US 20170041420A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2017/0041420 A1**

(43) **Pub. Date: Feb. 9, 2017**

Solis

(54) **TRANSFERRING STATE IN CONTENT CENTRIC NETWORK STACKS**

(71) Applicant: **Palo Alto Research Center Incorporated**, Palo Alto, CA (US)

(72) Inventor: **Ignacio Solis**, Scotts Valley, CA (US)

(73) Assignee: **PALO ALTO RESEARCH CENTER INCORPORATED**, Palo Alto, CA (US)

(21) Appl. No.: **14/816,743**

(22) Filed: **Aug. 3, 2015**

Publication Classification

(51) **Int. Cl.**

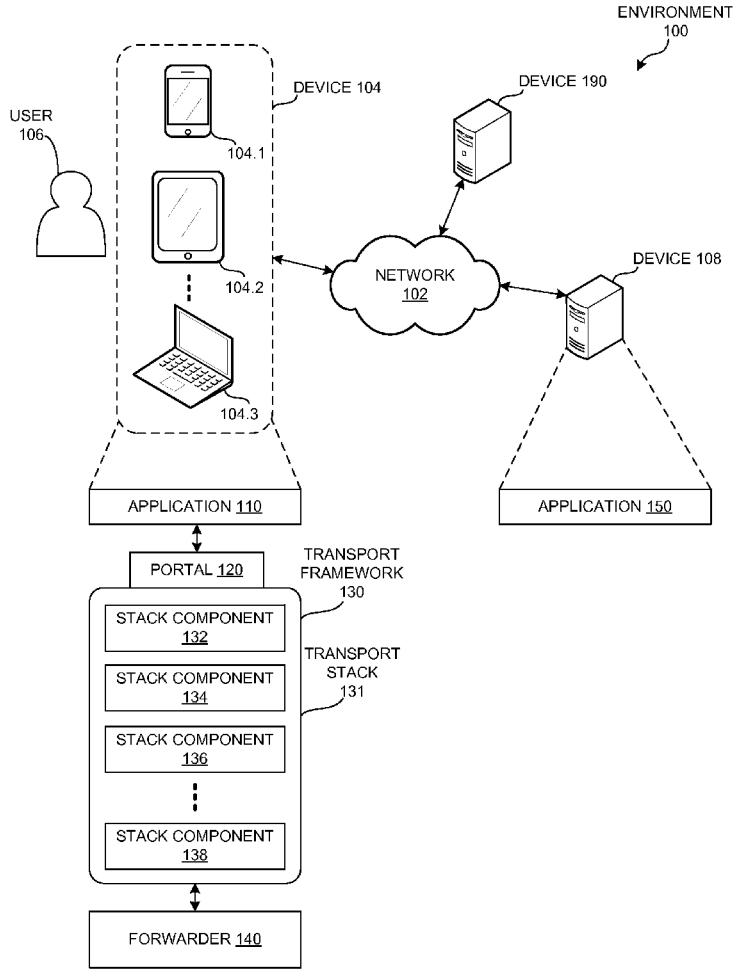
H04L 29/08	(2006.01)
H04L 12/24	(2006.01)
H04L 12/58	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/2842*** (2013.01); ***H04L 51/063*** (2013.01); ***H04L 67/1097*** (2013.01); ***H04L 41/044*** (2013.01)

(57) **ABSTRACT**

One embodiment of the present invention provides a system that facilitates the transfer of the state of a stack in a content centric network. During operation, the system receives, by a communication component from a coordinating entity, a command message to store a current state of the component, wherein the communication component is used in processing messages based on a name, and wherein a name is a hierarchically structured variable length identifier (HSVLI) which comprises contiguous name components ordered from a most general level to a most specific level. The system determines a current state for the communication component. Subsequently, the system stores the current state for the communication component in a data structure.



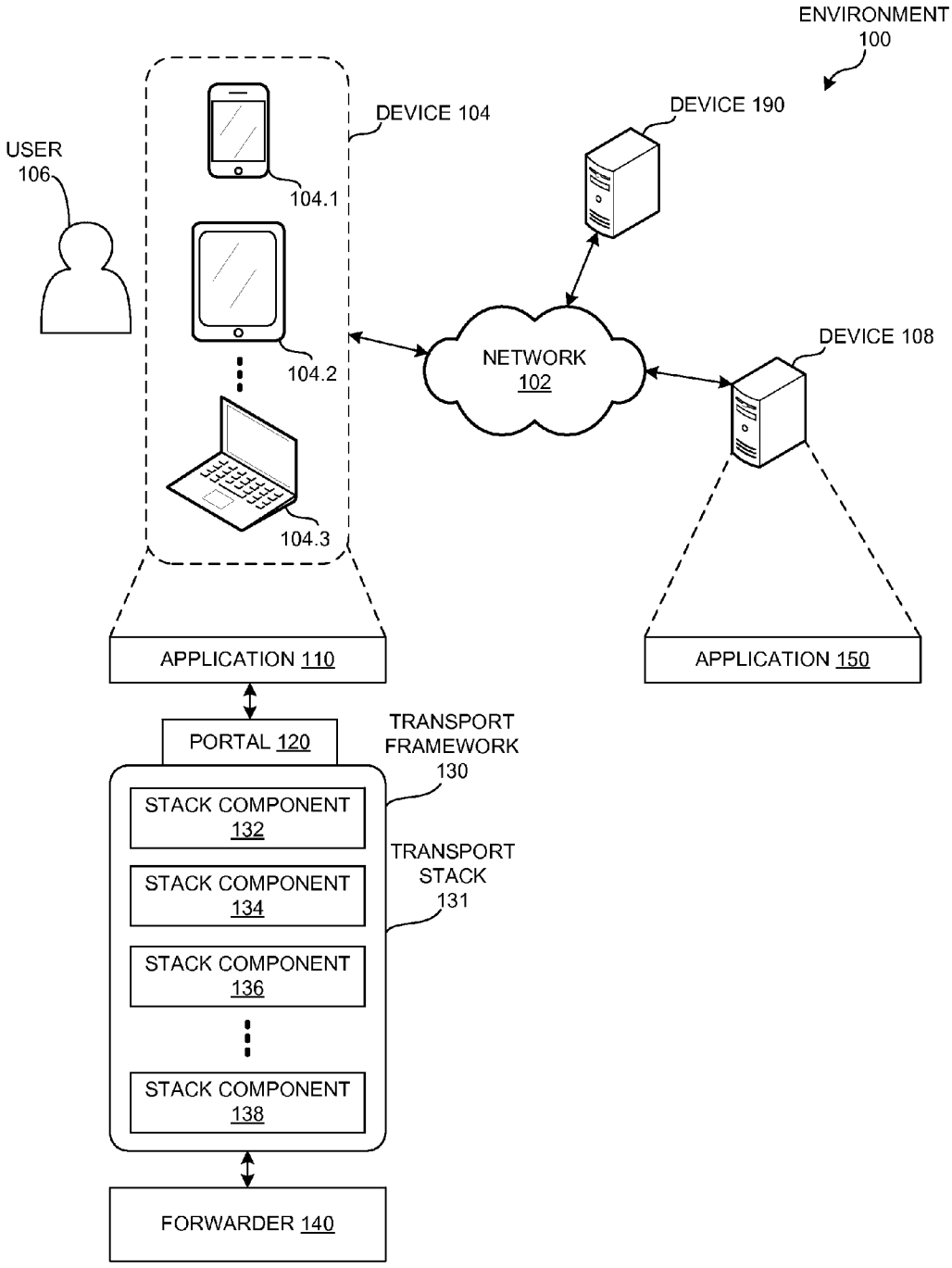

ENVIRONMENT
100
DEVICE 104
DEVICE 190
USER
106
104.1
104.2
104.3
NETWORK
102
DEVICE 108
APPLICATION 110
APPLICATION 150
PORTAL 120
TRANSPORT
FRAMEWORK
130
STACK COMPONENT
132
STACK COMPONENT
134
STACK COMPONENT
136
STACK COMPONENT
138
TRANSPORT
STACK
131
FORWARDER 140

FIG. 1A

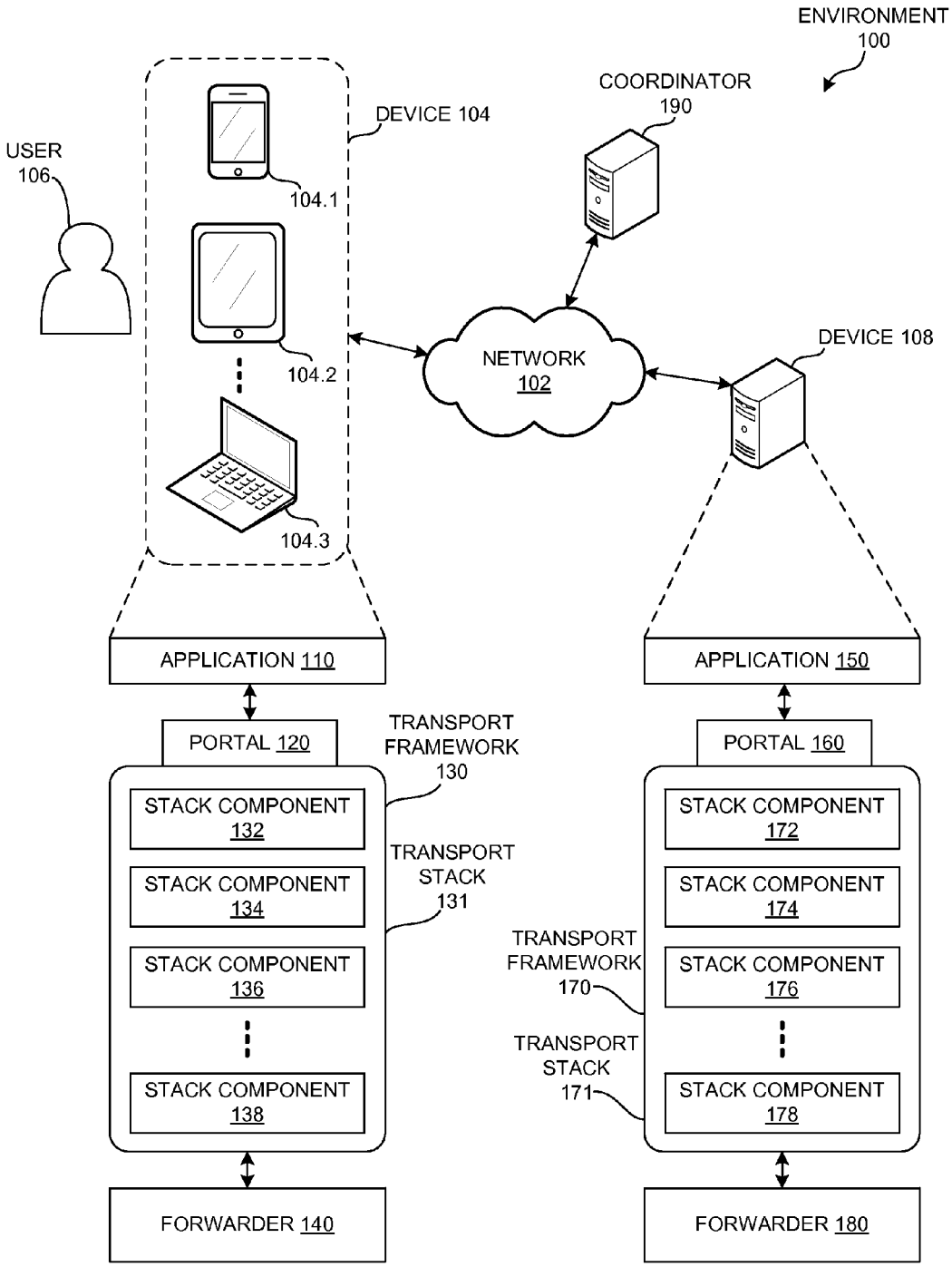

ENVIRONMENT
100
COORDINATOR
190
DEVICE 104
USER
106
104.1
104.2
104.3
NETWORK
102
DEVICE 108
APPLICATION 110
APPLICATION 150
PORTAL 120
TRANSPORT
FRAMEWORK
130
PORTAL 160
STACK COMPONENT
132
STACK COMPONENT
134
STACK COMPONENT
136
STACK COMPONENT
138
TRANSPORT
STACK
131
STACK COMPONENT
172
STACK COMPONENT
174
STACK COMPONENT
176
STACK COMPONENT
178
TRANSPORT
FRAMEWORK
170
TRANSPORT
STACK
171
FORWARDER 140
FORWARDER 180

FIG. 1B

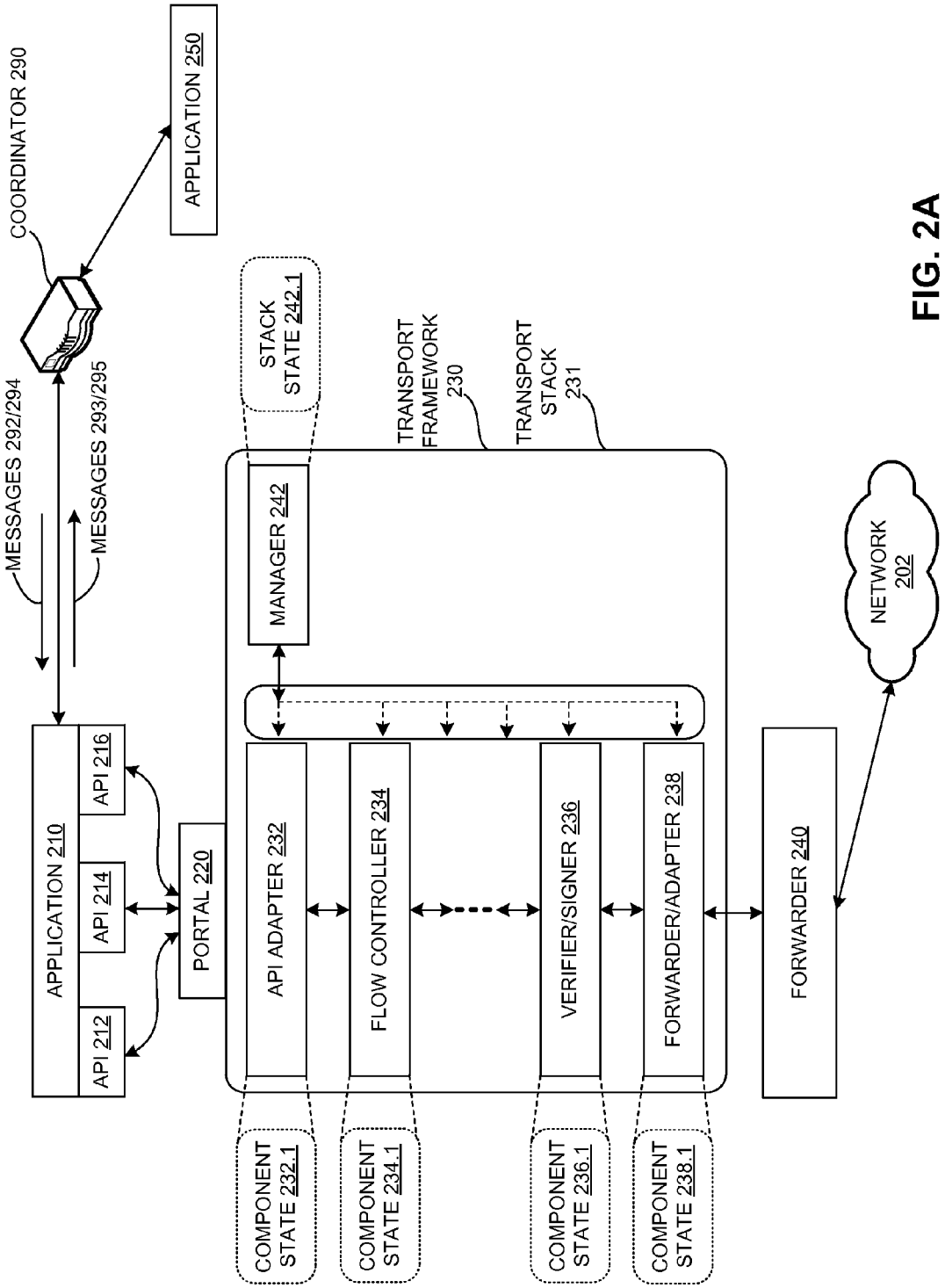

COORDINATOR 290
APPLICATION 250
MESSAGES 292/294
MESSAGES 293/295
STACK STATE 242.1
TRANSPORT FRAMEWORK 230
TRANSPORT STACK 231
MANAGER 242
APPLICATION 210
API 212
API 214
API 216
PORTAL 220
API ADAPTER 232
FLOW CONTROLLER 234
VERIFIER/SIGNER 236
FORWARDER/ADAPTER 238
FORWARDER 240
NETWORK 202
COMPONENT STATE 232.1
COMPONENT STATE 234.1
COMPONENT STATE 236.1
COMPONENT STATE 238.1

FIG. 2A

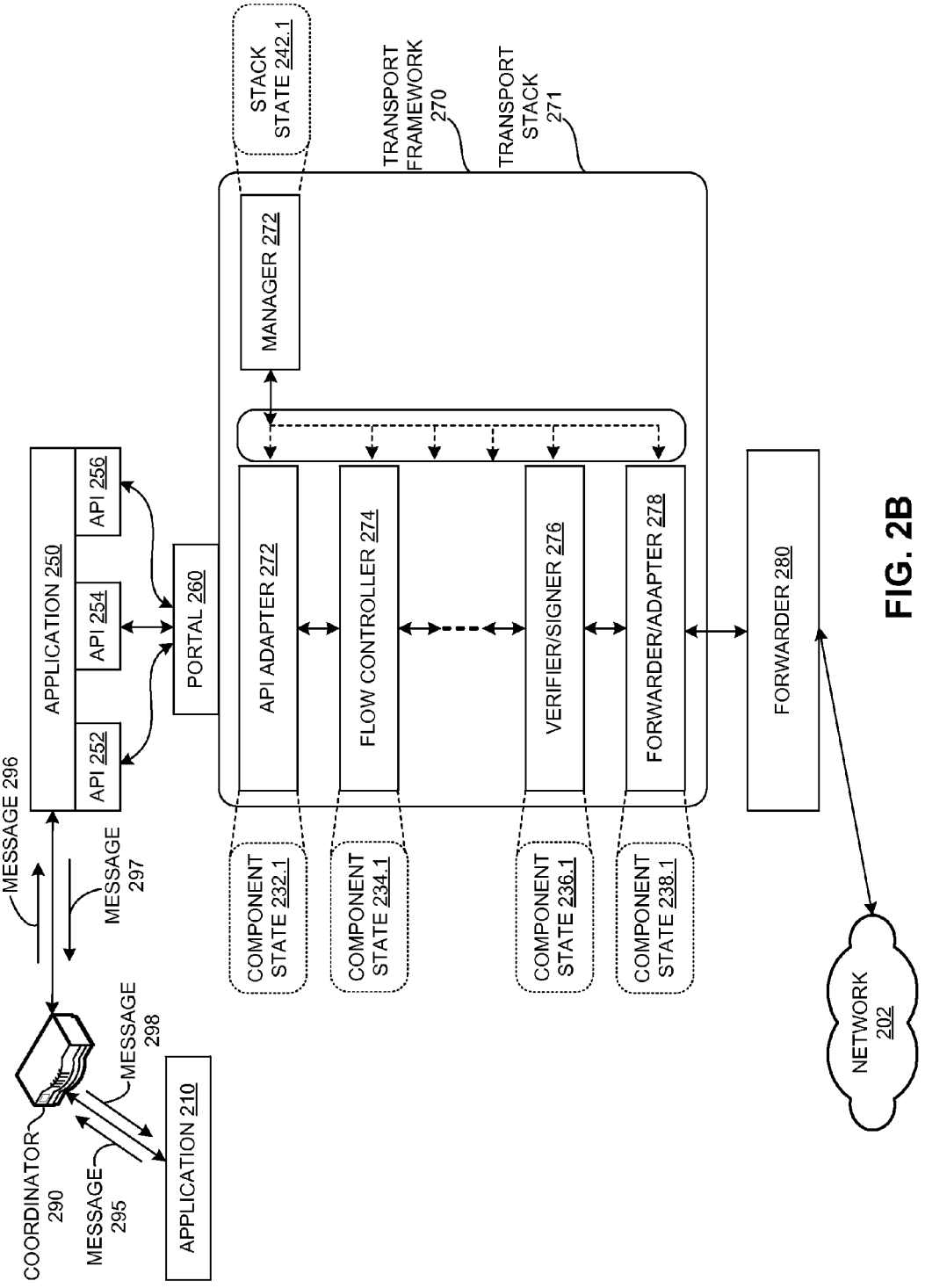

STACK STATE 242.1
TRANSPORT FRAMEWORK 270
TRANSPORT STACK 271
MANAGER 272
APPLICATION 250
API 252
API 254
API 256
PORTAL 260
API ADAPTER 272
FLOW CONTROLLER 274
VERIFIER/SIGNER 276
FORWARDER/ADAPTER 278
FORWARDER 280
COMPONENT STATE 232.1
COMPONENT STATE 234.1
COMPONENT STATE 236.1
COMPONENT STATE 238.1
MESSAGE 296
MESSAGE 297
MESSAGE 298
MESSAGE 295
COORDINATOR 290
APPLICATION 210
NETWORK 202

FIG. 2B

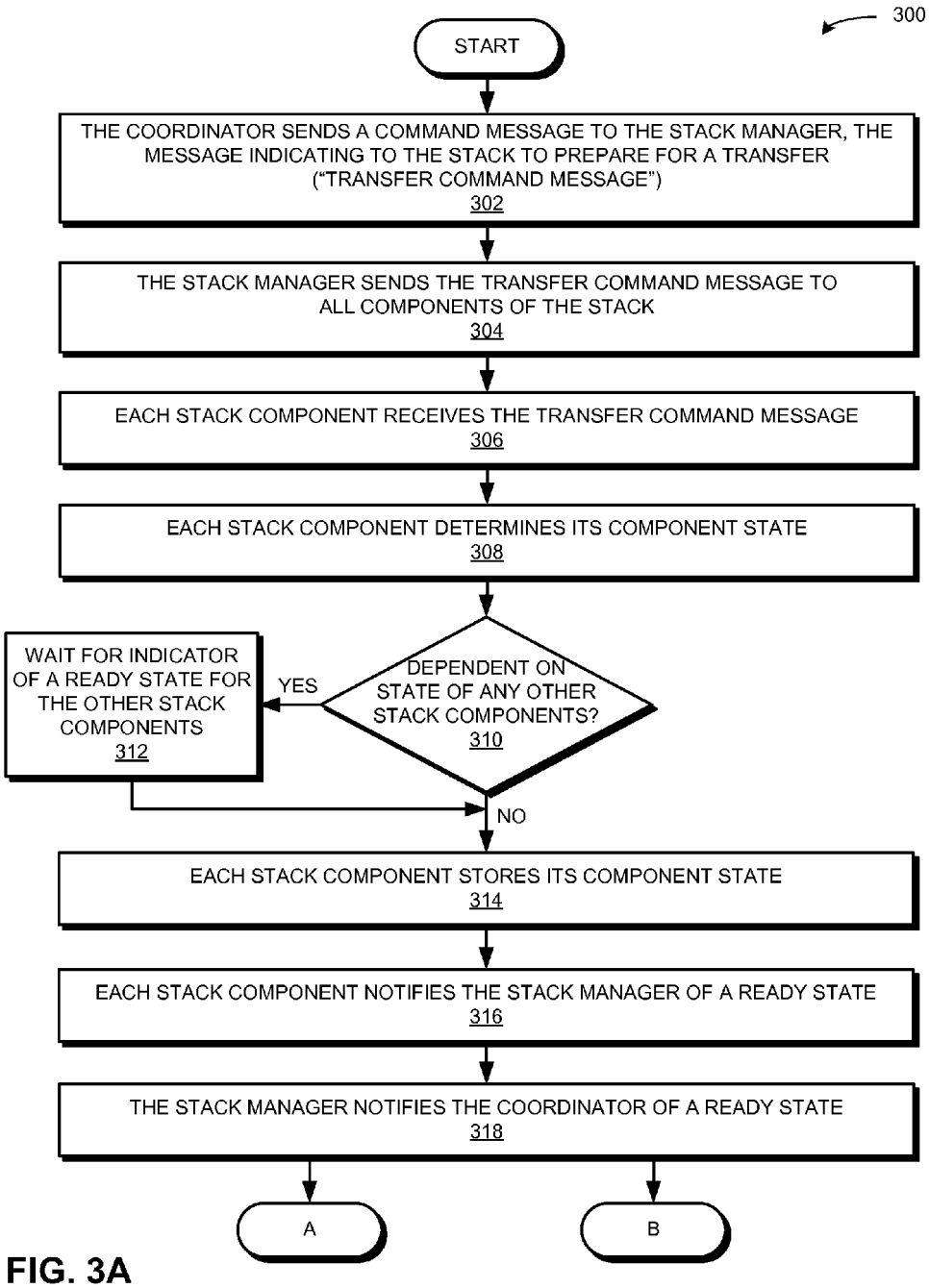

300
START
THE COORDINATOR SENDS A COMMAND MESSAGE TO THE STACK MANAGER, THE MESSAGE INDICATING TO THE STACK TO PREPARE FOR A TRANSFER ("TRANSFER COMMAND MESSAGE")
302
THE STACK MANAGER SENDS THE TRANSFER COMMAND MESSAGE TO ALL COMPONENTS OF THE STACK
304
EACH STACK COMPONENT RECEIVES THE TRANSFER COMMAND MESSAGE
306
EACH STACK COMPONENT DETERMINES ITS COMPONENT STATE
308
DEPENDENT ON STATE OF ANY OTHER STACK COMPONENTS?
310
YES
WAIT FOR INDICATOR OF A READY STATE FOR THE OTHER STACK COMPONENTS
312
NO
EACH STACK COMPONENT STORES ITS COMPONENT STATE
314
EACH STACK COMPONENT NOTIFIES THE STACK MANAGER OF A READY STATE
316
THE STACK MANAGER NOTIFIES THE COORDINATOR OF A READY STATE
318
A
B

FIG. 3A

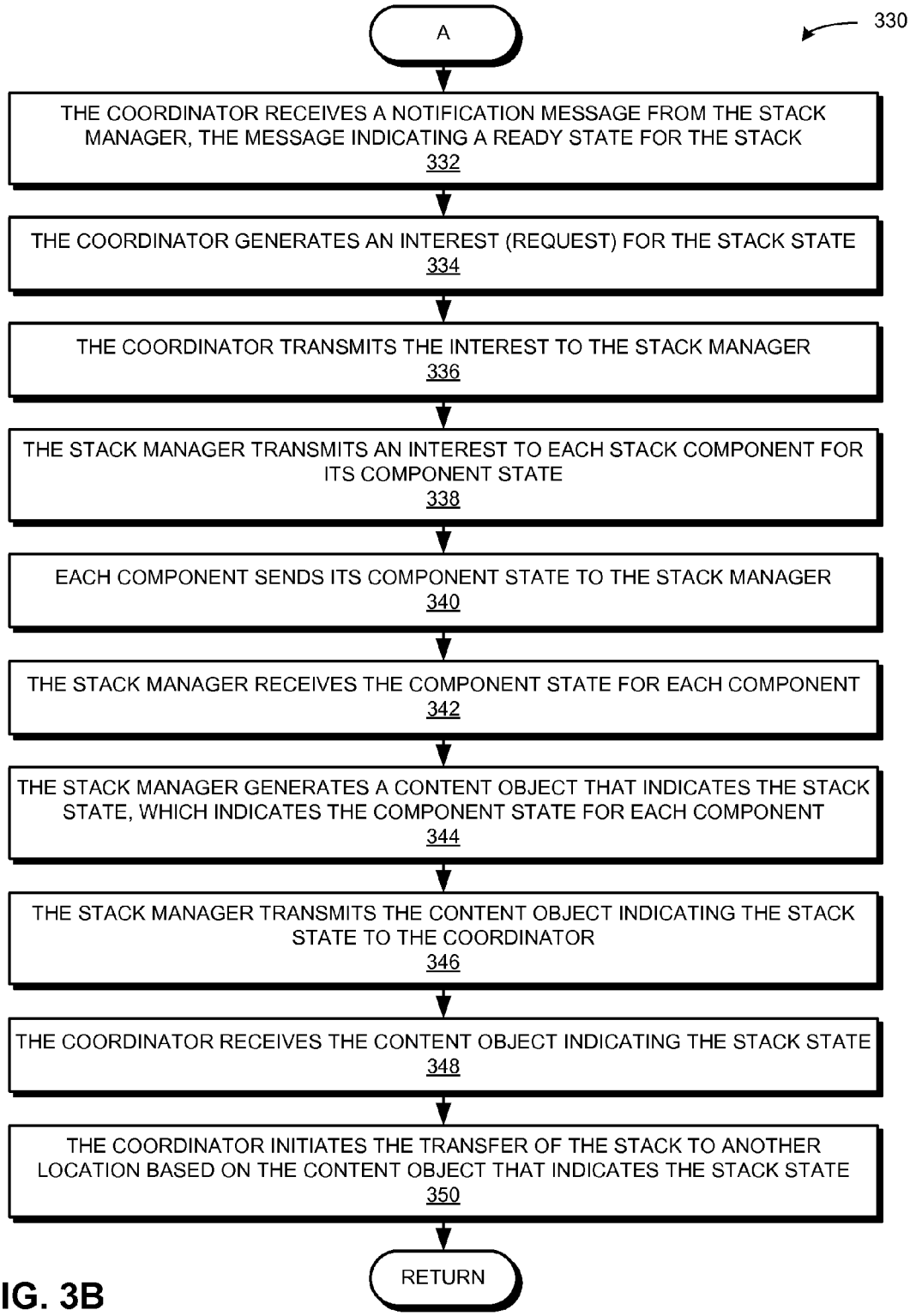

330
A
THE COORDINATOR RECEIVES A NOTIFICATION MESSAGE FROM THE STACK MANAGER, THE MESSAGE INDICATING A READY STATE FOR THE STACK
332
THE COORDINATOR GENERATES AN INTEREST (REQUEST) FOR THE STACK STATE
334
THE COORDINATOR TRANSMITS THE INTEREST TO THE STACK MANAGER
336
THE STACK MANAGER TRANSMITS AN INTEREST TO EACH STACK COMPONENT FOR ITS COMPONENT STATE
338
EACH COMPONENT SENDS ITS COMPONENT STATE TO THE STACK MANAGER
340
THE STACK MANAGER RECEIVES THE COMPONENT STATE FOR EACH COMPONENT
342
THE STACK MANAGER GENERATES A CONTENT OBJECT THAT INDICATES THE STACK STATE, WHICH INDICATES THE COMPONENT STATE FOR EACH COMPONENT
344
THE STACK MANAGER TRANSMITS THE CONTENT OBJECT INDICATING THE STACK STATE TO THE COORDINATOR
346
THE COORDINATOR RECEIVES THE CONTENT OBJECT INDICATING THE STACK STATE
348
THE COORDINATOR INITIATES THE TRANSFER OF THE STACK TO ANOTHER LOCATION BASED ON THE CONTENT OBJECT THAT INDICATES THE STACK STATE
350
RETURN

FIG. 3B

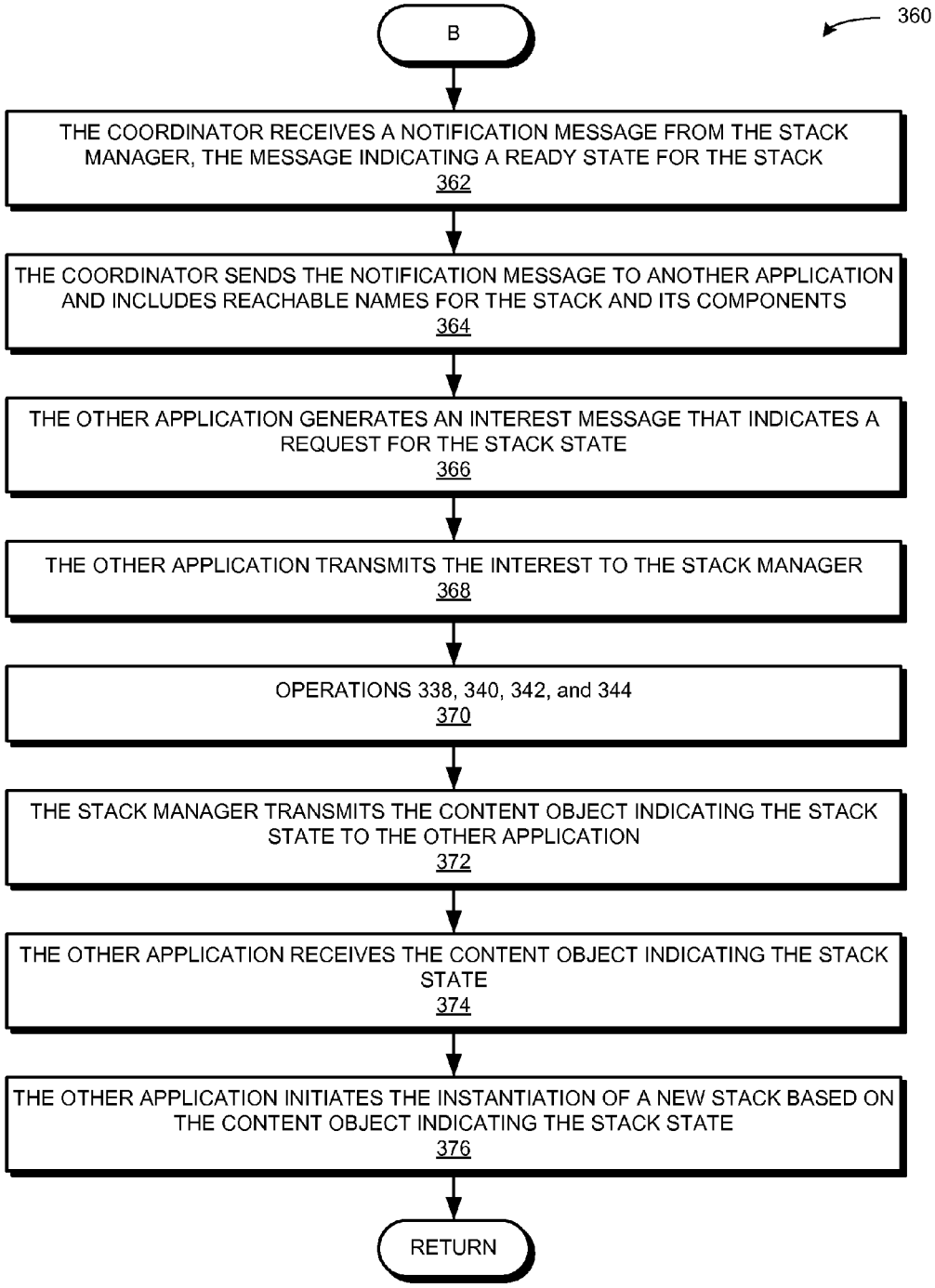

360
B
THE COORDINATOR RECEIVES A NOTIFICATION MESSAGE FROM THE STACK MANAGER, THE MESSAGE INDICATING A READY STATE FOR THE STACK
362
THE COORDINATOR SENDS THE NOTIFICATION MESSAGE TO ANOTHER APPLICATION AND INCLUDES REACHABLE NAMES FOR THE STACK AND ITS COMPONENTS
364
THE OTHER APPLICATION GENERATES AN INTEREST MESSAGE THAT INDICATES A REQUEST FOR THE STACK STATE
366
THE OTHER APPLICATION TRANSMITS THE INTEREST TO THE STACK MANAGER
368
OPERATIONS 338, 340, 342, and 344
370
THE STACK MANAGER TRANSMITS THE CONTENT OBJECT INDICATING THE STACK STATE TO THE OTHER APPLICATION
372
THE OTHER APPLICATION RECEIVES THE CONTENT OBJECT INDICATING THE STACK STATE
374
THE OTHER APPLICATION INITIATES THE INSTANTIATION OF A NEW STACK BASED ON THE CONTENT OBJECT INDICATING THE STACK STATE
376
RETURN

FIG. 3C

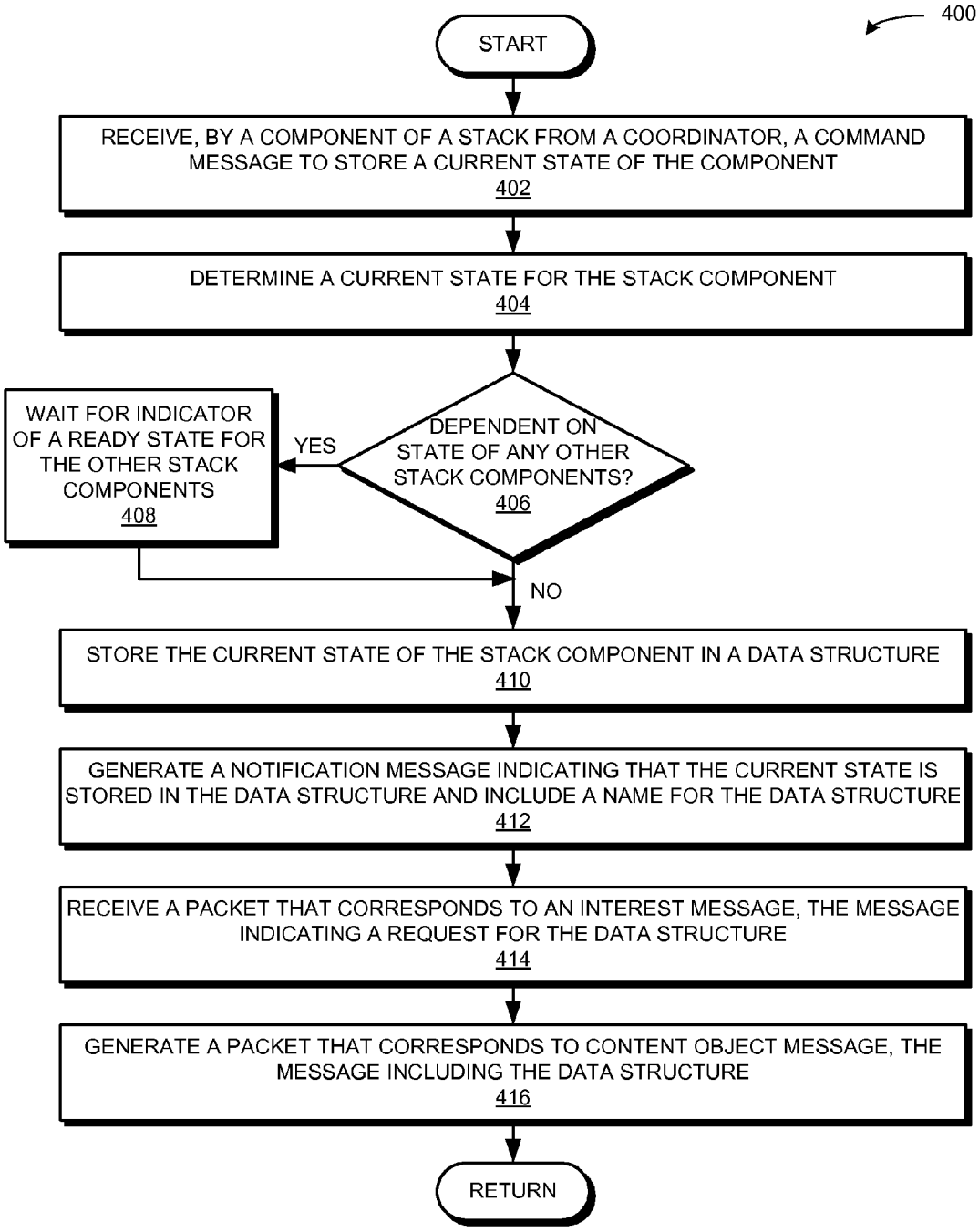

400
START
RECEIVE, BY A COMPONENT OF A STACK FROM A COORDINATOR, A COMMAND MESSAGE TO STORE A CURRENT STATE OF THE COMPONENT
402
DETERMINE A CURRENT STATE FOR THE STACK COMPONENT
404
DEPENDENT ON STATE OF ANY OTHER STACK COMPONENTS?
406
YES
WAIT FOR INDICATOR OF A READY STATE FOR THE OTHER STACK COMPONENTS
408
NO
STORE THE CURRENT STATE OF THE STACK COMPONENT IN A DATA STRUCTURE
410
GENERATE A NOTIFICATION MESSAGE INDICATING THAT THE CURRENT STATE IS STORED IN THE DATA STRUCTURE AND INCLUDE A NAME FOR THE DATA STRUCTURE
412
RECEIVE A PACKET THAT CORRESPONDS TO AN INTEREST MESSAGE, THE MESSAGE INDICATING A REQUEST FOR THE DATA STRUCTURE
414
GENERATE A PACKET THAT CORRESPONDS TO CONTENT OBJECT MESSAGE, THE MESSAGE INCLUDING THE DATA STRUCTURE
416
RETURN

FIG. 4

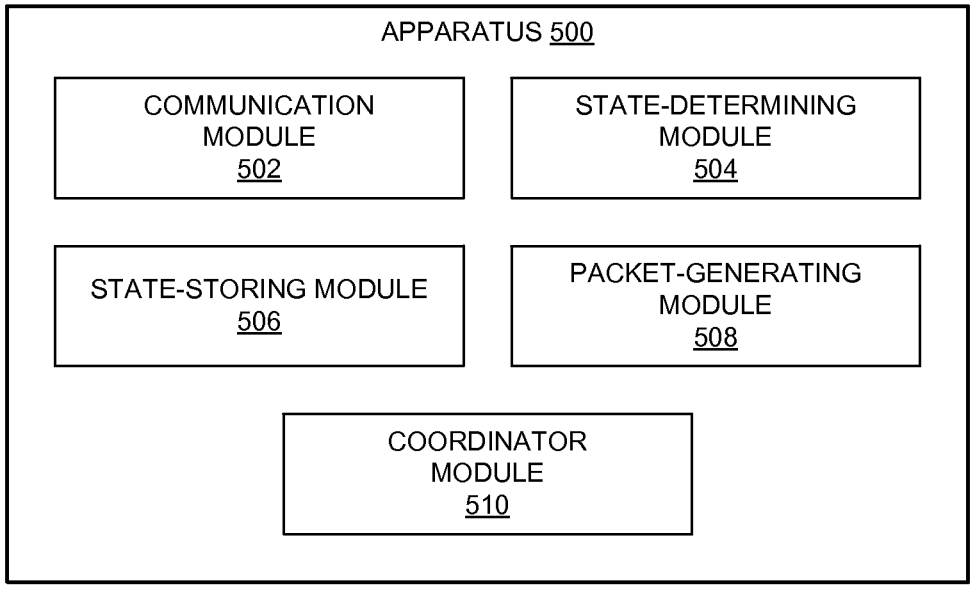

APPARATUS 500
COMMUNICATION MODULE 502
STATE-DETERMINING MODULE 504
STATE-STORING MODULE 506
PACKET-GENERATING MODULE 508
COORDINATOR MODULE 510

FIG. 5

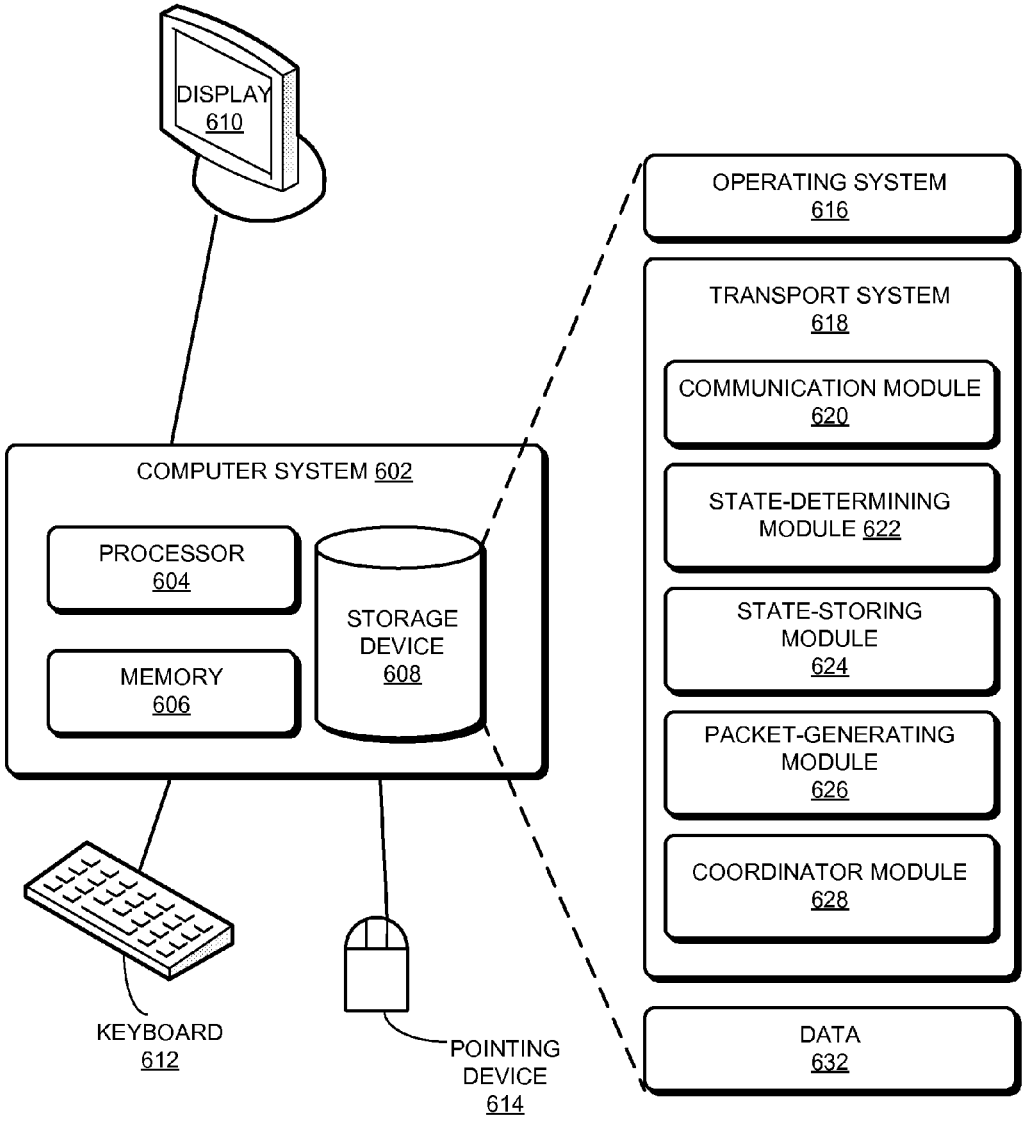

DISPLAY
610
OPERATING SYSTEM
616
TRANSPORT SYSTEM
618
COMMUNICATION MODULE
620
STATE-DETERMINING
MODULE 622
STATE-STORING
MODULE
624
PACKET-GENERATING
MODULE
626
COORDINATOR MODULE
628
COMPUTER SYSTEM 602
PROCESSOR
604
MEMORY
606
STORAGE
DEVICE
608
KEYBOARD
612
POINTING
DEVICE
614
DATA
632

FIG. 6

TRANSFERRING STATE IN CONTENT CENTRIC NETWORK STACKS

RELATED APPLICATIONS

[0001] The subject matter of this application is related to the subject matter in the following applications:

[0002] U.S. patent application Ser. No. 13/847,814 (Attorney Docket No. PARC-20120537-US-NP), entitled "ORDERED-ELEMENT NAMING FOR NAME-BASED PACKET FORWARDING," by inventor Ignacio Solis, filed 20 Mar. 2013 (hereinafter "U.S. patent application Ser. No. 13/847,814");

[0003] U.S. patent application Ser. No. 12/338,175 (Attorney Docket No. PARC-20080626-US-NP), entitled "CONTROLLING THE SPREAD OF INTERESTS AND CONTENT IN A CONTENT CENTRIC NETWORK," by inventors Van L. Jacobson and Diana K. Smetters, filed 18 Dec. 2008 (hereinafter "U.S. patent application Ser. No. 12/338,175"); and

[0004] U.S. patent application Ser. No. 14/595,060 (Attorney Docket No. PARC-20141111US01), entitled "AUTO-CONFIGURABLE TRANSPORT STACK," by inventors Ignacio Solis and Glenn C. Scott, filed 12 Jan. 2015 (hereinafter "U.S. patent application Ser. No. 14/595,060");

[0005] U.S. patent application Ser. No. 14/746,490 (Attorney Docket No. PARC-20141532US01), entitled "TRANSPORT STACK NAME SCHEME AND IDENTITY MANAGEMENT," by inventors Christopher A. Wood and Glenn C. Scott, filed 22 Jun. 2015 (hereinafter "U.S. patent application Ser. No. 14/746,490");

[0006] U.S. patent application Ser. No. 14/749,349 (Attorney Docket No. PARC-20141531US01), entitled "FLEXIBLE COMMAND AND CONTROL IN CONTENT CENTRIC NETWORKS," by inventors Christopher A. Wood and Glenn C. Scott, filed 24 Jun. 2015 (hereinafter "U.S. patent application Ser. No. 14/749,349"); and

[0007] U.S. patent application Ser. No. 14/231,515 (Attorney Docket No. PARC-20140190US01), entitled "AGGREGATE SIGNING OF DATA IN CONTENT CENTRIC NETWORKING," by inventors Ersin Uzun, Marc E. Mosko, Michael F. Plass, and Glenn C. Scott, filed 31 Mar. 2014 (hereinafter "U.S. patent application Ser. No. 14/231,515"); the disclosures of which are herein incorporated by reference in their entirety.

BACKGROUND

[0008] Field

[0009] This disclosure is generally related to a transport framework. More specifically, this disclosure is related to a system and method for transferring or duplicating the state of a stack in a content centric network.

[0010] Related Art

[0011] The ubiquitous nature of mobile computing devices and the Internet is making it possible for people to experience digital content from anywhere. People can use applications in their mobile computing devices to consume or interact with content from service providers across the Internet, such as to stream movies or music or to play games with others. These advances in mobile computing are also increasing the quality of content that can be reproduced by these mobile devices and greatly increases the number of devices that can generate and capture digital content and share with others over the Internet. Nowadays, even small mobile devices such as smartphones can produce full high-definition video with high-quality color reproduction, and high-speed cellular and broadband networks make it possible for users to share this content with others over various Internet services, such as the YouTube (from Google, Inc.) and Facebook (from Facebook, Inc.) content-sharing services.

[0012] Many computer applications leverage these computer networks and Internet services to provide social features to its users, which greatly enhances the user experience. When an application wants to use the network, it does so by using one or more Application Programming Interfaces (APIs) that run on the computing device's operating system. These APIs provide a way for applications to send, receive, store, configure data or otherwise communicate with other computers across the network.

[0013] For example, an application instantiates a protocol stack that implements a network API before the application can use the API to send or receive data over the network. In a traditional protocol stack based on, e.g., the Open Systems Interconnection (OSI) model, each layer can only communicate with the layer above or below it. In a model based on a content-centric network (CCN), a protocol stack can be dynamically created to suit the needs of APIs used by various applications. While the creation of these application-driven protocol stacks can increase the flexibility of a system, other requirements (e.g., failover, load-balancing, and other network-related needs) may result in the need to move, transfer, or duplicate a stack by transferring the stack state from one location to another.

SUMMARY

[0014] One embodiment provides a transport framework system that facilitates transferring the state of a stack in a CCN. During operation, the system receives, by a communication component from a coordinating entity, a command message to store a current state of the component, wherein the communication component is used in processing messages based on a name, and wherein a name is a hierarchically structured variable length identifier (HSVLI) which comprises contiguous name components ordered from a most general level to a most specific level. The system determines a current state for the communication component.

[0015] The system then stores the current state for the communication component in a data structure.

[0016] In some embodiments, the system receives a first packet that corresponds to an interest message, wherein the interest message indicates a request for the data structure and includes a name for the data structure.

[0017] In some embodiments, the first packet is received from one or more of: a manager component of a stack of communication modules, wherein a manager component is used in processing messages between the coordinating entity and the communication modules, wherein a stack does not require a respective communication module to communicate only with a layer above or below thereof, and wherein the communication component belongs to the stack; and the coordinating entity, which is one or more of: an entity or a service external to the stack; and an application associated with the stack.

[0018] In some embodiments, the system generates a second packet that corresponds to a content object message, wherein the content object message includes the data structure and a name for the data structure.

[0019] In some embodiments, the system generates a notification message indicating that the current state is stored in the data structure, wherein the notification message includes a name for the data structure.

[0020] In some embodiments, the data structure indicates one or more interest or content object messages stored by the communication component in processing the messages.

[0021] In some embodiments, the data structure is a manifest, wherein a manifest indicates a set of content objects and their corresponding digests, wherein a respective content object is a data object or another manifest, and wherein a manifest and a content object each indicate a name.

[0022] In some embodiments, responsive to receiving a request for the current state, the system transmits the current state.

[0023] In a further variation, the communication component is a component of a stack of communication modules, wherein the stack does not require a respective communication module to communicate only with a layer above or below thereof.

[0024] In some embodiments, the system determines a current state for one or more other communication components of the stack. Responsive to determining that the current state for the other communication components is a pending state, the system waits for an indicator of a ready state for the other communication components.

[0025] In some embodiments, the coordinating entity is one or more of: an entity or a service external to the stack; and an application associated with the stack.

[0026] In some embodiments, the command message is received from the coordinating entity via a manager component of the stack, wherein a manager component is used in processing messages between the coordinating entity and the communication modules.

BRIEF DESCRIPTION OF THE FIGURES

[0027] FIG. **1**A illustrates an exemplary environment which facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention.

[0028] FIG. **1**B illustrates an exemplary environment which facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention.

[0029] FIG. **2**A illustrates an exemplary transport framework and transport stack, in accordance with an embodiment of the present invention.

[0030] FIG. **2**B illustrates an exemplary transport framework and transport stack, corresponding to FIG. **2**A, in accordance with an embodiment of the present invention.

[0031] FIG. **3**A presents a flow chart illustrating a method for transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention.

[0032] FIG. **3**B presents a flow chart illustrating a method for transferring the state of a stack in a content centric network, where the coordinator communicates with the stack via the stack manager, in accordance with an embodiment of the present invention.

[0033] FIG. **3**C presents a flow chart illustrating a method for transferring the state of a stack in a content centric network, where the coordinator communicates with another application, in accordance with an embodiment of the present invention.

[0034] FIG. **4** presents a flow chart illustrating a method by a stack component for facilitating the transfer of a stack, in accordance with an embodiment of the present invention.

[0035] FIG. **5** illustrates an exemplary apparatus that facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention.

[0036] FIG. **6** illustrates an exemplary computer system that facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention.

[0037] In the figures, like reference numerals refer to the same figure elements.

DETAILED DESCRIPTION

[0038] The following description is presented to enable any person skilled in the art to make and use the embodiments, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present disclosure. Thus, the present invention is not limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

Overview

[0039] Embodiments of the present invention provide a transport framework system that facilitates transferring the state of a transport stack in a content centric network (CCN). In CCN, the transport framework enables high-level APIs to instantiate one or more transport stacks within the framework. A transport stack can include multiple components or communication modules, and does not adhere to a traditional layered model (e.g., OSI) where each component communicates only with the component below or above it. A transport stack can be created dynamically and configured at runtime, where each component within the transport stack performs a specific function. For example, one component of a transport stack can be a verifier component which is responsible for verifying the digital signature of content objects received in response to an interest sent over the network. CCN transport stacks are configurable and extensible, as described in U.S. patent application Ser. No. 14/595,060, which is herein incorporated by reference.

[0040] An application can initiate a communication over the network by issuing a call to an API. During the communication, a CCN transport stack and each component of the stack updates its corresponding state. For example, a flow controller component may maintain a running window size for 30 outstanding interests, and a list of outstanding or pending interests in a queue for issuance. A verifier component may store a set of keys for current use and a set of rotating keys for future use. In the case of failover, load-balancing, or other requirements for resource distribution, the application may wish to instantiate the stack in a different location. To do this, the system transfers the stack

state from one transport framework to another transport framework (e.g., from one device to another device residing in a different physical location). The system can extract the stack state from the stack and instantiate another stack at a different location using the extracted stack state, creating a newly "transferred" or "duplicated" stack.

[0041] Specifically, an external coordinating entity ("coordinator") can trigger the process of transferring a stack. The coordinator informs the stack (via, e.g., a stack manager component) to prepare for a transfer, and the stack in turn informs each component to prepare for the transfer. Each component stores its state, and informs the stack of its ready state. When all components have informed the stack of its ready state, the stack (e.g., the stack manager component) in turn informs the coordinator of a ready state. Subsequently, the coordinator can request the stack state from the stack and send the stack state to another application for transfer (e.g., instantiation). Alternatively, the coordinator can send a ready message to another application, allowing the other application to request the stack state directly from the stack, without further involvement from the coordinator. The coordinator can also provide to the other application unique names for the stack, the stack components, and their respective states. A transport stack name scheme and identity management is described in U.S. patent application Ser. No. 14/746,490, which is herein incorporated by reference.

[0042] In some embodiments, the transport framework operates under the CCN architecture. In CCN, each piece of content is individually named, and each piece of data is bound to a unique name that distinguishes the data from any other piece of data, such as other versions of the same data or data from other sources. This unique name allows a network device to request the data by disseminating a request or an interest that indicates the unique name, and can obtain the data independent from the data's storage location, network location, application, and means of transportation. The following terms are used to describe the CCN architecture:

[0043] Content Object (or "content object"): A single piece of named data, which is bound to a unique name. Content Objects are "persistent," which means that a Content Object can move around within a computing device, or across different computing devices, but does not change. If any component of the Content Object changes, the entity that made the change creates a new Content Object that includes the updated content, and binds the new Content Object to a new unique name.

[0044] Unique Names: A name in a CCN is typically location independent and uniquely identifies a Content Object. A data-forwarding device can use the name or name prefix to forward a packet toward a network node that generates or stores the Content Object, regardless of a network address or physical location for the Content Object. In some embodiments, the name may be a hierarchically structured variable-length identifier (HSVLI). The HSVLI can be divided into several hierarchical components, which can be structured in various ways. For example, the individual name components parc, home, ccn, and test.txt can be structured in a left-oriented prefix-major fashion to form the name "/parc/home/ccn/test.txt." Thus, the name "/parc/home/ccn" can be a "parent" or "prefix" of "/parc/home/ccn/test.txt." Additional components can be used to distinguish between different versions of the content item, such as a collaborative document.

[0045] In some embodiments, the name can include an identifier, such as a hash value that is derived from the Content Object's data (e.g., a checksum value) and/or from elements of the Content Object's name. A description of a hash-based name is described in U.S. patent application Ser. No. 13/847,814, which is herein incorporated by reference. A name can also be a flat label. Hereinafter, "name" is used to refer to any name for a piece of data in a name-data network, such as a hierarchical name or name prefix, a flat name, a fixed-length name, an arbitrary-length name, or a label (e.g., a Multiprotocol Label Switching (MPLS) label).

[0046] Interest (or "interest"): A packet that indicates a request for a piece of data, and includes a name (or a name prefix) for the piece of data. A data consumer can disseminate a request or Interest across an information-centric network, which CCN/NDN routers can propagate toward a storage device (e.g., a cache server) or a data producer that can provide the requested data to satisfy the request or Interest.

[0047] The methods disclosed herein are not limited to CCN networks and are applicable to other architectures as well. A description of a CCN architecture is described in U.S. patent application Ser. No. 12/338,175, which is herein incorporated by reference.

Exemplary Network and Communication

[0048] FIG. **1**A illustrates an exemplary environment **100** which facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention. Computing environment **100** can include a computer network **102**, such as a CCN. Environment **100** can also include a user **106** associated with a local computing device **104**, and remote computing devices **108** and **190**. Device **104** can have an internal transport stack **131** associated with transport framework **130**. In a traditional IP architecture, a forwarder is an IP-based forwarder that looks at the header of a packet to determine the source and the destination for the packet, and forwards the packet to the destination. The stack performs TCP/UDP, and an application interacts with the stack via a socket. In contrast, device **104** of the present invention does not use a conventional "stack." Rather, device **104** via application **110** can request a portal API instance corresponding to a portal **120** which corresponds to transport framework **130**. Application **110** can generate a request to retrieve or create the portal API instance associated with portal **120**. Portal instance creation is described in U.S. patent application Ser. No. 14/746,490, which is herein incorporated by reference.

[0049] Device **104** can include any computing device coupled to network **102**, such as a smartphone **104.1**, a tablet computer **104.2**, and/or a server or personal computer **104.3**. Specifically, device **104** can include application **110** which communicates via portal **120** with transport framework **130**. Transport framework **130** can include stack components **132**, **134**, **136**, and **138**. Device **104** can also include a forwarder **140** (e.g., a network interface card, or a router in a local area network) which can transfer packets between a stack (and individual stack components) of transport framework **130** and network **102**. Devices **108** and **190** can include any computing device coupled to network **102**, such as a server or an end host device. Device **108** can further include an application **150**, and device **190** can further

include any computing device with coordinating functionality to determine and initiate a stack transfer, as described herein.

[0050] FIG. **1**B illustrates exemplary environment **100** which facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention. During operation, device or "coordinator" **190** can issue a notification message to transport stack **131** and stack components **132-138** to prepare for a move. Stack components **132-138** store their respective states, and inform stack **131** of a ready state. Stack **131** in turn informs coordinator **190** of a ready state, and coordinator **190** can then request the state of the stack ("stack state") from stack **131**. The stack state can be an object which indicates the individual states as stored by each stack component, described below in relation to FIG. **2**A. Upon receiving the stack state from stack **131**, coordinator **190** can transfer the stack state by sending the stack state to device **108**, which can instantiate, via application **150**, a transport stack **171** of a transport framework **170**. Transport stack **171** can be instantiated based on the stack state of stack **131** received from coordinator **190**. Transport stack **171** can include stack components **172**, **174**, **176**, and **178**, which have the same state as stack components **132-138** of stack **131**. Thus, coordinator **190** initiates and handles the transfer of the state of stack **131** from framework **130** to framework **170**. While the transfer of the stack state can be performed for failover reasons (and thus result in the destruction of stack **131**), the transfer can also be performed for active load balancing purposes (and thus may leave stack **131** intact). In some embodiments, upon instantiating stack **171**, application **150** of device **108** may send a message to coordinator **190** indicating completion of the stack transfer. Coordinator **190** in turn may transmit to application **110** the message indicating completion of the stack transfer and can also include a command message to resume operation of the stack.

Exemplary Transport Framework

[0051] FIG. **2**A illustrates an exemplary transport framework **230** and a transport stack **231**, in accordance with an embodiment of the present invention. A coordinator **290** can communicate with an application **210**, which can reside on a node or device that communicates over a network **202**. Application **210** can use APIs **212**, **214**, and **216** to communicate over network **202**, and APIs **212-216** can interact via a portal **220** with a transport framework **230**. Transport framework **230** can include one or more transport stacks which each include multiple stack components or communication modules. In FIG. **2**A, transport framework **230** depicts one transport stack (e.g., transport stack **231**) which includes stack components **232**, **234**, **236**, **238**, and **242**. An API adapter **232** can communicate between an API and a specific transport stack and transport framework **230**. A flow controller **234** can shape and manage traffic, pipeline and transmit interests, and order content objects. A verifier/signer **236** can encode and sign content objects destined for a network element, decode and verify content objects destined for an associated application, encode interests destined for a network element, and decode interests destined for an associated application. A forwarder/adapter **238** can communicate with a forwarder **240**. Forwarder **240** can communicate with other forwarders over network **202**. Other stack components (not shown) can include functionality related to security (e.g., encryption, decryption, authentication, data signing, signature verification, trust assessment, and filtering), data-processing (e.g., encoding, decoding, encapsulating, decapsulating, transcoding, compression, extraction, and decompression), and storage (e.g., data storage, data retrieval from storage, deduplication, segmentation, and versioning).

[0052] Coordinator **290** can send a message **292** to stack **231** via application **210** and one of APIs **212-216** and portal **220**. Message **292** can be a command message informing the stack to prepare for a transfer ("transfer command message"). A stack manager component **242** can receive and transmit the transfer command message to stack components **232-238**, each of which determine and store its state in a data structure. For example, flow controller **234** can determine and store its state in a component state **234.1** data object. Upon storing its state, each component can then send a notification message to manager **242** of its ready state ("component ready message"), and, upon receiving the component ready message from each stack component, manager **242** can send a notification message **293** to coordinator **290** indicating that stack **231** is in a ready state ("stack ready message"). In some embodiments, manager **242** can assemble a stack state **242.1** content object which indicates each of the data objects for component states **232.1**, **234.1**, **236.1**, and **238.1**. Manager **242**, like the other stack components, can determine and store its own stack state (not shown), which can also be indicated in stack state **242.1**. Subsequently, coordinator **290** can send an interest message **294** to stack **231** requesting stack state **242.1** from manager **242**. Application **210** can return a content object message **295** to coordinator **290** indicating stack state **242.1** ("stack state message").

[0053] FIG. **2**B illustrates an exemplary transport framework **270** and a transport stack **271**, corresponding to FIG. **2**A, in accordance with an embodiment of the present invention. Upon receiving stack state **242.1** via stack state message **295**, coordinator **290** can continue the transfer by sending a command message **296** that includes stack state **242.1** to application **250**, where command message **296** includes a command to instantiate a new stack based on stack state **242.1**. Application **250** can instantiate transport stack **271** in transport framework **270** (via one of APIs **252-256**, a portal **260**, and, in some embodiments, a manager **272**), such that each component of stack **271** has the same component state **232.1-238.1**, as indicated in stack state **242.1**. Upon completion of a successful instantiation of stack **271**, application **250** can send a notification message **297** to coordinator **290**, indicating a completion of the transfer of the stack. Coordinator **290** can subsequently transmit a notification message **298** to application **210** indicating either or both of the successful completion of the transfer and an indication to application **210** that corresponding stack **231** may resume normal operation.

Exemplary Method for Transferring Stack State in a CCN

[0054] FIG. **3**A presents a flow chart **300** illustrating a method for transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention. During operation, a coordinating entity ("coordinator") sends a command message to a stack manager, the message indicating to the stack to prepare for a transfer ("transfer command message") (operation **302**). The stack manager sends the transfer command message to all components of the stack (operation **304**). Each stack com-

ponent receives the transfer command message (operation **306**) and determines its component state (operation **308**). Each stack component determines if its component state is depends on the state of any of the other stack components (decision **310**). If it does not, the stack component stores its component state (operation **314**). If it does, the stack component waits for an indicator of a ready state for the other dependent stack components (operation **312**) before proceeding to operation **314**. Each stack component then notifies the stack manager of its ready state (operation **316**). The stack manager notifies the coordinator of a ready state for the stack (operation **318**) and the operation continues as described by Label A of FIG. **3**B or by Label B of FIG. **3**C.

[0055] FIG. **3**B presents a flow chart **330** illustrating a method for transferring the state of a stack in a content centric network, where the coordinator communicates with the stack via the stack manager, in accordance with an embodiment of the present invention. During operation, the coordinator receives a notification message from the stack manager, the message indicating a ready state for the stack (operation **332**). The coordinator generates an interest message that is a request for the stack state (operation **334**), and transmits the interest message to the stack manager (operation **336**). The stack manager transmits an interest to each stack component requesting its respective component state (operation **338**). Each stack component sends it component state to the stack manager (operation **340**). The stack manager receives the component state for each component (operation **342**) and generates a content object that indicates the stack state (operation **344**). The content object can further indicate the state for each component. The stack manager then transmits the content object that indicates the stack state to the coordinator (operation **346**). The coordinator receives the content object that indicates the stack state (operation **348**). Subsequently, the coordinator initiates the transfer of the stack to another location based on the content object that indicates the stack state (operation **350**). In some embodiments, the transfer can be a duplication for failover, load-balancing, or other resource-distribution purposes.

[0056] FIG. **3**C presents a flow chart **360** illustrating a method for transferring the state of a stack in a content centric network, where the coordinator communicates with another application, in accordance with an embodiment of the present invention. During operation, the coordinator receives a notification message from the stack manager, the message indicating a ready state for the stack (operation **362**). The coordinator sends the notification message to another application and includes unique, routable (e.g., reachable) names for the stack and its components, based on the naming scheme described in U.S. patent application Ser. No. 14/746,490 (operation **364**). The other application can be an application on another stack or a manager component of another stack, and can reside on the same or a different device. The other application generates an interest message that is a request for the stack state (operation **366**), and transmits the interest message to the stack manager (operation **368**). Operation **370** is similar to operations **338**-**344** described in relation to FIG. **3**B: the stack manager transmits an interest to each stack component requesting its respective component state (operation **338**); each stack component sends it component state to the stack manager (operation **340**); the stack manager receives the component state for each component (operation **342**) and generates a content object that indicates the stack state (operation **344**).

[0057] Subsequently, the stack manager transmits the content object indicating the stack state to the other application (operation **372**). The other application receives the content object indicating the stack state (operation **374**), and initiates the instantiation of a new stack based on the content object indicating the stack state (operation **376**).

Role of Stack Component

[0058] FIG. **4** presents a flow chart **400** illustrating a method by a stack component for facilitating the transfer of a stack, in accordance with an embodiment of the present invention. During operation, the system receives, by a component of a stack from a coordinating entity ("coordinator"), a command message to store a current state of the component (operation **402**). The component can be a communication component used in processing messages based on a name, and the name can be an HSVLI which comprises contiguous name components ordered from a most general level to a most specific level. The component determines its current state (operation **404**). The component then determines whether it is dependent on the state of any other stack components in order to enter a ready state (decision **406**). If it is not, the component stores its current state in a data structure (operation **410**). If the component is dependent on any other stack components, the component waits for an indicator of a ready state for the dependent stack components. In some embodiments, a dependent stack component can signal the component based on a ready state or any other state as required by the dependent stack component. Upon receiving the indicator that all dependent stacks are in a ready state, the component stores its current state in a data structure (operation **410**). The component also generates a notification message indicating that the current state is stored in the data structure and includes a name for the data structure (operation **412**). The data structure can be a manifest that indicates a set of content objects and their corresponding digests, and a respective content object can be a data object or another manifest. Manifests (e.g., secure content catalogs) are described in U.S. patent application Ser. No. 14/231,515, which is herein incorporated by reference.

[0059] Subsequently, the component can receive a packet that corresponds to an interest message, where the message indicates a request for the data structure which indicates the current state (operation **414**). The component generates a packet that corresponds to a content object message, where the message includes the data structure (operation **416**).

Exemplary Apparatus and Computer System

[0060] FIG. **5** illustrates an exemplary apparatus **500** that facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention. Apparatus **500** can comprise a plurality of modules which may communicate with one another via a wired or wireless communication channel. Apparatus **500** may be realized using one or more integrated circuits, and may include fewer or more modules than those shown in FIG. **5**. Further, apparatus **500** may be integrated in a computer system, or realized as a separate device which is capable of communicating with other computer systems and/or devices. Specifically, apparatus **500** can comprise a communication module **502**, a state-determining module **504**, a state-storing module **506**, a packet-generating module **508**, and a coordinator module **510**.

[0061] In some embodiments, communication module **502** can send and/or receive data packets to/from other network nodes across a computer network, such as a content centric network, where the data packets can correspond to: a command or a notification message for a communication component; an interest for a data structure; and a content object that includes a data structure.

[0062] Destination-determining module **504** can determine a current state for the component, and state-storing module **506** can store the current state for the component in a data structure. Packet-generating module **508** can generate a packet that corresponds to a content object message, which includes the data structure and a name for the data structure. Packet-generating module **508** can also generate a notification message indicating that the current state is stored in the data structure. Communication module **502** can, responsive to receiving a request for the current state, transmit the current state. State-determining module **504** can further determine a current state for one or more other communication components of the stack. Responsive to determining that the current state for the other components is a pending state, state-determining module **504** can wait for an indicator of a ready state for the other components. Coordinator module **510** can generate a command message for a manager component of the stack, where a manager component is used in processing messages between the coordinator and the components of the stack. Coordinator module **510** can also generate an interest message which is a request for the data structure.

[0063] FIG. **6** illustrates an exemplary computer system that facilitates transferring the state of a stack in a content centric network, in accordance with an embodiment of the present invention. Computer system **602** includes a processor **604**, a memory **606**, and a storage device **608**. Memory **606** can include a volatile memory (e.g., RAM) that serves as a managed memory, and can be used to store one or more memory pools. Furthermore, computer system **602** can be coupled to a display device **610**, a keyboard **612**, and a pointing device **614**. Storage device **608** can store an operating system **616**, a transport system **618**, and data **632**.

[0064] Transport system **618** can include instructions, which when executed by computer system **602**, can cause computer system **602** to perform methods and/or processes described in this disclosure. Specifically, transport system **618** may include instructions for sending and/or receiving data packets to/from other network nodes across a computer network, such as a content centric network (communication module **620**). For example, transport system **618** can include instructions for receiving, by a communication component from a coordinating entity, a command message to store a current state of the component, where the communication component is used in processing messages based on a name (communication module **620**). Transport system **618** can include instructions for determining a current state for the component (state-determining module **620**), and for storing the current state for the component in a data structure (state-storing module **624**).

[0065] Transport system **618** can also include instructions for generating a packet that corresponds to a content object message, which includes the data structure and a name for the data structure (packet-generating module **626**). Transport system **618** can include instructions for generating a notification message indicating that the current state is stored in the data structure (packet-generating module **626**). Transport system **618** can also include instructions for, responsive to receiving a request for the current state, transmitting the current state (communication module **620**). Transport system **618** can include instructions for determining a current state for one or more other communication components of the stack (state-determining module **622**). Transport system **618** can further include instructions for, responsive to determining that the current state for the other components is a pending state, waiting for an indicator of a ready state for the other components (state-determining module **622**).

[0066] Transport system **618** can additionally include instructions for generating a command message for a manager component of the stack, where a manager component is used in processing messages between the coordinator and the components of the stack (coordinator module **628**). Transport system **618** can also include instructions for generating an interest message which is a request for the data structure (coordinator module **628**).

[0067] Data **632** can include any data that is required as input or that is generated as output by the methods and/or processes described in this disclosure. Specifically, data **632** can store at least: a command message to store a current state of a component; a name; a name that is an HSVLI which comprises contiguous name components ordered from a most general level to a most specific level; a state for a component; a state for a stack of components; a data structure which represents a component state; a data structure which represents a stack state; a packet that corresponds to an interest, where the interest indicates a request for a data structure; a transport framework; a stack; one or more components of a stack; a coordinating entity; a manager component; a packet that corresponds to a content object message that includes a data structure; a notification message indicating that the current state is stored in a data structure; a manifest that indicates a state of a component or a stack; and an indicator of a ready, pending, or other state.

[0068] The data structures and code described in this detailed description are typically stored on a computer-readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. The computer-readable storage medium includes, but is not limited to, volatile memory, non-volatile memory, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs), DVDs (digital versatile discs or digital video discs), or other media capable of storing computer-readable media now known or later developed.

[0069] The methods and processes described in the detailed description section can be embodied as code and/or data, which can be stored in a computer-readable storage medium as described above. When a computer system reads and executes the code and/or data stored on the computer-readable storage medium, the computer system performs the methods and processes embodied as data structures and code and stored within the computer-readable storage medium.

[0070] Furthermore, the methods and processes described above can be included in hardware modules. For example, the hardware modules can include, but are not limited to, application-specific integrated circuit (ASIC) chips, field-programmable gate arrays (FPGAs), and other programmable-logic devices now known or later developed. When the hardware modules are activated, the hardware modules perform the methods and processes included within the hardware modules.

[0071] The foregoing descriptions of embodiments of the present invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.

What is claimed is:

1. A computer system for forwarding packets, the system comprising:

a processor; and

a storage device storing instructions that when executed by the processor cause the processor to perform a method, the method comprising:

receiving, by a communication component from a coordinating entity, a command message to store a current state of the component, wherein the communication component is used in processing messages based on a name, and wherein a name is a hierarchically structured variable length identifier (HSVLI) which comprises contiguous name components ordered from a most general level to a most specific level;

determining a current state for the communication component; and

storing the current state for the communication component in a data structure.

2. The computer system of claim **1**, wherein the method further comprises:

receiving a first packet that corresponds to an interest message, wherein the interest message indicates a request for the data structure and includes a name for the data structure.

3. The computer system of claim **2**, wherein the first packet is received from one or more of:

a manager component of a stack of communication modules, wherein a manager component is used in processing messages between the coordinating entity and the communication modules, wherein a stack does not require a respective communication module to communicate only with a layer above or below thereof, and wherein the communication component belongs to the stack; and

the coordinating entity, which is one or more of:

an entity or a service external to the stack; and

an application associated with the stack.

4. The computer system of claim **1**, wherein the method further comprises:

generating a second packet that corresponds to a content object message, wherein the content object message includes the data structure and a name for the data structure.

5. The computer system of claim **1**, wherein the method further comprises:

generating a notification message indicating that the current state is stored in the data structure, wherein the notification message includes a name for the data structure.

6. The computer system of claim **1**, wherein the data structure indicates one or more interest or content object messages stored by the communication component in processing the messages.

7. The computer system of claim **1**, wherein the data structure is a manifest, wherein a manifest indicates a set of content objects and their corresponding digests, wherein a respective content object is a data object or another manifest, and wherein a manifest and a content object each indicate a name.

8. The computer system of claim **1**, wherein the method further comprises:

responsive to receiving a request for the current state, transmitting the current state.

9. The computer system of claim **1**, wherein the communication component is a component of a stack of communication modules, wherein the stack does not require a respective communication module to communicate only with a layer above or below thereof.

10. The computer system of claim **9**, wherein determining the current state further comprises:

determining a current state for one or more other communication components of the stack;

responsive to determining that the current state for the other communication components is a pending state, waiting for an indicator of a ready state for the other communication components.

11. The computer system of claim **9**, wherein the coordinating entity is one or more of:

an entity or a service external to the stack; and

an application associated with the stack.

12. The computer system of claim **9**, wherein the command message is received from the coordinating entity via a manager component of the stack, wherein a manager component is used in processing messages between the coordinating entity and the communication modules.

13. A computer-implemented method, comprising:

receiving, by a communication component from a coordinating entity, a command message to store a current state of the component, wherein the communication component is used in processing messages based on a name, and wherein a name is a hierarchically structured variable length identifier (HSVLI) which comprises contiguous name components ordered from a most general level to a most specific level;

determining a current state for the communication component; and

storing the current state for the communication component in a data structure.

14. The method of claim **13**, further comprising:

receiving a first packet that corresponds to an interest message, wherein the interest message indicates a request for the data structure and includes a name for the data structure.

15. The method of claim **14**, wherein the first packet is received from one or more of:

a manager component of a stack of communication modules, wherein a manager component is used in processing messages between the coordinating entity and the communication modules, wherein a stack does not require a respective communication module to communicate only with a layer above or below thereof, and wherein the communication component belongs to the stack; and

the coordinating entity, which is one or more of:

an entity or a service external to the stack; and

an application associated with the stack.

16. The method of claim **13**, further comprising:

generating a second packet that corresponds to a content object message, wherein the content object message includes the data structure and a name for the data structure.

17. The method of claim **13**, further comprising:

generating a notification message indicating that the current state is stored in the data structure, wherein the notification message includes a name for the data structure.

18. The method of claim **13**, wherein the data structure indicates one or more interest or content object messages stored by the communication component in processing the messages.

19. The method of claim **13**, wherein the data structure is a manifest, wherein a manifest indicates a set of content objects and their corresponding digests, wherein a respective content object is a data object or another manifest, and wherein a manifest and a content object each indicate a name.

20. The method of claim **13**, further comprising:

responsive to receiving a request for the current state, transmitting the current state.

21. The method of claim **13**, wherein the communication component is a component of a stack of communication modules, wherein the stack does not require a respective communication module to communicate only with a layer above or below thereof.

22. The method of claim **21**, wherein determining the current state further comprises:

determining a current state for one or more other communication components of the stack;

responsive to determining that the current state for the other communication components is a pending state, waiting for an indicator of a ready state for the other communication components.

23. The method of claim **21**, wherein the coordinating entity is one or more of:

an entity or a service external to the stack; and

an application associated with the stack.

24. The method of claim **21**, wherein the command message is received from the coordinating entity via a manager component of the stack, wherein a manager component is used in processing messages between the coordinating entity and the communication modules.

* * * * *