April 18, 2017

Adding Token to Prevent Cycles

Michael Harm

Seth Howard

Adding Token to Prevent Cycles

Abstract:

In a computer system, a token added to a message can be reused in messages that are part of the same transaction. If the application that originally sent a request that included the token receives a request that includes the token, the application will not perform an additional action based on receiving the request. The stopping of further actions can occur based on receiving the same token once or a threshold number of times.

Some computer systems can include applications that send a message such as a request to a second application, prompting the second application to perform an action and to send a message such as a request or response to the first application. An error in the computer program code in either the first application or the second application can cause the first application to perform another action, which also prompts a similar request to the second action, creating a cycle that consumes resources or leads to other undesirable interactions. For example, a publish/subscribe system can allow subscriber applications to subscribe to events that they are interested in from publisher applications. Subscriber applications can receive an event and, in some instances, respond to the publisher application with a request that causes the publisher application to publish and/or to send the same event to the subscriber application, creating a cycle that continues indefinitely. The cycles can occur in computer systems with two applications or machines sending messages to each other, or in computer systems with more than two applications or machines.

To prevent undesired cycles, applications can add a token to messages such as requests and responses. The token is reused, with applications including the token in messages sent in response to the requests and responses. If an application receives a message with a token matching a token that the application included in a previously sent message, then the application will not perform actions such as responding to the received message.

FIG. 1 is a timing diagram showing actions performed by, and messages exchanged between, a first application 102 and a second application 104. The first application 102 can include a publisher that sends notifications of events to applications that subscribe to the first application 102. The second application 104 can include a subscriber that is subscribed to the first application 102 and receives notifications of events from the first application 102.

A trigger 106 event can occur. The trigger 106 can be receipt of information or data by the first application 102 that the first application 102 determines should be sent to the second application 104, and/or the trigger 106 can be the determination by the first application 102 that the information or data should be sent to the second application 104.

In response to the trigger 106, the first application 102 generates and sends a message 108 to the second application 104. The message 108 can include the data or information desired by the second application and that is the subject of the subscription, such as news about certain types of events.

In generating the message 108, the first application 102 can also generate a token. The token will identify the transaction that includes the message 108, and will be reused by the second application 104 in any response to the message 108. The message 108 will include the token. The first application 102 will store the token for later comparison to tokens included in messages received from other applications, such as the second application 104. The first application 102 can generate the message 108 in the form of a packet.

FIG. 2 is a diagram of a packet 200. The packet 200 represents an example of the message 108 shown in FIG. 1. The packet 200 includes a header 202 and a body 204. The body 204 includes the data or information that is the subject of the subscription of the second application 104 to the first application 102.

The header 202 includes a source address 206 identifying an address of the first application 102, a destination address 208 identifying an address of the second application 104, a token 210 generated by the first application 102 to identify the transaction, and an instruction 212 identifying a type of the packet 200. The token 210 can be an identifier of the transaction and/or event that is reused and/or copied by other applications such as the second application 104 that respond to the message 200. The token 210 can be included in the packet 200 and easily read, may be encrypted according to an encryption scheme known by both applications 102, 104, and/or may be sent as part of a side-channel message. The instruction 212 can identify a type of the packet 200, such as, in the context of Hypertext Transfer Protocol (HTTP) communication, a GET request, HEAD request, PUT request, a Success response, or a Redirection response.

Returning to FIG. 1, the second application 104 performs an action 110 in response to receiving the message 108. In the context of a publish/subscribe system, the action 110 can include presenting to a user of the second application 104 the data or information included in the message 108.

The action 110 and/or message 108 can prompt the second application 104 to send a message 112 to the first application 102. In a well functioning system, the message 112 can simply be an acknowledgment, informing the first application 102 that the message 108 was successfully received by the second application 104. However, poor design of the computer program code for the second application 104 can cause the message 112 to be a message that prompts the first application 102 to send another message similar to the message 108, creating a cycle of messages 108, 112. For example, the message 112 can be a request for a subscription to receive notifications of data or information similar the data or information included in the message 108, which would prompt the first application 102 to send another message similar or

identical to the message 108, which would prompt the second application 104 to request the subscription again, creating an indefinite cycle.

To prevent such a cycle, the second application 104 can include, in the message 112, the token that the first application 102 included in the message 108. The message 112 can have the format of the packet 200 shown in FIG. 2, and can include the token 210 included in the message 108. The token included in the message 112 can be identical to the token included in the message 108.

In response to receiving the message 112, the first application 102 can end (114) the cycle and/or transaction. The first application 102 can end the cycle and/or transaction based on comparing the token included in the message 112 to the token that the first application 102 stored when generating and sending the message 108 to the second application 104.

While FIG. 1 has been described in the context of a publish/subscribe system, the token can be used to prevent cycles in other systems. For example, changes to a scroll bar in a web browser can call a function which triggers another change to the scroll bar, which in turn calls the function that triggers another change to the scroll bar, resulting in a scroll bar that appears to be vibrating. The web browser application can generate a token for each user input to the scroll bar, preventing the cycle. Another example is a web-based calendar application that launches a script in response to user modification to the calendar, with the script prompting another change to the calendar, and the change to the calendar again launching the script. The calendar application can generate a token for each change to the calendar, preventing changes to the calendar from launching a script that leads to an indefinite cycle. Some functions or applications can prevent cycles based on the generated tokens, whereas other functions or applications can allow the cycles to run indefinitely or only a predetermined number of times.

While FIG. 1 shows two applications 102, 104 performing actions and exchanging messages, the token can be used to prevent undesired cycles in computer systems with more than two applications. While the first application 102 was described as ending the cycle and/or transaction after receiving the stored token once, the first application 102 can also increment a counter each time the first application 102 received the stored token. The first application 102 can respond to the message 112, even though the message included the stored token, as long as the counter did not meet or exceed a threshold. Once the counter meets or exceeds the threshold, the first application 102 can end 114 the cycle and/or transaction in response to receiving the stored token and/or in response to the counter meeting or exceeding the threshold.

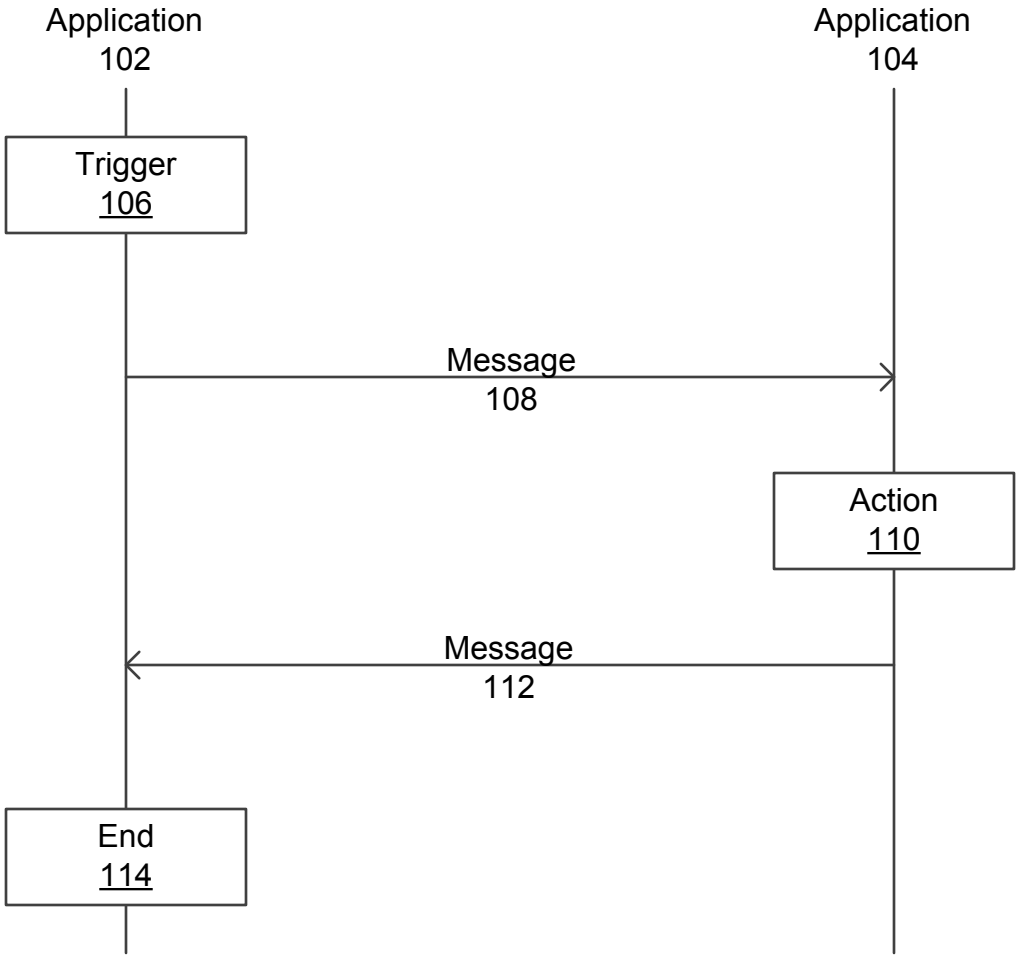
Application
102
Application
104
Trigger
106
Message
108
Action
110
Message
112
End
114

FIG. 1

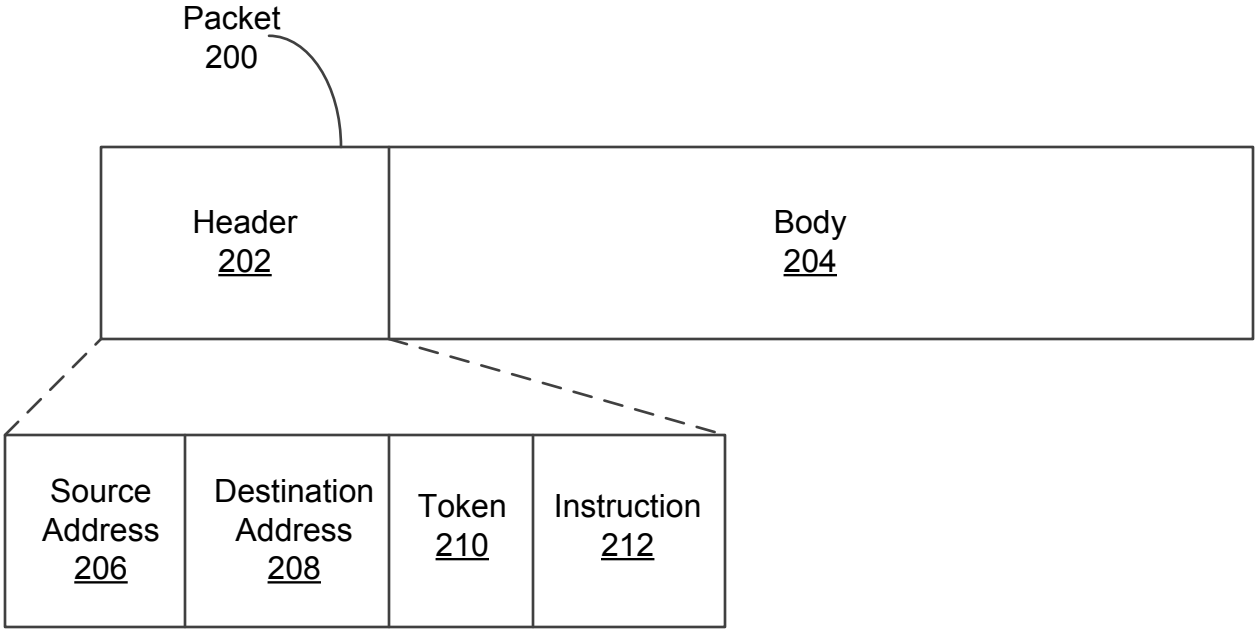
Packet
200
Header
202
Body
204
Source
Address
206
Destination
Address
208
Token
210
Instruction
212

FIG. 2