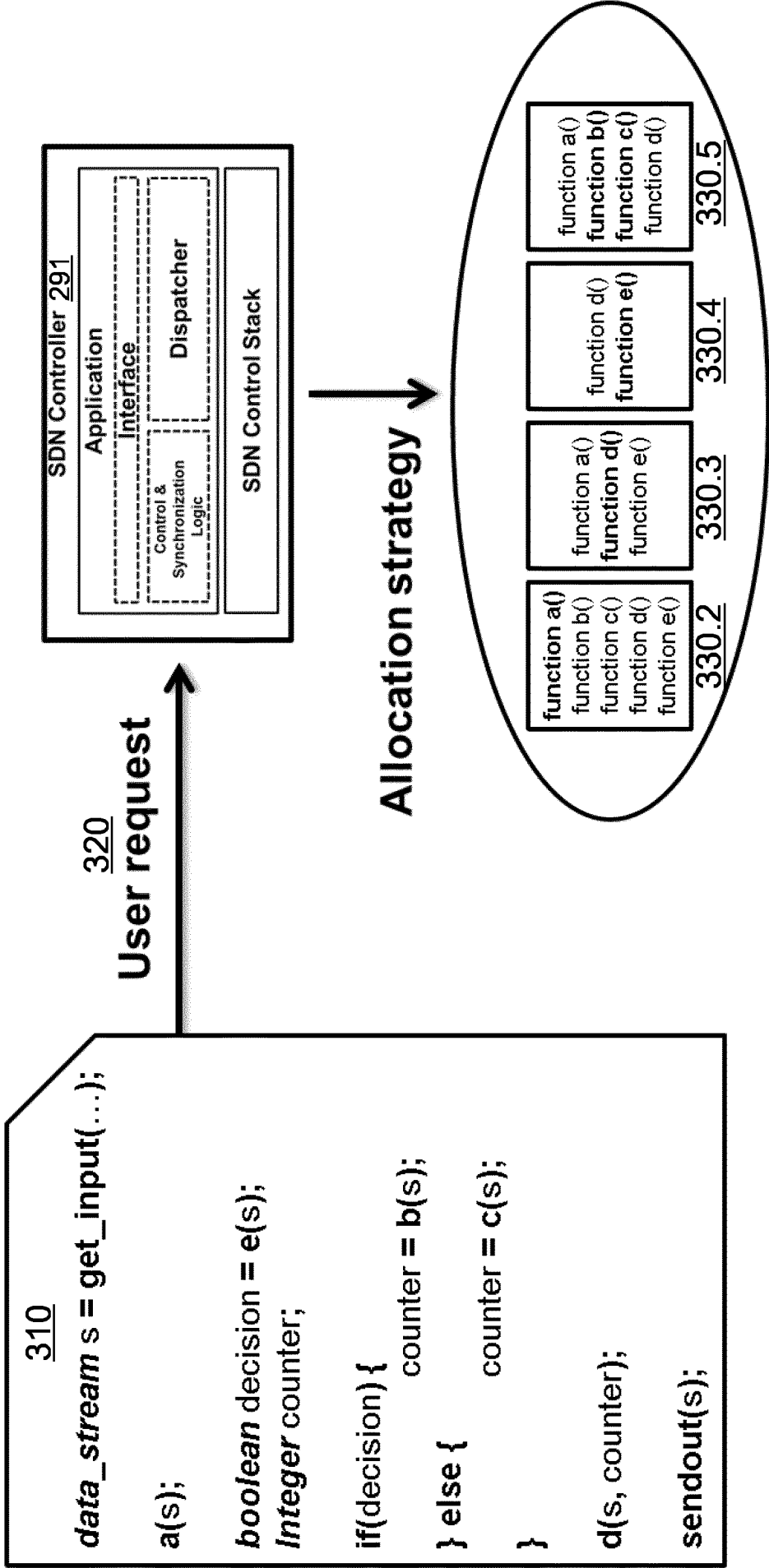
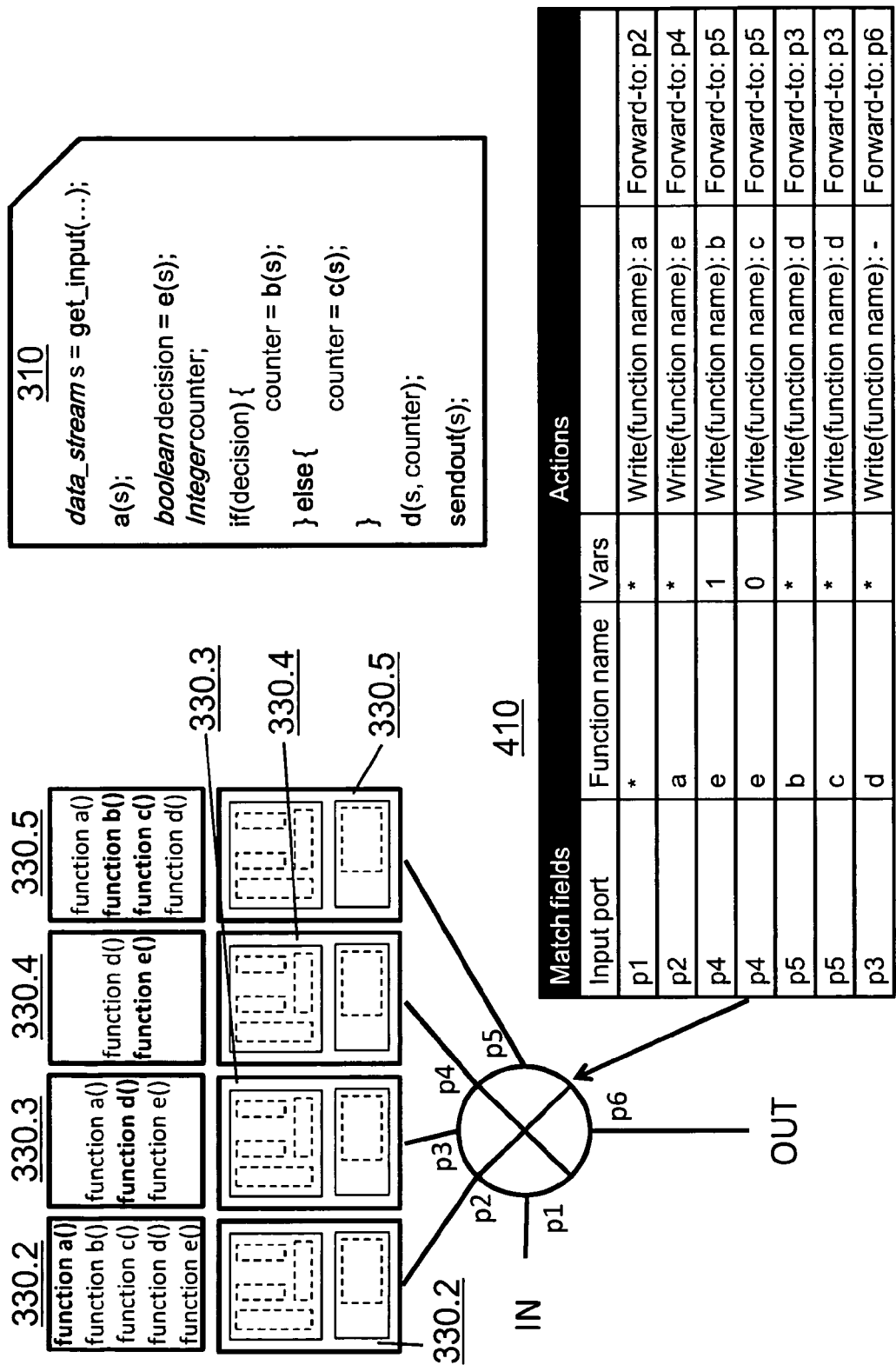


Figure 3
Network centric execution environment: workflow.



INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2015/052877

A. CLASSIFICATION OF SUBJECT MATTER
 INV. H04L12/721 H04L12/717
 ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	YING ZHANG ET AL: "SteERING: A software-defined networking for inline service chaining", 2013 21ST IEEE INTERNATIONAL CONFERENCE ON NETWORK PROTOCOLS (ICNP), IEEE, 7 October 2013 (2013-10-07), pages 1-10, XP032563772, DOI: 10.1109/ICNP.2013.6733615 [retrieved on 2014-02-06]	1,3,5,6, 8,10,11, 13,15
Y	Abstract, Sections I, III-VI ----- -/-	2,4,7,9, 12,14



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

12 May 2015

Date of mailing of the international search report

21/05/2015

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040,
 Fax: (+31-70) 340-3016

Authorized officer

Plata-Andres, Isabel

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2015/052877

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	BOUCADAI R C JACQUENET FRANCE TELECOM R PARKER AFFIRMED NETWORKS D LOPEZ TELEFONICA I+D P YEGANI M: "Differentiated Service Function Chaining Framework; draft-boucadair-network-function-chaining- 03.txt", DIFFERENTIATED SERVICE FUNCTION CHAINING FRAMEWORK; DRAFT-BOUCADAI R-NETWORK-FUNCTION-CHAINING- 03.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARDWORKINGDRAFT, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 21 August 2013 (2013-08-21), pages 1-22, XP015093513, [retrieved on 2013-08-21] abstract, Sections 1, 3 -9 -----	2,4,7,9, 12,14
A	FOSTER N ET AL: "Languages for software-defined networks", IEEE COMMUNICATIONS MAGAZINE, IEEE SERVICE CENTER, PISCATAWAY, US, vol. 51, no. 2, 1 February 2013 (2013-02-01), pages 128-134, XP011505038, ISSN: 0163-6804, DOI: 10.1109/MCOM.2013.6461197 the whole document -----	1-15