An approach to Contextual Transformation in EBSs

Suhas Chikkanna

1 Background

Among Distributed Systems, the Event-based Systems(EBSs) are used to disseminate event notifications on the occurrence of an event, generated by an event producer. The disseminated event notification is delivered to the event consumer, which only subscribes to an event of particular interest. The event notifications are delivered to the event consumers based on a match/satisfying criteria between the subscription and event notification. The EBSs are predominantly used in the supply chain, new ticker and share market. These applications are widely distributed, with heterogeneous components making up a system-wide architecture. An event detected &/or generated by a component, is propagated through event-notification service to the event consumer. The current publish/subscribe systems utilizes a common context, to infer an event produced by an event producer and, subscription subscribed by an event consumer. The producer and consumers may often differ in their respective contexts. And the differing contexts between producers and consumers leads to a difference in the interpretation of the event. Below in the related work section, ACTrESS suggests a solution to this differing context.

$$2\mathrm{Mg} + \mathrm{O_2} \longrightarrow 2\mathrm{MgO}$$

2 Related Work

ACTrESS suggests automatic contextual transformation for differ- ing contexts in Event-Based Software Systems at runtime, transpar- ent to end users and also, ACTrESS overcomes major challenges related to contextual transformations. In ACTrESS, the Producer Context is transformed to Root Context (at the broker the produc- er/publisher is connected to) and the Root Context is transformed to Consumer Context at the fringe of the broker network(i.e., at last broker the consumer is connected to).

3 Problem Statement

The ACTrESS prototype implements greedy algorithm and works best in most cases. But in ACTrESS, there may result unnecessary work in spite of producer and consumer being in the same context. For instance, in a channel-based pub/sub system and Topic-based pub/sub system. And also, certain topologies such as mobile net- works where the clients(i.e., producer/consumer) move from one broker to another, the same transformation (Producer Context Root Context Consumer Context) performed in the former broker has to be performed again on the latter broker, where the client may newly join. We try to address this with our heuristic in the below section.

4 Our Approach

Our approach focuses on, contextual transformation with respect to, taking in account the number of transformations to be performed & the number of messages generated in the network due to the transformations. Our approach focuses on keeping the number of contextual transformations and the messages generated due to these contextual transformations low. Our approach proposes to perform contextual transformation, at the first broker which has a subscriber in different context in comparison to the publisher's context.For instance, as shown in the Figure 2, the first broker which consists of a subscriber with different context in comparison to publisher's context is Broker B5. Hence, the contextual transformation is performed on Broker B5. This is the first key-point to be noted about our approach. Below is Figure 1, which represents a sample architecture of an EBS.

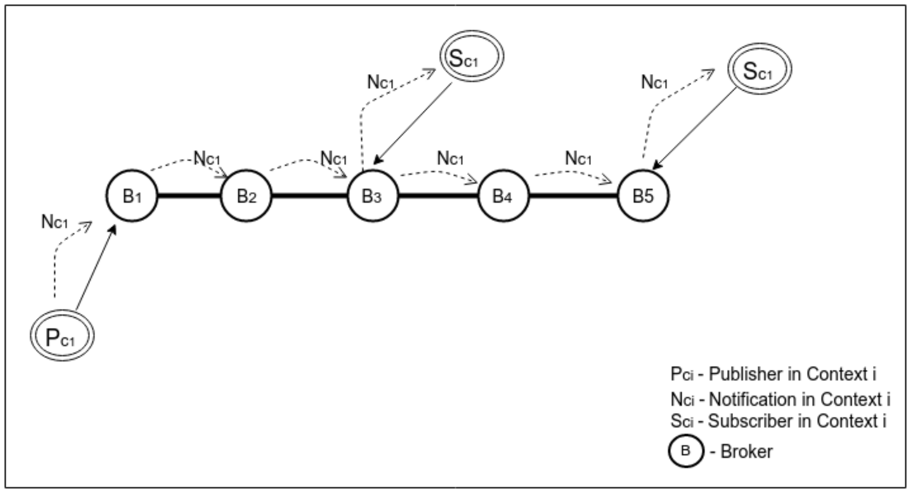

Figure 1: Publishers & Subscribers in the same context.

We will divide the above sample architecture into 3 different cases in order to explore the consequences of our proposed approach. It should be noted that all these 3 cases are a super-set of any other case that can be derived from the sample architecture in Figure 1.(i.e., the three cases mentioned below cover all the scenarios required to consider our approach). Hence, below are list of Cases/Figures, which are compared with each other to our approach of contextual transformation with respect to the subscribers connected to a particular broker. We compare all of these figures and the comparison is enumerated in the Table 1. as shown below.

Case 1

In the Figure 2, it can be noticed that, the Publisher Pc1 and Subscriber Sc1 are in the same context, whereas the Subscriber Sc2 is in a different context. As stated above, the contextual transformation occurs in the first broker where the Subscriber's context differs in comparison to the Publisher's context. Hence, in this case the contextual transformation takes place in Broker B5(As depicted by a Red dot in Figure 2). In such a case, we perform only 1 Contextual Transformation and in total there are 7 messages in the network, depicted as notifications in the Figure 2

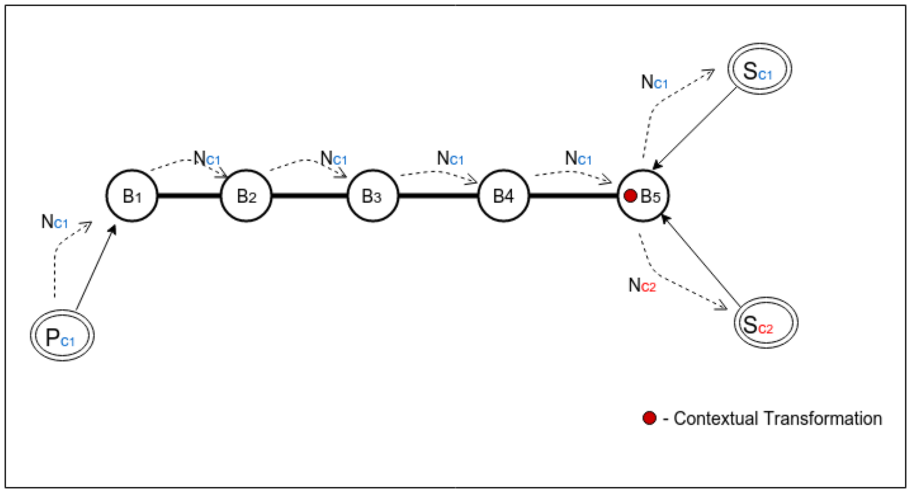

Figure 2: Publisher & Subscriber in different context.

Case 2

Now let us consider the case 2, where the Subscriber Sc2 in a different context is on the same broker as Publisher Pc1, as depicted in the Figure 3. In this case, as known the Contextual transformation is performed on the Broker B1, & after the contextual transformation, we obtain the notification Nc2 and along with this notification, the notification Nc1 is routed along the path to the other subscribers which are in the context Sc1 and Sc2. Note that performing contextual transformation as early as in the same broker where the publisher is connected adds extra notification messages in the network. But this will not be the same in all scenarios as depicted in the case 3.

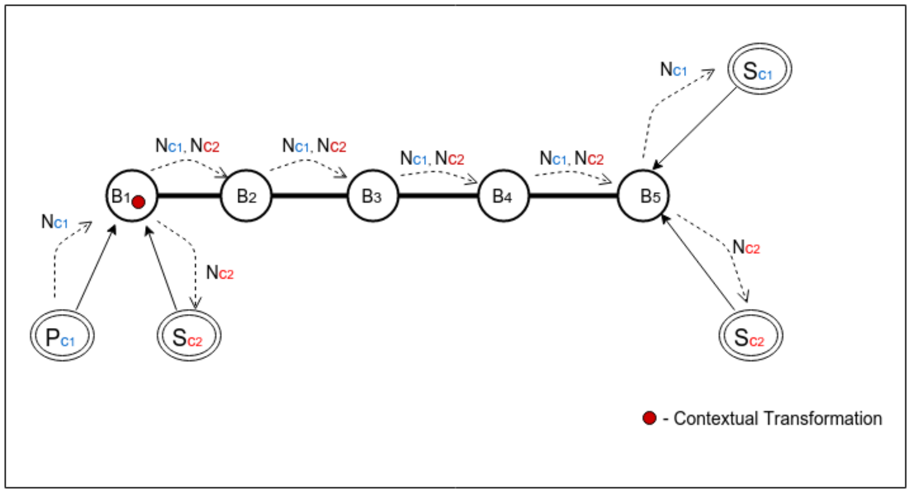

Figure 3: Contextual Transformation at B1 and propagation of Notification Nc2 along with Notification Nc1.

Case 3

The case 3 is similar to that of the case 2, except for the fact that another Subscriber Sc2 on the final broker is not available. In such a case the contextual transformation performed in the first broker is not propagated throughout the network. This is the second-key point, which shows that our approach does not propagate the contextually transformed notification when there are no other subscribers, post the contextual transformation.

Finally, as a side-note, with respect to Case-2/Figure 3, it might be encouraging to perform transformation on every broker with a Subscriber in different context, without propagating that extra transformed notification performed on the first broker but this would not scale as the number of contextual Transformation would drastically increase because of the dynamic nature of the EBS which could result in large number of Subscribers with different context in every or many brokers and hence large number of contextual transformation at each of these brokers that are connected to the subscribers which are in different context. Hence, we do not opt for this idea.

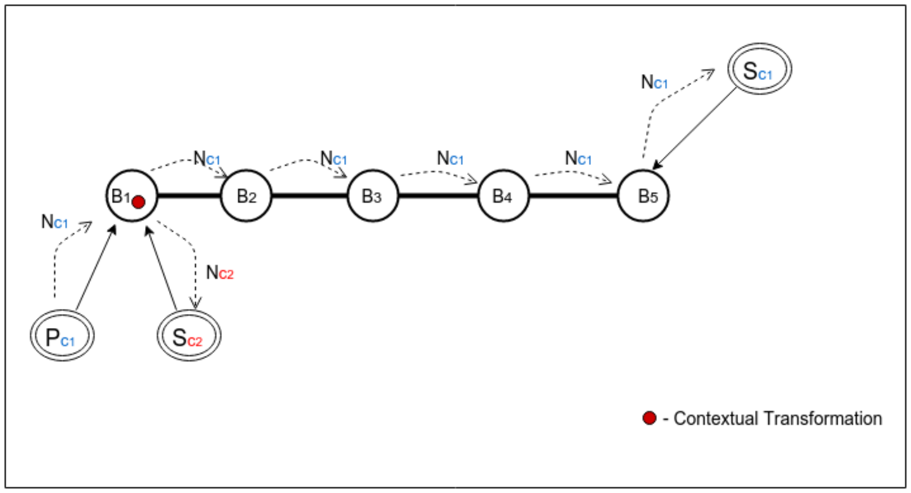

Figure 4: Depicting fewer notifications in Case 3 in comparison to Case 2.

	No. of Transformations	No. of Notifications
General Case (Fig. 1)	Null or Not required	7
Case 1	1	7
Case 2	1	12
Case 3	1	7

Table 1: Comparison of Various Cases

5 Future Work

In Future Work, we would like to assess our approach with respect to the presence of only local knowledge about the contexts at every broker. Another interesting thought, would be to include contextual subsumption in our approach, with an aim further improve our approach.