Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2402254 A1 2001/02/22

(21) 2 402 254

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2000/08/15
(87) Date publication PCT/PCT Publication Date: 2001/02/22
(85) Entrée phase nationale/National Entry: 2002/09/05
(86) N° demande PCT/PCT Application No.: SE 2000/001581
(87) N° publication PCT/PCT Publication No.: 2001/013226
(30) Priorité/Priority: 1999/08/16 (09/375,177) US

(51) Cl.Int.^{7}/Int.Cl.^{7} G06F 9/46, G06F 9/445

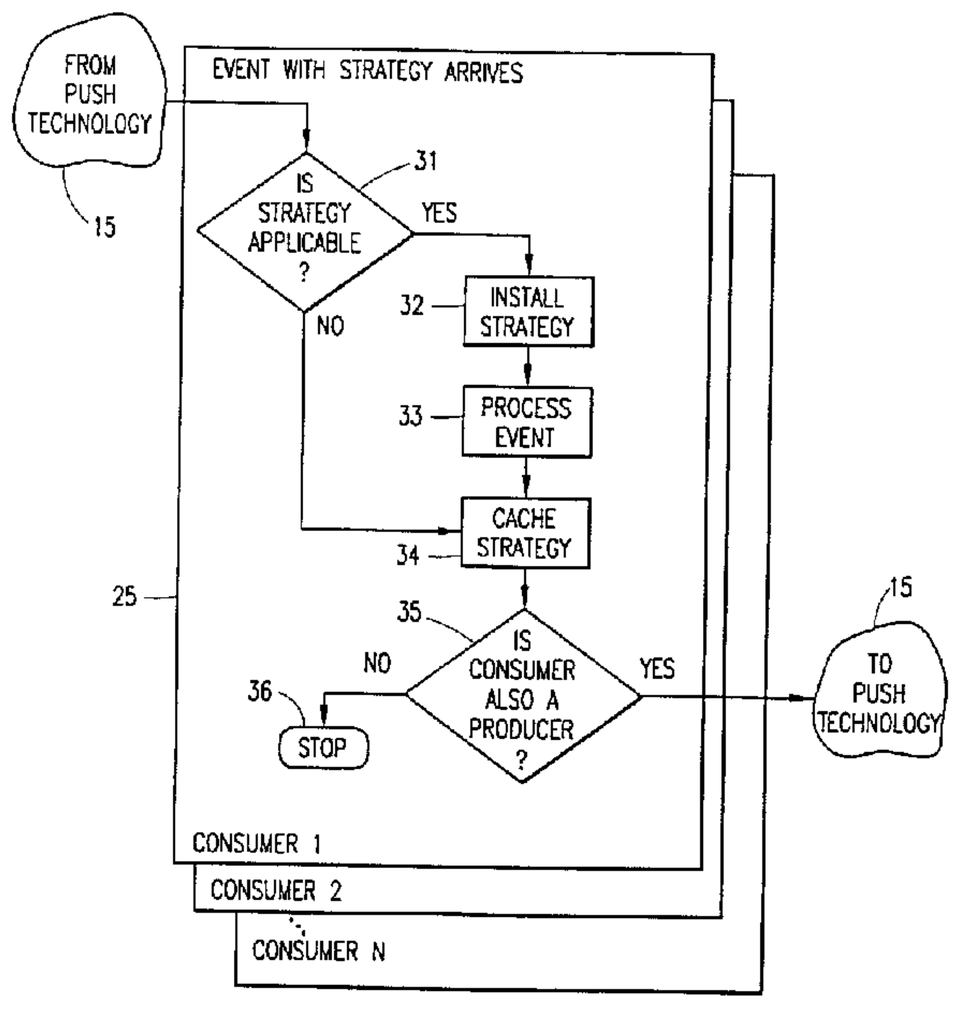
(71) Demandeur/Applicant:
TELEFONAKTIEBOLAGET LM ERICSSON (PUBL), SE

(72) Inventeurs/Inventors:
GOSSELIN, NICOLAS, CA;
TSE, EDWIN, CA;
KELLEDY, FERGUS, IE;
O'FLANAGAN, DAVID, IE

(74) Agent: BEAUCHESNE, NICOLAESCU

(54) Titre : PROCEDE DE DISTRIBUTION DE MISES A JOUR D'UN LOGICIEL DANS UN RESEAU DIRIGE PAR LES EVENEMENTS

(54) Title: METHOD OF DISTRIBUTING SOFTWARE UPDATES IN AN EVENT-DRIVEN NETWORK



(57) Abrégé/Abstract:
A method of distributing software to handle previously unknown event types in a large multi-domain "federation" network of decoupled, event-driven components. The components comprise producer components (10) which produce events and consumer components (25) which consume events to fulfill their specific roles in the federation. A producer component first determines a plurality of new events occurring in the network as a result of a new event type, and classifies the plurality of new events into at least one event-specific class. The producer component then pushes a single event (14) of an event-specific class into the network along with a new software strategy (13) for handling the event-specific class. The new strategy is distributed to the components in the network utilizing push technology (15). If the new strategy is applicable to a given component (31), the new strategy is cached (32) by the component and utilized to process events of the new type. If not applicable to the component, the new strategy is cached (34) by the component for use by downstream components. The process of distributing a strategy is then repeated for each event-specific class. The strategies are cached in the network components and are pushed to new components that bind to the network and subscribe to events of the new type. Thereafter, the new strategies are utilized by the components that received them to process events of the new type.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 February 2001 (22.02.2001)

PCT

(10) International Publication Number
WO 01/13226 A1

(51) International Patent Classification[7]: G06F 9/46, 9/445

(21) International Application Number: PCT/SE00/01581

(22) International Filing Date: 15 August 2000 (15.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/375,177 16 August 1999 (16.08.1999) US

(71) Applicant: **TELEFONAKTIEBOLAGET LM ERICSSON (publ)** [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: **GOSSELIN, Nicolas**; 110, Du Blainvillier, Blainville, Quebec J7C 4Y1 (CA). **TSE, Edwin**; 4976 Jean Brillant, Montreal, Quebec H3W 1T7 (CA). **KELLEDY, Fergus**; 10 Oriel Terrace, Demense Road, Dundalk, Co. Louth (IE). **O'FLANAGAN, David**; 18 Haddington Square, Ballsbridge, Dublin 4 (IE).

(74) Agent: **ALBIHNS PATENTBYRA STOCKHOLM AB**; P.O. Box 5581, S-114 85 Stockholm (SE).

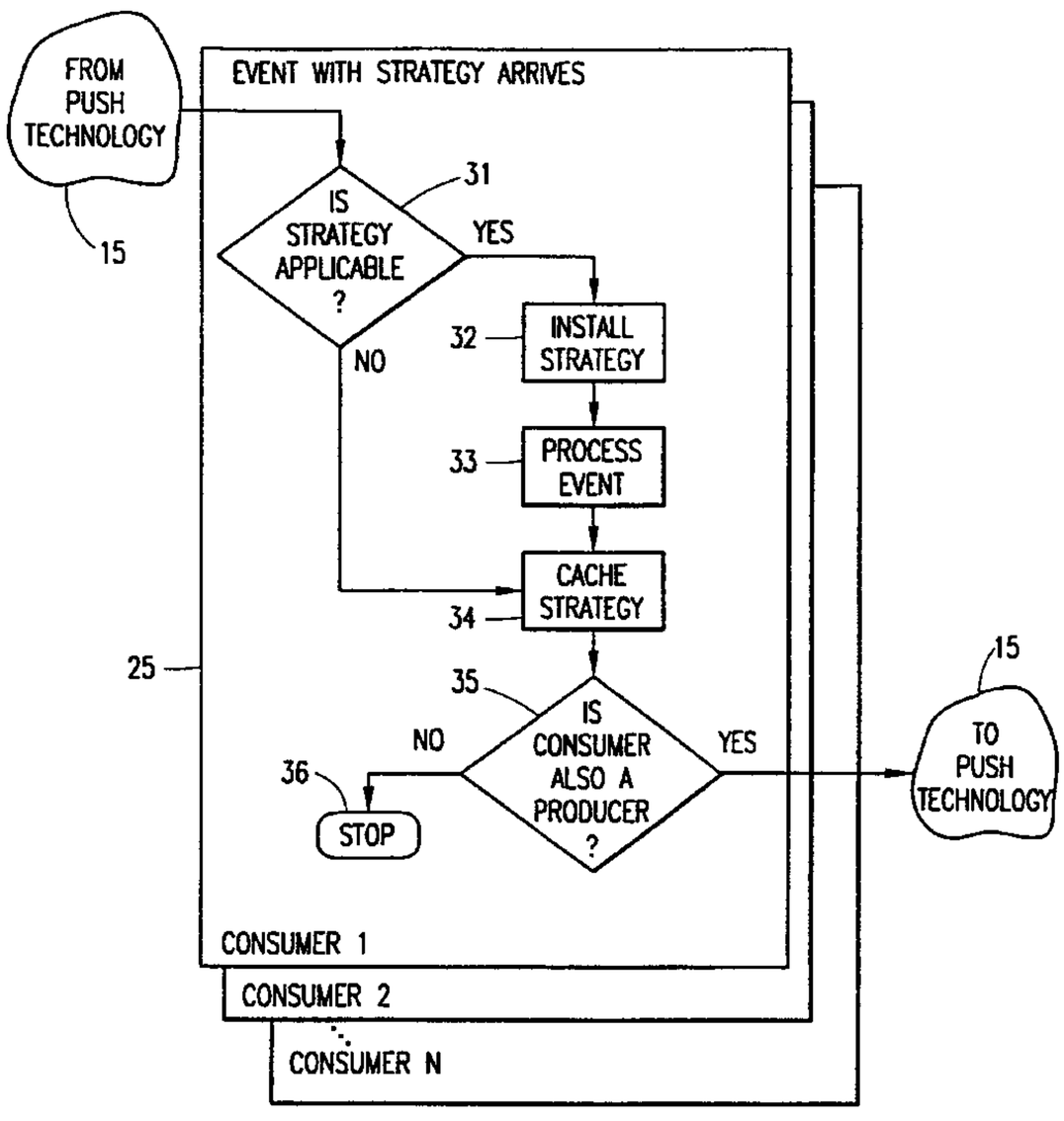
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD OF DISTRIBUTING SOFTWARE UPDATES IN AN EVENT-DRIVEN NETWORK



(57) Abstract: A method of distributing software to handle previously unknown event types in a large multi-domain "federation" network of decoupled, event-driven components. The components comprise producer components (10) which produce events and consumer components (25) which consume events to fulfill their specific roles in the federation. A producer component first determines a plurality of new events occurring in the network as a result of a new event type, and classifies the plurality of new events into at least one event-specific class. The producer component then pushes a single event (14) of an event-specific class into the network along with a new software strategy (13) for handling the event-specific class. The new strategy is distributed to the components in the network utilizing push technology (15). If the new strategy is applicable to a given component (31), the new strategy is cached (32) by the component and utilized to process events of the new type. If not applicable to the component, the new strategy is cached (34) by the component for use by downstream components. The process of distributing a strategy is then repeated for each event-specific class. The strategies are cached in the network components and are pushed to new components that bind to the network and subscribe to events of the new type. Thereafter, the new strategies are utilized by the components that received them to process events of the new type.

WO 01/13226 A1

METHOD OF DISTRIBUTING SOFTWARE UPDATES IN AN EVENT-DRIVEN NETWORK

BACKGROUND OF THE INVENTION

Technical Field of the Invention

This invention relates to software systems, and more particularly, to a method of distributing software updates to handle previously unknown event types in a large multi-domain federation network of event-driven components.

Description of Related Art

Currently, an increasing number of software systems (for example, telecommunication systems) rely on distributed architectures whose components are controlled across multiple administrative domains. The industry is looking for ways to avoid tightly coupled characteristics of components so that portions of the distributed system can be upgraded or changed without impact on other parts of the system.

In large telecommunications Network Management Systems (NMSs), various network elements under management are evolving rapidly over time. In addition, cooperative management agreements for the management of network elements are also changing frequently due to the rapid making and breaking of business partnerships in the telecommunications industry. These facts make it more critical that methods be devised to make upgrades or changes to network elements without adversely impacting other parts of the network. However, when a physical device (like a switch) or a logical device (like a software application) of a new type is brought into the network to be managed, problems may arise. At best, the existing installed management software may not be able to process the new event types, and the benefits of the new types of information generated by the new device are lost. At worst, the introduction of previously undefined event types may introduce instability throughout the system. Furthermore, it is likely that information related to the new event types may require some components in the federation to behave differently toward those events in terms of event storage, display, and event correlation.

There are currently two major solutions to the problems caused by the introduction of new events into large multi-domain federation networks of event-driven components. First, the network operator may upgrade all of the affected software components in the federation at the same time. In a large federation which is administered by various cooperative management domains in real-time, this can be a large and difficult task to administer. The second option is to design new event types to be backward-compatible with event types currently known in the federation. This approach, however, also has disadvantages. First, the new generations of event types must also carry functionality from older generations that may not be needed. Second, the requirement to be backward-compatible may limit the capabilities that the new generation can have. Third, after costly and time-consuming design of new generations to be backward-compatible, the older generation may be retired after installation of the new generation, thus making the effort and expense to achieve backward-compatibility unnecessary.

U.S. Patent No. 5,862,325 discloses an automated communication system that transfers data and methods from a provider computer to a consumer computer through a communications network. The system uses push technology to selectively distribute the information to those consumers that are in the provider's address database. Thus, the provider computer must maintain a database regarding which event types go to which consumers.

European Patent Application EP 0 759 591 A discloses an Event Management System (EMS) in which event consumers must first register with the EMS in order to start receiving events. The EMS stores the consumers in a Consumer Database. Thus, the EMS must maintain a database regarding which event types go to which consumer components.

In an IEEE paper entitled, "Interoperability of Event Service in Java ORB Environment" by Gil Seong Na et al., middle objects may act as proxy components to pass information between event channels or to other components. However, the producer component must still maintain a database regarding which event types go to which consumer components. The proxy components do not selectively use the information they receive if it is applicable to them, or store the information for use by downstream components if it is not applicable to them.

European Patent Application EP 0 537 098 A discloses an event handling mechanism having a filtering process and an action association process. The events of an event stream are filtered into categories of events. Once categorized, actions are associated with the categorized event. A system administrator uses a filter table to determine which event group is appropriate for a subject event. Thus, the administrator must maintain a database regarding which event types go to which components.

In order to overcome the disadvantage of existing solutions, it would be advantageous to have a method of distributing software updates which does not require the producer component to maintain a database regarding which event types go to which consumer components, and does not require the costly and sometimes unnecessary process of designing new event types to be backward-compatible with currently known event types. The present invention provides such a method.

SUMMARY OF THE INVENTION

In one aspect, the present invention is a method of distributing software to handle a new event type in a network having a plurality of decoupled, event-driven components which produce and consume events. The method includes the steps of pushing a new software strategy for handling a new event type from a producer component into the network, and utilizing push technology to distribute the new strategy to the components in the network. The step of distributing the new strategy may include selectively distributing the new strategy utilizing filtering capabilities in the push technology to determine whether the new strategy is applicable to each component in the network, and pushing the new strategy to those components for

which the new strategy is applicable.

In another aspect, the method of the present invention may utilize a publish and push technology that does not utilize filtering in the individual components. Additionally, the invention may be applied only to update existing components. In this case, the method comprises the steps of pushing a new software strategy for handling a new event type from a producer component to consumer components in the network, and utilizing the new strategy by the existing components to process events of the new type. If the method is to be utilized for new components as well, then the method also includes the steps of caching the new strategy in the components in the network, determining when a new consumer component binds into the network and subscribes to events of the new event type, and pushing the new strategy to the new component. The new strategy is then utilized by the new component to process events of the new type.

In another aspect, the method includes the steps of pushing a new software strategy for handling a new event type from a producer component into the network, and utilizing push technology to selectively distribute the new strategy to the components in the network. It is determined whether the new strategy is applicable to each component in the network, and upon determining that the new strategy is applicable to a given component, the strategy is cached by the given component and utilized by the given component to process events of the new type. Upon determining that the new strategy is not applicable to a given component, the strategy is cached by the given component for use by downstream components. When a new consumer component binds into the network and subscribes to events of the new event type, the new strategy is pushed to the new component.

The method may also begin by having the producer component first determine a plurality of new events occurring in the network as a result of a new event type. The producer component classifies the plurality of new events into at least one event-specific class. The producer component then pushes into the network, a single event of a first event-specific class along with a first new software strategy for handling the first event-specific class. The first new strategy is selectively distributed to the components in the network utilizing push technology. If the first new strategy is

applicable to a given component, the first new strategy is cached by the component and utilized to process events of the new type. If not applicable to the component, the first new strategy is cached by the component for use by downstream components. The process of distributing a strategy is then repeated for each event-specific class. The strategies are cached in the network components and are pushed to new components that bind to the network and subscribe to events of the new type. Thereafter, the new strategies are utilized by the components that received them to process events of the new type.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood and its numerous objects and advantages will become more apparent to those skilled in the art by reference to the following drawings, in conjunction with the accompanying specification, in which:

FIG. 1 is a flow chart illustrating the steps performed by a producer component in the preferred embodiment of the method of the present invention;

FIG. 2 is a flow chart illustrating the steps performed by the push technology in the preferred embodiment of the method of the present invention;

FIG. 3 is a flow chart illustrating the steps performed by a consumer component in the preferred embodiment of the method of the present invention; and

FIG. 4 is an illustrative drawing of an exemplary scenario showing the introduction of a new event type in a federation network according to the teachings of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS

The present invention describes a method of distributing software to handle previously unknown event types in a large multi-domain "federation" network of decoupled, event-driven components which manage, distribute, and process events. One way to build such a decoupled-component network is to use "publish and push" technologies. An information producer publishes the information and pushes it into the network. One or more information consumer(s) registers itself with the network, and the network pushes information to the information consumer. In this paradigm,

the information producer and the information consumer are decoupled by a store-and-forward medium. The producer need not be aware of the consumer(s), and states of the producers are independent from the states of the consumers (and vice versa).

In the specific case of telecommunication systems, the information may be network events such as network alarms, network performance data, network accounting data, etc. Information producers may be network elements, and information consumers may be telecommunications Network Management Systems (NMSs).

Thus, publish and push technologies facilitate the construction of a federation of distributed, event-driven components. In the federation, each component produces and/or consumes events to fulfill its specific role in the federation. Some of these components may be gateways that receive information emitted by devices or other systems, and translate the information into events. Then, they push the events into the network. Other components, interested in receiving those event types, subscribe to the network so that those events will be pushed to them in the future. These components, after receiving the events, process them and may generate new events based on the processed information which are also pushed into the network.

In the method of the present invention, software updates are automatically distributed in the federation. Components issuing new event types (producers) are responsible for supplying consumer components with appropriate software (hereinafter called Strategy) for handling the new event types. The Strategy, which is normally stored within the event, may be entirely new code for handling events of the new type, or it may be a patch for existing code. When the event with Strategy is received at a consumer component, the consumer decides whether the strategy is applicable to it. If so, the Strategy is installed and thereafter affects the behavior of the consumer component according to information carried in the new event type.

FIG. 1 is a flow chart illustrating the steps performed by a producer component (Producer) 10 in the preferred embodiment of the method of the present invention. A plurality of Producers, represented as Producers 1-N may be performing the steps of FIG. 1 for different events. At step 11, the Producer creates a new event. The Producer may use its knowledge of component types in the federation to determine a

plurality of new events that will be generated by the introduction of a new event type in the federation. The Producer classifies the new events into event-specific classes, and determines an associated Strategy for each event-specific class. The number of required Strategies depends on the number of roles a new event will affect. At step 12, the Producer determines whether or not the Strategy for the event has already been pushed into the network. If the Strategy has not been pushed, the method moves to step 13 where the strategy is stored in the event. At step 14, the event is then pushed to the push technology 15 for distribution in the network regardless of whether there are consumers for the event type or not. Ideally, the event conveying the class should include a release number corresponding to the format of the produced event type it is designed to handle. If it is determined at step 12 that the Strategy for the event has already been pushed, then the event is pushed at step 14 without Strategy.

FIG. 2 is a flow chart illustrating the steps performed by the push technology 15 in the preferred embodiment of the method of the present invention when an event is received from a Producer 10. At step 21, it is determined whether or not there is a filtering capability in the push technology. If so, it is determined at 22 whether or not the event is to be filtered. If so, the event is filtered out and discarded at 23. If there are no filtering capabilities, or the event is not to be filtered, the method moves to step 24 where the event is pushed to consumer components (Consumers) 25.

In push technology with filtering capabilities, filters in the push technology selectively distribute the new Strategy to consumer components in the federation. Thus, the filtering capabilities of the push technology being utilized in the network for inter-component communications determine whether or not a given Strategy is applicable to a given component. For example, if a filter is inserted upstream from a particular component, and the filter blocks the distribution of a particular Strategy, then the component has no knowledge of the Strategy until the filter is removed.

Some publish and push technologies do not utilize filtering; if such a technology is utilized, the Strategy is pushed to all of the existing components. After the Strategy is distributed, the consumers utilize the new Strategy to process events of the new type instead of the generic Strategy they were provided with at installation.

FIG. 3 is a flow chart illustrating the steps performed by a consumer

component 25 in the preferred embodiment of the method of the present invention when an event with Strategy arrives from the push technology 15. At step 31, it is determined whether or not the Strategy is applicable for the receiving Consumer. If so, the Consumer installs the Strategy at 32 and then uses the Strategy to process events of the new type at 33. The Consumer then caches the Strategy at step 34. If it is determined at step 31 that the Strategy is not applicable for the receiving Consumer, the method moves directly to step 34 and caches the Strategy for subsequent distribution to downstream components that require it.

At step 35, it is determined whether or not the Consumer 25 is also a Producer of events for downstream components. If so, the event with Strategy is then pushed to the push technology 15 for distribution to downstream components in the network. If the Consumer is not a Producer, the method moves to step 36 where the processing within the Consumer stops.

The distributed Strategies implement a role-based interface. Each role a component can play is associated with an abstract interface. For example, a component responsible for storing information in a database may use a Strategy which implements an interface that returns a unique key for a given event (to be stored). That key is usually specific to one event type. A producer of a new event type should then produce an appropriate Strategy that can create a key based on the new information in the new event type. If a new consumer binds into the federation and subscribes for events of the new type, the federation downloads the appropriate Strategy to the consumer in its initial download, and the new Strategy is then utilized by the new consumer to process events of the new type.

FIG. 4 is an illustrative drawing of an exemplary scenario showing the introduction of a new event type in a federation network according to the teachings of the present invention. Components P1 41 and P2 42 are Producers of events 43 and 44, respectively, in the federation. Along with the events, P1 and P2 provide associated Strategies stored in the events regardless of whether there are consumers in the federation for those event types or not. The events and Strategies are provided via push technology (PT) 15 to consumer C1 45. After forwarding the events utilizing PT 15, the events and Strategies are cached in C1.

PT 15 then pushes the events to consumers C3 46 and C4 47. In this example, C3 is a component which correlates alarm messages it receives and issues correlation events 48 back into the federation through PT 15. Likewise, C4 is exemplified as a component which displays events to an operator through a user interface, and provides the operator with a software connection 49 to a physical device 51 connected to P2.

In the scenario of FIG. 4, Producer P2 connects to the new physical device 51, and determines a Strategy associated with this event. The new event and the new Strategy are pushed through the federation, as described above, to consumer C1 where they are cached. The new Strategy is pushed to consumers C3 and C4 where it is cached. Consumer C3 in this example determines that the new Strategy is not applicable, and therefore disregards the new Strategy. Consumer C4 determines that the new Strategy is applicable to it, and installs the new Strategy. C4 may then use the new Strategy to provide the software connection 49 to the physical device 51, or to communicate with P2.

If a new consumer C5 52 binds to the federation and subscribes to the new event, C1 sends the new Strategy to C5. Consumer C5 caches the new Strategy and thereafter utilizes the new Strategy when it receives events of the new type.

It is thus believed that the operation and construction of the present invention will be apparent from the foregoing description. While the method, apparatus and system shown and described has been characterized as being preferred, it will be readily apparent that various changes and modifications could be made therein without departing from the scope of the invention as defined in the following claims.

WHAT IS CLAIMED IS:

1. A method of distributing software to handle a new event type (11) in a network having a plurality of decoupled, event-driven components, said components being producers of events (10) and consumers of events (25), said method comprising the steps of:

pushing from a producer component into the network, a new software strategy for handling the new event type (14);

utilizing push technology to distribute the new software strategy to all of the consumer components in the network (15);

determining in each given consumer component, whether the new software strategy is applicable to the given consumer component (31);

upon determining by a first consumer component that the new strategy is applicable to the first consumer component, caching the strategy by the first consumer component (34) and utilizing the new strategy by the first consumer component to process events of the new type (33);

upon determining that the new strategy is not applicable to a second consumer component:

caching the strategy by the second consumer component (34);

determining whether the second consumer component is also a producer component for downstream consumer components (35); and

upon determining that the second consumer component is also a producer component for downstream consumer components, utilizing push technology by the second consumer component to distribute the new software strategy to the downstream consumer components (15).

2. The method of claim 1 further comprising the steps of:

determining when a new consumer component binds into the network and subscribes to events of the new event type; and

utilizing push technology to distribute the new software strategy to the new component.

3. A method of distributing software to handle a new event type (11) in a network having a plurality of decoupled, event-driven components, said components including a producer component (10) which produces events and at least one consumer component (25) which consumes events, said method comprising the steps of:

(A) determining by the producer component, a plurality of new events occurring in the network as a result of the new event type (11);

(B) classifying by the producer component, the plurality of new events into at least one event-specific class;

(C) pushing from the producer component into the network, a single event of a first event-specific class along with a first new software strategy for handling the first event-specific class (14);

(D) utilizing push technology to distribute the first new strategy to all of the components in the network (15);

(E) determining in each particular component whether the first new strategy is applicable to that particular component (31);

(F) upon determining that the first new strategy is applicable to the particular component, caching the first new strategy by the particular component (34) and utilizing the first new strategy by the particular component to process events of the new type (33);

(G) upon determining that the first new strategy is not applicable to a particular component:

caching the strategy by the particular component (34);

determining whether the particular component is also a producer component for downstream consumer components (35); and

upon determining that the particular component is also a producer component for downstream consumer components, utilizing push technology by the particular component to distribute the new software strategy to the downstream consumer components (15);

(H) determining whether the event-specific class is the last class classified by the producer component in step (B);

(I) repeating steps (C) through (H) for each additional event-specific class classified by the producer component in step (B), thereby distributing software strategies corresponding to each event-specific class; and

(J) utilizing the new strategies by the components to process events of the new type.

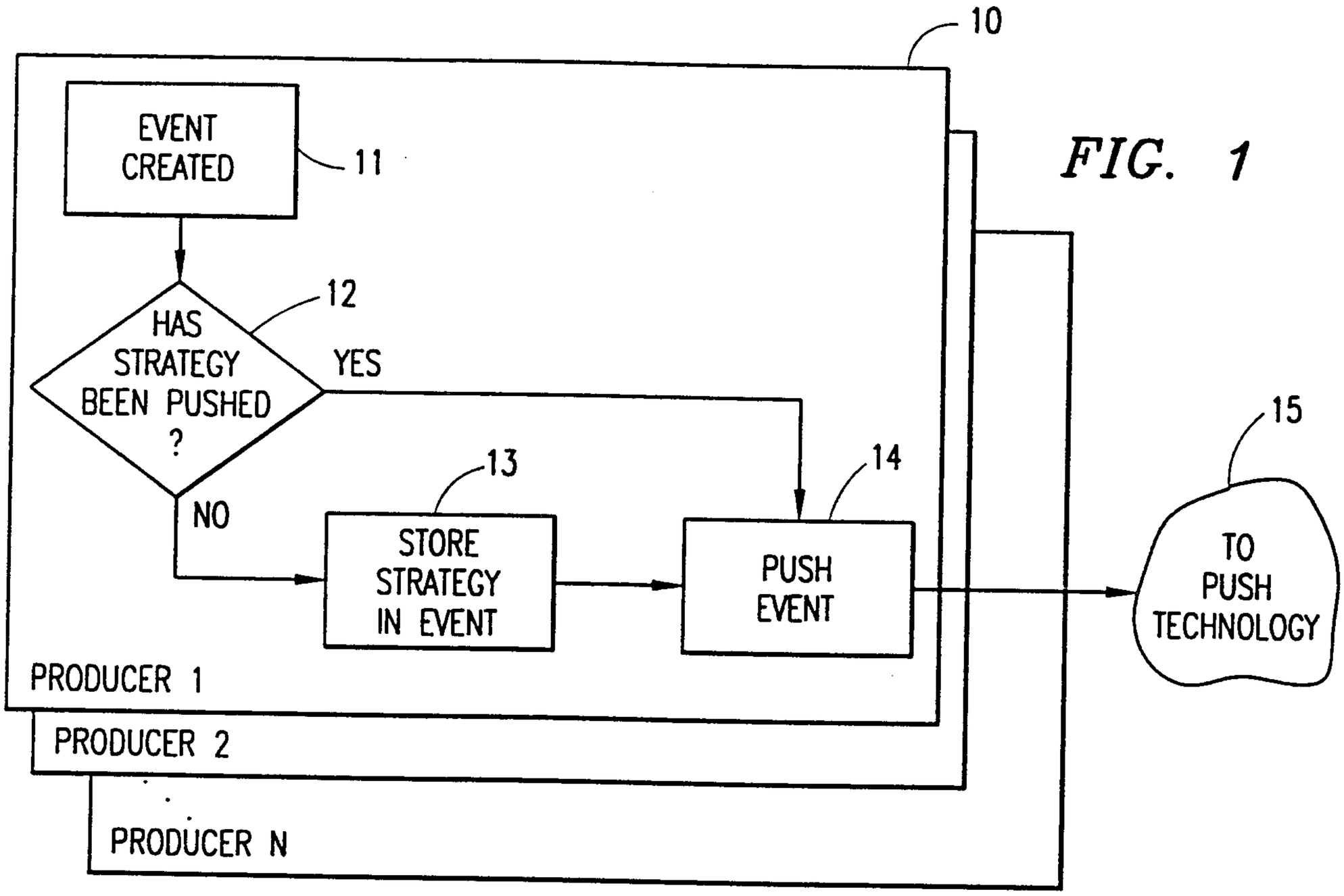

10
EVENT CREATED
11
HAS STRATEGY BEEN PUSHED ?
12
YES
NO
13
STORE STRATEGY IN EVENT
14
PUSH EVENT
15
TO PUSH TECHNOLOGY
PRODUCER 1
PRODUCER 2
PRODUCER N

FIG. 1

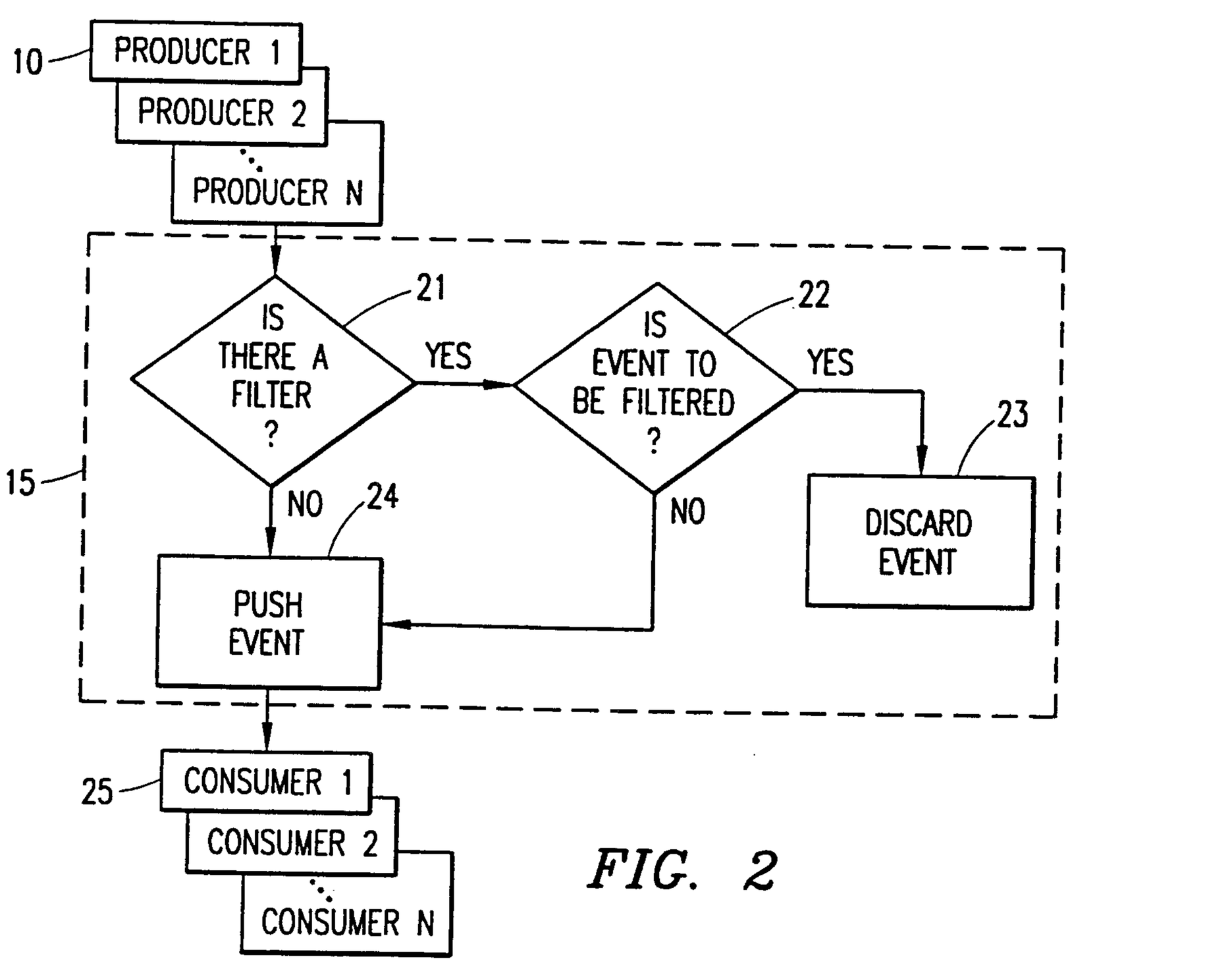

10
PRODUCER 1
PRODUCER 2
PRODUCER N
IS THERE A FILTER ?
21
YES
IS EVENT TO BE FILTERED ?
22
YES
NO
23
DISCARD EVENT
15
NO
24
PUSH EVENT
25
CONSUMER 1
CONSUMER 2
CONSUMER N

FIG. 2

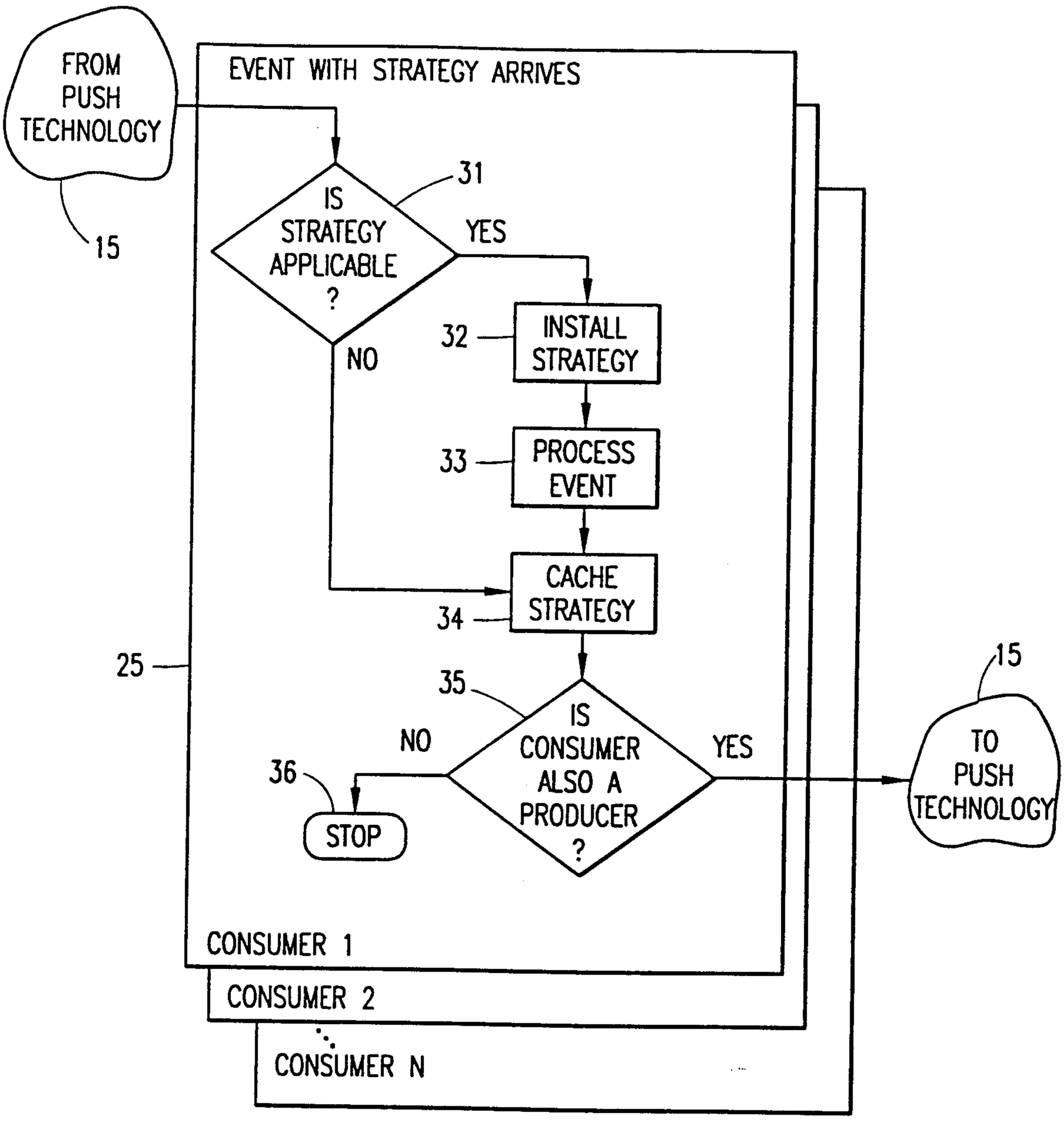
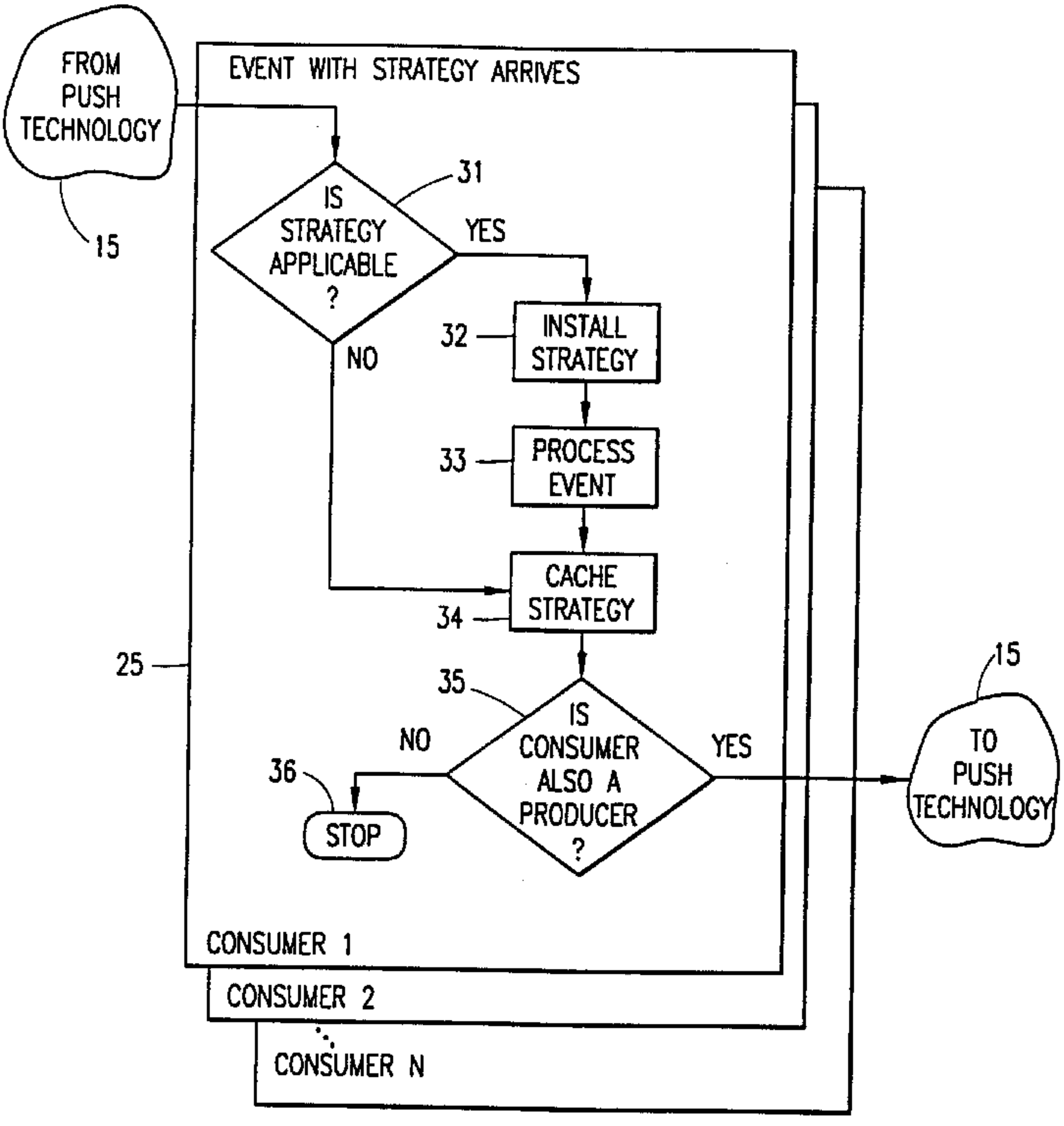

FROM PUSH TECHNOLOGY
15
EVENT WITH STRATEGY ARRIVES
31
IS STRATEGY APPLICABLE ?
YES
NO
32
INSTALL STRATEGY
33
PROCESS EVENT
34
CACHE STRATEGY
25
35
IS CONSUMER ALSO A PRODUCER ?
NO
YES
36
STOP
15
TO PUSH TECHNOLOGY
CONSUMER 1
CONSUMER 2
CONSUMER N

FIG. 3

48
41
43
46
P1
45
C3
51
PT
C1
PT
C5
P.D.
P2
44
15
15
C4
52
42
47
49

FIG. 4


FROM PUSH TECHNOLOGY
15
EVENT WITH STRATEGY ARRIVES
31
IS STRATEGY APPLICABLE ?
YES
NO
32
INSTALL STRATEGY
33
PROCESS EVENT
34
CACHE STRATEGY
25
35
IS CONSUMER ALSO A PRODUCER ?
NO
YES
36
STOP
15
TO PUSH TECHNOLOGY
CONSUMER 1
CONSUMER 2
CONSUMER N