Parameterised Verification of Publish/Subscribe Networks with Exception Handling

Giorgio Delzanno

DIBRIS, University of Genova

RP 2019, Bruxelles

Summary of Contributions

- A Formal Model of Publish/Subscribe Networks
  - Single broker, multiple clients
  - Different forms of exception handling in the notification phase
  - Retained messages
  - Different ways of handling exceptions
- Some Decidability Results for Parameterised Verification based on WSTS/Petri Nets

Contents

Publish/Subscribe Systems

- Architecture
  - A broker is in charge of distributing messages to clients subscribed to certain topics
  - Individual nodes can perform operations such as subscription and push notifications
- Advantages
  - Multicast communication in open networks with heterogenous components, mitigation of security and privacy issues (data are published only by broker nodes)
- Application of Pub/Sub to IoT Systems
  - Communication among Edge Devices and Cloud (Micro)Services
  - Ensure Consistency in Distributed Storage and Microservices Architectures

Broker Architecture

- A broker must ensure
  - consistency of internal data structures
  - efficient data propagation (e.g. data structures indexed on topic names, client id's, etc)
  - robustness with respect to exceptions due to network/client failures
- We will consider different ways of exception handling using a Java implementation of a broker

Formal Model of Publish/Subscribe Networks

Publish/Subscribe Networks: A Broker Perspective

- Given
  - sets T (topics names), Q (client state), and M (message labels)
  - a client automata $P = \langle Q, q_0, R \rangle$
  - client transition labels taken from $A = \{local, (un)subscribe(s), publish(m, t) | s \in 2^T, m \in M, t \in T\}$
- The set N of configurations of a Pub/Sub Network S consists of multi-sets $\gamma = \{\!\{c_1, \ldots, c_k\}\!\}$ of client configurations
- The relation $\rightarrow \subseteq N \times N$ defines the broker semantics

Client Configuration

A client configuration c is a tuple $\langle q, s, b, f \rangle$, where

- $q \in Q$ is the current client state,
- $s \in 2^T$ is the set of topics for which the client is a subscriber,
- $b \in 2^M$ is the set of messages received so far
- $f \in \{\top, \bot\}$ is a flag that defines the connection status of the client with respect to the global network
  $\top =$ normal operating status, $\bot =$ disconnected

Broker Semantics: Local Operations

Local $\{\!\!\{\langle q, s, b, \top\rangle\}\!\!\} \oplus C \ \rightarrow\ \{\!\!\{\langle q', s, b, \top\rangle\}\!\!\} \oplus C$

Subscription $\{\!\!\{\langle q, s, b, \top\rangle\}\!\!\} \oplus C \ \rightarrow\ \{\!\!\{\langle q', s \cup s_1, b, \top\rangle\}\!\!\} \oplus C$
if $\langle q, \mathit{subscribe}(s_1), q'\rangle \in R$

Unsubscription $\{\!\!\{\langle q, s, b, \top\rangle\}\!\!\} \oplus C \ \rightarrow\ \{\!\!\{\langle q', s \setminus s_1, b, \top\rangle\}\!\!\} \oplus C$
if $\langle q, \mathit{unsubscribe}(s_1), q'\rangle \in R$

Disconnection $\{\!\!\{\langle q, s, b, \top\rangle\}\!\!\} \oplus C \ \rightarrow\ \{\!\!\{\langle q, s, b, \bot\rangle\}\!\!\} \oplus C$

Broker Semantics: Push Notification

- The broker acknowledges a request and, inside a synchronisation block, forwards the message to the clients subscribed to the topic
- Communication failures are captured locally via try-catch statements

```
boolean publish(String topic, String news, String sender)
throws Exception {
 ClientInterface client;
  synchronized(topicRelation) {
    Map<Integer, ClientInterface>
        subscriberList = topicRelation.get(topic);
    synchronized(subscriberList) {
     Iterator<Map.Entry<Integer, ClientInterface>> entries =
            subscriberList.entrySet().iterator();
      while (entries.hasNext()) {
        Map.Entry<Integer, ClientInterface>
            entry = entries.next();
        client = entry.getValue();
        try {   stub.send(topic,sender,news); }
        catch (RemoteException e) {
                System.out.println("Notification error);
        }
      }
    }
 }
 return true;
}
```

Broker Semantics: Global Operation

Publish $\{\!\{\langle q, s, m, \top\rangle\}\!\} \oplus \gamma \rightarrow \{\!\{\langle q', s, m, \top\rangle\}\!\} \oplus \gamma'$

if

- $\langle q, publish(m, t), q'\rangle \in R$,
- $\xi = E_\top(t, \gamma)$ (multiset of connected clients)
- $\mu = \gamma \ominus \xi$,
- $\gamma' = Add_m(t, \xi) \oplus \mu$ (m is added to t-subscribers in ξ)

This rule models exceptions handled locally, i.e., notifications reach *all* connected clients

Parameterized Verification

The Coverability Decision Problem

- Given
  - A Pub/Sub Network $\langle N, \rightarrow \rangle$ defined over the sets T, M, Q, A
  - A Client specification P
  - An ordering $\leq$ on Network Configurations
  - A set N_0 of Initial Network Configuration
  - A finite set of configuration $F \subseteq N$
- The Coverability Decision Problem consists in checking whether

$$N_0 \cap Pre^*(uc_{\leq}(F)) = \emptyset$$

where

- $Pre(C) = \{\gamma | \exists \gamma' \in C \ s.t. \ \gamma \rightarrow \gamma'\}$
- $uc_{\leq}(S) = \{\gamma' | \gamma \leq \gamma', \ \gamma \in S\}$

Decidability

The $\leq_n$ Ordering

- Given client configurations c_1, c_2, $c_1 = \langle q_1, s_1, b_1, f_1 \rangle \leq_c c_2 = \langle q_2, s_2, b_2, f_2 \rangle$ iff $q_1 = q_2$, $s_1 = s_2$, $b_1 \subseteq b_2$, and $f_1 = f_2$.
- Given Network Configurations γ_1, γ_2, $\gamma_1 \leq_n \gamma_2$ iff there exists an injective map h from the configurations in $\gamma_1 = \{c_1, \ldots, c_k\}$ to configurations in $\gamma_2 = \{d_1, \ldots, d_n\}$ such that $c_i \leq_c h(c_i)$ for $i : 1, \ldots, k$.

Decidability 1

The Coverability Decision Problem is decidable for Pub/Sub Networks equipped with ordering $\leq_n$

Proof

We first observe that the ordering $\leq_n$ is obtained embedding equality over finite sets and finite set inclusion into multiset inclusion.
By Higman Lemma's, the resulting ordering is a well-quasi-ordering
The transition relation $\rightarrow$ induced by a client specification P is monotone w.r.t. $\leq_n$, i.e., if $\gamma_1 \leq_n \gamma_2$ and $\gamma_1 \rightarrow \gamma_3$, then there there exists γ_4 s.t. $\gamma_2 \rightarrow \gamma_4$ and $\gamma_3 \leq_n \gamma_4$.
Given a finite set of configuration C it is possibile to compute a finite representation of $Pre(uc_{\leq_n}(C))$ via an encoding into transfer arc operations on Petri Nets
Decidability of coverability follows then from the general results on well-structured transition systems (WSTS)

Retained Messages: Push Notifications

$Push_r \quad \langle g, \{\!\{\langle q, s, m, \top\rangle\}\!\} \oplus \gamma\rangle \rightarrow \langle g', \{\!\{\langle q', s, m, \top\rangle\}\!\} \oplus \gamma'\rangle$

if

- $\langle q, publish(m, t), q'\rangle \in R$,
- $\xi = E_\top(t, \gamma)$,
- $\mu = \gamma \ominus \xi$,
- $\gamma' = Add_m(t, \xi) \oplus \mu$,
- $g'(t) = g(t) \cup \{m\}$, $g'(r) = g(r)$ for $r \neq t$

where g is a global map from T to 2^M

Retained Messages: Subscription

$$Subscribe_r \quad \langle g, \{\!\{ \langle q, s, b, \top \rangle \}\!\} \oplus \gamma \rightarrow \ \{\!\{ \langle q', s \cup s_1, b \cup g(s_1), \top \rangle \}\!\} \oplus \gamma \rangle$$

when $\langle q, subscribe(s_1), q' \rangle \in R$.

Decidability 2

The Coverability Decision Problem remains decidable for Pub/Sub Networks with retained messages and equipped with the $\leq_n$ ordering

Handling Exception Globally

Broker Internal Structure: Second Scenario

- Every invocation of the *publish* method is embedded into a try-catch statement to propagate error notifications to the server or to modify the current list of active clients.

```
boolean publish(String topic, String news, String sender)
    throws Exception {
 synchronized(topicRelation) {
     Map<Integer, InfoSub> subscriberList =
         topicRelation.get(topic);
    synchronized(subscriberList) {
        Iterator<Map.Entry<Integer, InfoSub>>
            entries = subscriberList.entrySet().iterator();
        while (entries.hasNext()) {
            Map.Entry<Integer, InfoSub>
                entry = entries.next();
            entry.getValue().send(topic,sender,news);
        }
    }
 }
 return true;
}
```

Modified Semantics

$$Publish_e \qquad \{\!\{\langle q, s, m, \top\rangle\}\!\} \oplus \gamma \rightarrow \{\!\{\langle q', s, m, \top\rangle\}\!\} \oplus \gamma'$$

if

- $\langle q, publish_e(m,t), q'\rangle \in R$
- $E_\top(t,\gamma) = \eta_r \oplus \eta_n \oplus \eta_f$ (multiset of connected clients)
  - η_r represents clients who receive the notification
  - η_f represents clients who fail during notification
  - η_n represents clients who do receive the notification and do no fail
- $\mu = \gamma \ominus (\eta_r \oplus \eta_f)$
- $\gamma' = Add_m(\eta_r) \oplus Up_\perp(\eta_f) \oplus \mu$

where $Up_\perp(\eta_f)$ sets all connection flags in η_f to $\perp$

Decidability 3

The Coverability Decision Problem is decidable for Pub/Sub Networks with *publish*$_e$ semantics and equipped with the $\leq_n$ ordering

Proof sketch

- Flatten configuration (move topic and messages in control states)
- Counter representation as a symbolic representation of global configurations as in Petri Nets
- To model $publish_e$ we associate an auxiliary variable $AuxX$ to each counter X and a gadget to move a non-deterministic amount of tokens from X to $AuxX$.
- Gadgets can be pipelined in a globally locked series of transitions in order to simulate the non-deterministic split of active clients in three classes (receive notification, do not receive notification, fail)
- Reduction to coverability of a Petri Net

Conclusions

Conclusions

- Contributions
  - A Formal Model of Publish/Subscribe Network with different ways of handling communication exceptions
  - Reductions to (extended) Petri nets for obtaining decidable fragment
- Possible Extensions
  - Refinement of the broker model (e.g. internal data structures, asynchronous phases)
  - Federation of brokers (e.g. broker hierarchies)
  - Other protocol properties (e.g. success of notifications?)