Model-based Testing IoT Communication via Active Automata Learning

Martin Tappler Bernhard K. Aichernig
Institute of Software Technology
Graz University of Technology, Austria
{martin.tappler, aichernig}@ist.tugraz.at

Roderick Bloem
Institute of Applied Information Processing and Communications
Graz University of Technology, Austria
roderick.bloem@iaik.tugraz.at

ICST `16 (International Conference on Software Testing, Verification and Validation)
Presenter: Dongwoo kim

Software Safety Engineering LAB

Motivation

- Active Automata Learning과 같은 Black-box System의 행위를 파악하기 위한 연구가 활발히 진행되고있다.
- Active Automata Learning은 Black-box System의 행위를 파악하고 Automata 형식의 모델을 생성한다.
  - e.g.) Java LearnLib
- 같은 specification을 공유하는 다수의 implementation이 있을 경우 Learning을 통하여

  각 implementation을 학습하고 서로 비교할 수 있다.
  - e.g.) MQTT protocol implementations, Kakaotalk GUI
- 비교 결과 차이점이 발견되면 Implementation 다수 또는 하나에 오류가 있음을 알 수 있다.

Approach

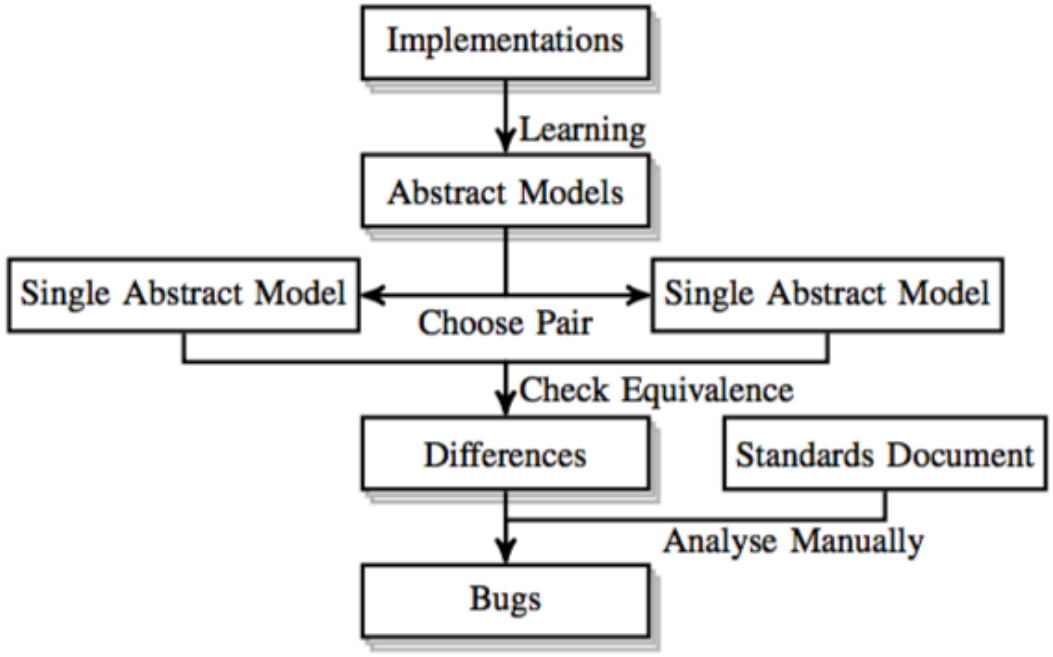


Fig. 1. Overview of bug-detection process.

- Learning-based method를 이용
- IoT 기반의 reactive system의 failure case를 탐지
- 방법

1. 기존에 구현된 여러 시스템의 행위를

   Active Automata Learning을 이용하여

   Mealy machine 형식의 Abstract Model로 추출

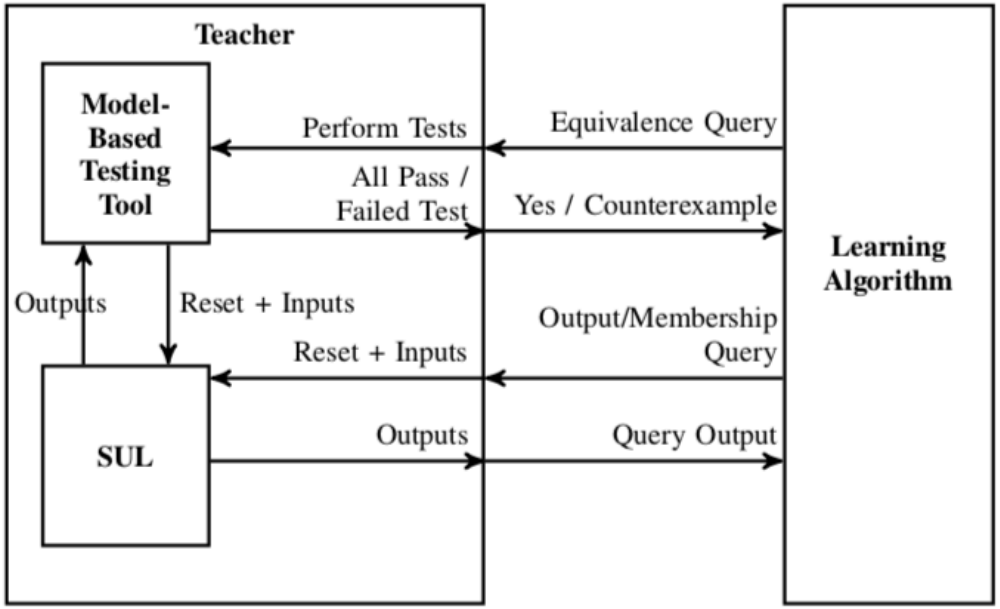
2. 추출된 model 간의 equivalence를 확인

   하여 동일하지 않은 경우 fault가 있다고 의심

3. 수작업으로 동일하지 않은 행위에 대해서

   Specification을 참조하여 분석하고 fault 여부 확인

Active Automata Learning



<Active Automata Learning 개념도>

- Learning algorithm이 Teacher에게 질의하여 모델을 생성
- 생성된 모델은 Mealy machine의 형태
- **다음의 세가지 질의를 이용하여 Learning을 수행**
  - reset: SUL를 초기화
  - output query: input을 제공하여 어떤 output이 출력되는지 확인
  - equivalence query: learned model이 충분히 학습되었는지 확인
    - 몇 가지 input을 이용하여 learned model이 같은 결과를 출력하는지 확인

Equivalence Checking

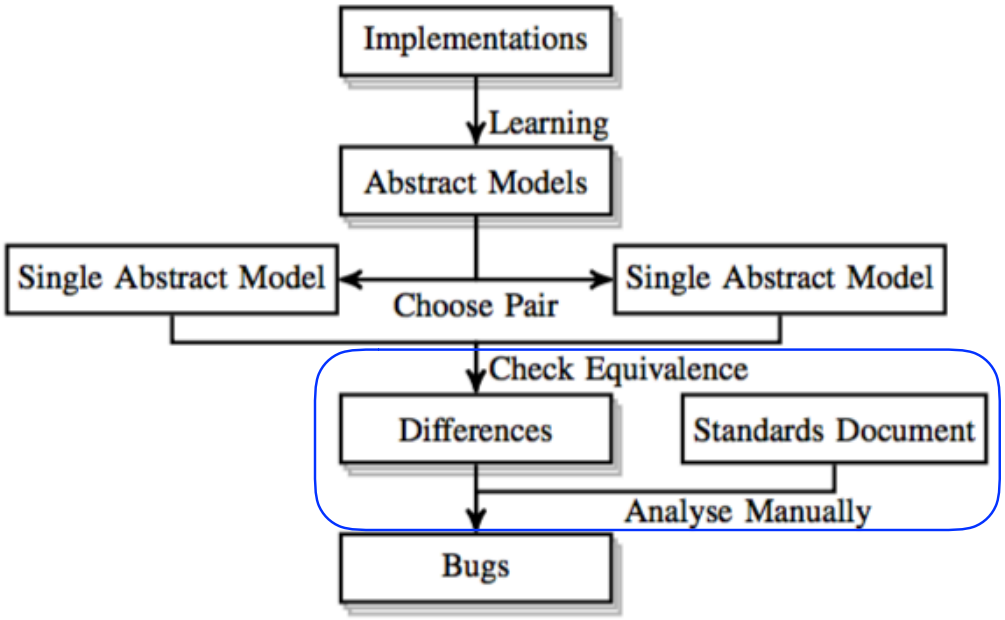


Fig. 1. Overview of bug-detection process.

- Bi-simulation Checking
- 두 모델이 동일하다는 결론을 내리거나 반례를 탐지할 수 있다.
- 반례는 한 모델 또는 두 모델 모두에 이상이 있음을 나타낸다.
- Bi-simulation checking 방법

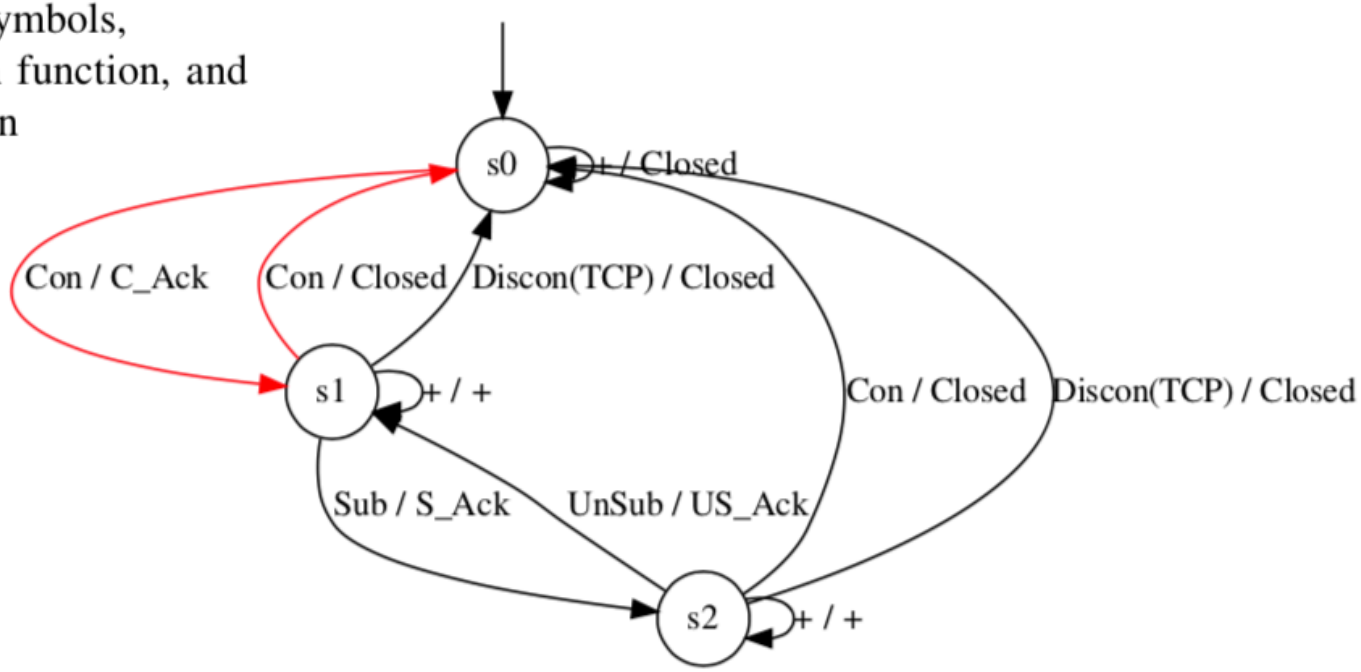
1. 각 상태가 두 모델의 상태의 pair가 되는 Product graph 생성
2. equivalent하지 않음을 나타내는 fail-state 추가
3. 두 모델 모두에 같은 transition(input/output이 동일)이 있으면 product graph에 추가
4. 둘 중 하나의 모델에만 존재하는 transition은 fail state로 연결
5. 반례가 발견된 경우 fail state에 도달하는 state 시퀀스를 보고

Mealy Machines

Definition III.1 (Mealy Machines). A Mealy machine is a 6-tuple $\langle Q, q_0, I, O, \delta, \lambda \rangle$ where

- Q is a finite set of states,
- q_0 is the initial state,
- I/O is a finite set of input/outputs symbols,
- $\delta : Q \times I \to Q$ is the state transition function, and
- $\lambda : Q \times I \to O$ is the output function

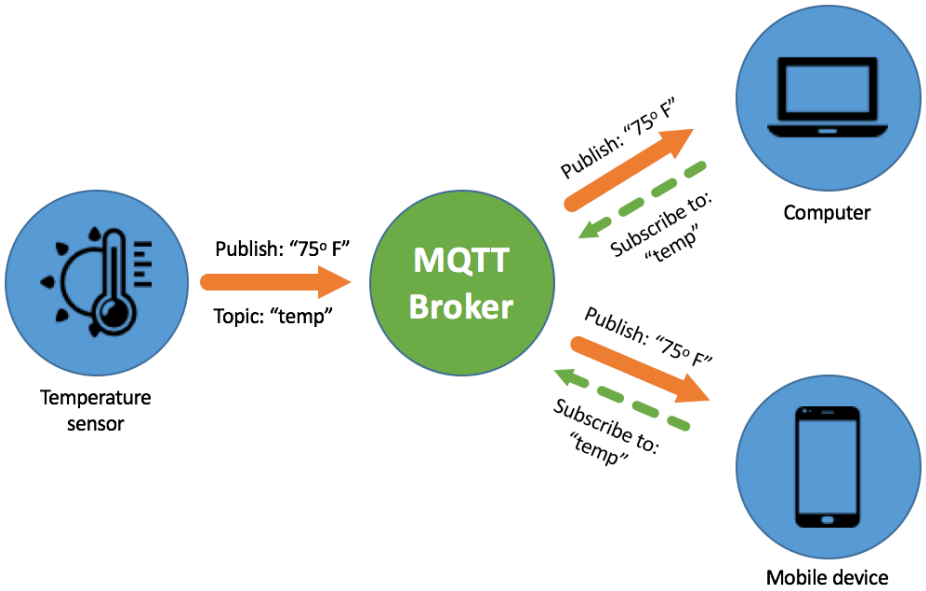
- 상태 Q = {s0, s1, s2}
- 시작 상태 q0 = s0
- I/O = {Con, C_Ack, Closed, …}
- $\delta = \{(s0, Con) \to s1, ...\}$
- $\lambda = \{(s0, Con) \to C\_Ack, \ldots\}$



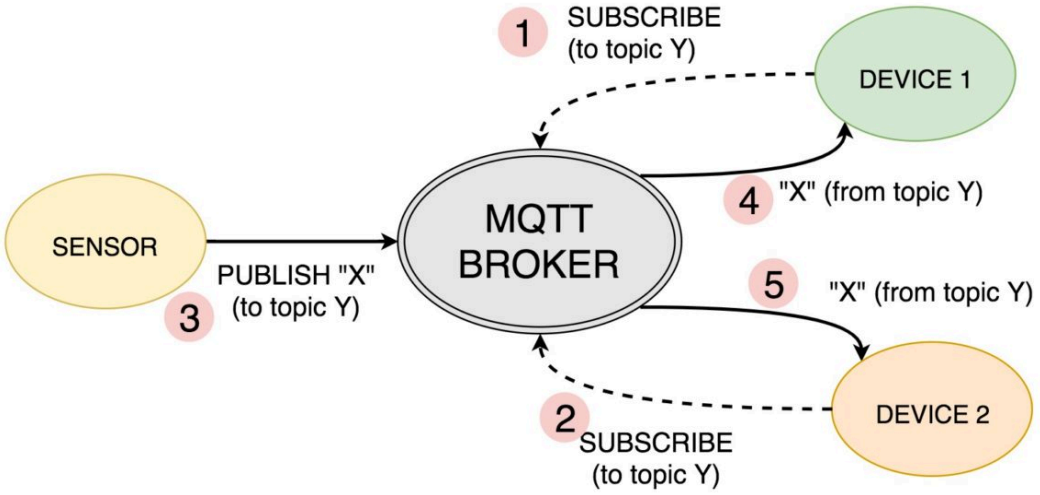
<Example of Mealy Machine>

Case study 대상 MQTT Protocol

- Light-weight publish/subscribe protocol
- Publisher : 메시지 전송을 수행 / Subscriber : 메시지 수신
- Publisher가 전송할 메시지를 Broker에 등록하면 Broker가 Subscriber에게 Forward
- Well-suited in resource constrained environment such as IoT
- 본 연구에서는 MQTT Broker implementation 5종에게 Case study를 수행함

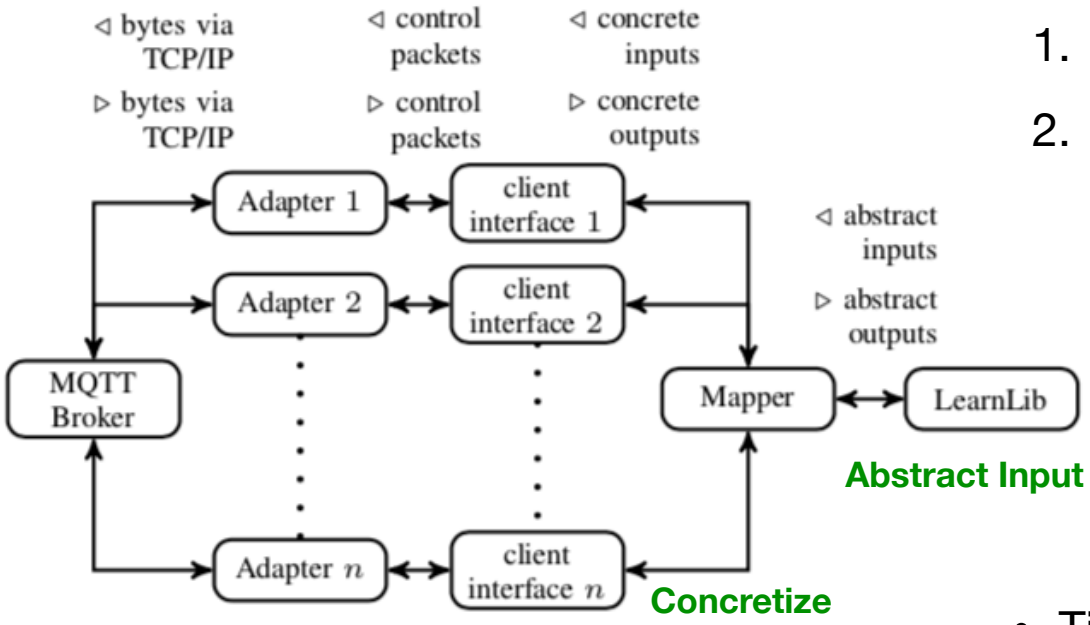


<Broker와 Client의 관계>



<Message publish 과정>

Environment 구성



<MQTT implementation을 학습하기 위한 환경>

- 모든 구성 환경과 시나리오를 테스트할 수 없으므로
  본 연구에서는 7가지 환경을 구상하여 테스트 진행

1. Simple: 하나의 client / 기본적인 publish & subscribe
2. Two clients with Retained will: 두 개의 client 존재
   - 하나의 client c1이 topic p1에 will을 남기고 비성장 종료
   - 다른 client c2가 p1을 subscribe

Abstract Input 선택

- Timeout: 일정 시간 내에 오지 않는 메시지는 empty로 간주
- Output: 임의로 순서로 Mapper에 도착하는 메시지는 정렬
- Restrictions: 시간과 관련된 행위는 mealy machine으로 충분히 표현할 수 없기 때문에 제외 (ex.ping)

Experiment

- Implementation
  - Scala / Java-library LearnLib
  - Mapper는 직접 구현 - 7종
- Experiment
  - MQTT protocol을 구현한 implementation 5종으로부터 모델을 생성
  - Case study를 통해서 17개의 버그를 발견 (Mosquitto에서는 발견되지 않음)
    - 이 중 네 개는 이미 reported 나머지는 수정되었거나 검토 중

- Apache ActiveMQ 5.13.3[2]
- emqttd 1.0.2 [3]
- HBMQTT 0.7.1 [4]
- Mosquitto 1.4.9 [5]
- VerneMQ 0.12.5p4 [6]

Implementation	Timeout in Milliseconds
Apache ActiveMQ	300
emqttd	25
HBMQTT	100
Mosquitto	100
VerneMQ	300

각 implementation은 다른 timeout 수치를 사용하며
이 수치를 timeout bound로 사용

Experiment (1)

- Simple Mapper
- 입력 : Connect, Connect
- 출력 - Mosquitto : C_ACK, Closed / HBMQTT : C_ACK Empty

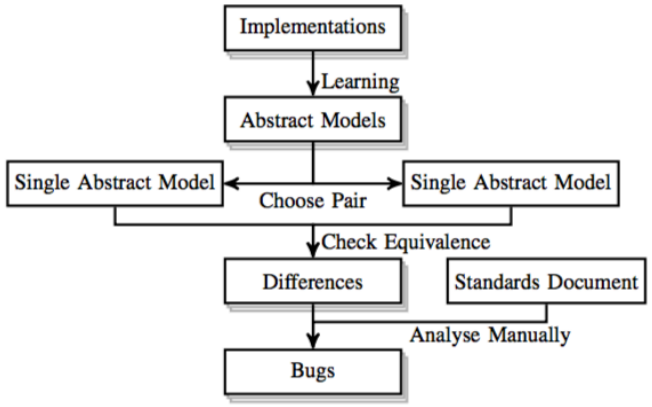
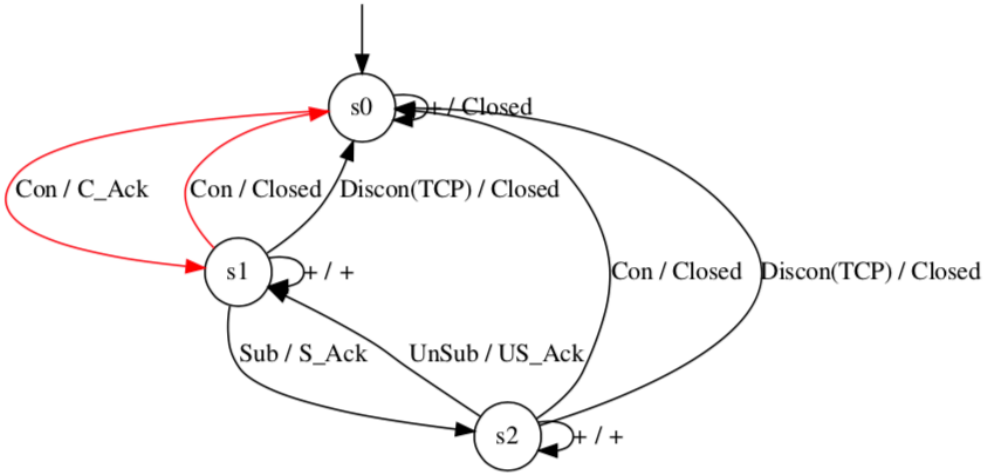


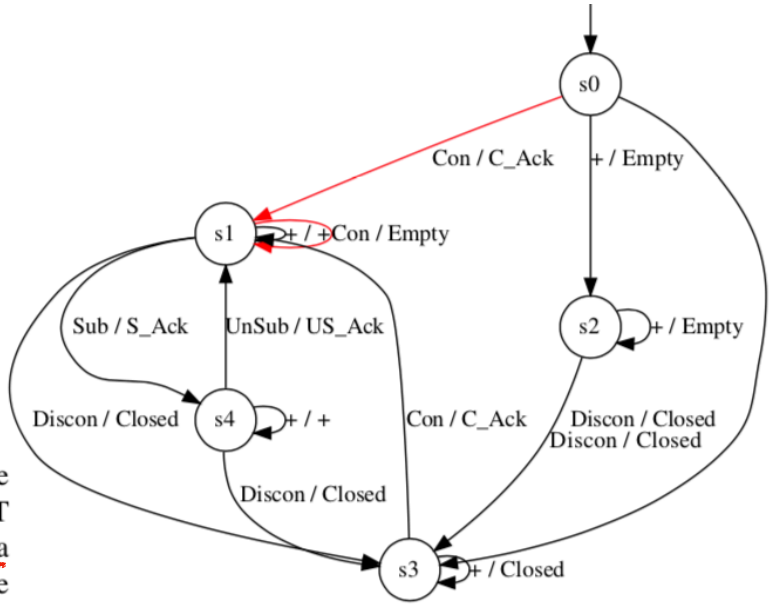
Fig. 1. Overview of bug-detection process.



(a) Mosquitto model

A Client can only send the CONNECT Packet once over a Network Connection. The Server MUST process a second CONNECT Packet sent from a Client as a protocol violation and disconnect the Client [MQTT-3.1.0-2].

Spec에 따라 Mosquitto가 바른 행위를 보여주었음을 확인



(b) HBMQTT model

Experiment (2)

- Mapper : Two Clients with Retained Will
- 입력 : Connect, Connect
- 출력 - Mosquitto : bye / <else> : empty

1) A client connects with client identifier `Client1`
2) A client connects with client identifier `Client2` with retained will message `bye` for topic `c2_will`
3) `Client2` disconnects unexpectedly (such that the will message is published)
4) `Client1` subscribes to `c2_will`
5) `Client1` subscribes to `c2_will`

// 다섯번째 step에서 Mosquitto와 emitted,ActiveMQ가 다르게 행동

<버그를 발견한 시나리오>

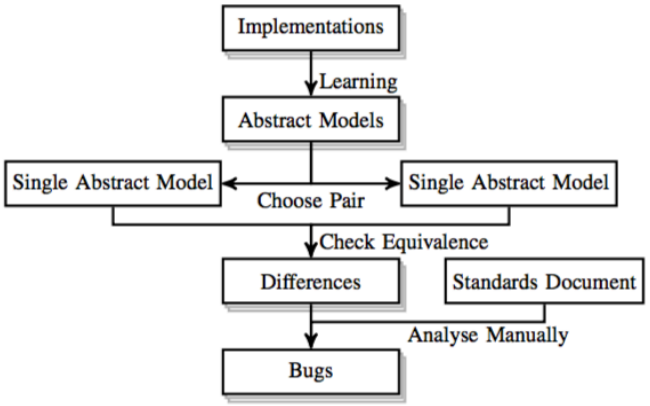


Fig. 1. Overview of bug-detection process.

- Mosquitto는 5번째 step에서 bye를 전송, 나머지는 empty를 전송
- 표준에 따라 Mosquitto가 바른 행위를 하는 것을 확인

cording to [MQTT-3.8.4-3] [14] which states that repeated subscription requests must replace existing subscriptions and that "any existing retained messages matching the Topic Filter MUST be re-sent".

- 표준에 따르면 subscription을 반복할 시에 will message를 반복해서 보내주어야한다고 명시

Experiment (2)

- 가장 큰 모델은 18개의 상태를 가짐
- 한 번의 query에 소요되는 실행시간이 크기 때문에 모델 생성에 오랜 시간이 걸린다.
- ActiveMQ나 VerneMQ의 경우 Timeout 수치가 크기 때문에 항상 많은 시간이 소요
  - domain-specific optimization, smart test selection 등을 활용하여 개선 가능

	ActiveMQ	emqttd	HBMQTT	Mosquitto	VerneMQ
# states	4	3	5	3	3
MQ time[s]	59.72	3.87	31.94	14.01	43.91
MQ # queries	88	59	110	56	57
CT time[s]	914.18	78.3	491.06	278.21	915.77
CT # queries	525	519	482	487	490
# equivalence queries	4	3	4	3	3

<Simple Mapper>

	ActiveMQ	emqttd	HBMQTT	Mosquitto	VerneMQ
# states	18	18	17	18	17
MQ time[s]	1855.55	167.32	557.14	641.89	1570.8
MQ # queries	732	735	640	730	625
CT time[s]	4787.92	481.36	2022.47	1612.59	4355.97
CT # queries	672	816	613	670	658
# equivalence queries	13	12	11	9	11

<Two clients with Retained Will Mapper>

Implementation	Timeout in Milliseconds
Apache ActiveMQ	300
emqttd	25
HBMQTT	100
Mosquitto	100
VerneMQ	300

<Time out for each implementations>

Conclusion

- 학습기법을 통해 Black-box 시스템으로부터 버그를 탐지하는 방법을 소개
- Active Automata Learning을 통해 MQTT Protocol의 Implementation 들을 학습하고 Equivalence를 비교함으로써 Implementation의 비정상 행위 탐지
- 5개의 MQTT Implementation으로부터 학습한 모델을 서로 비교해본 결과 총 18개의 행위 불일치를 탐지할 수 있었음
- 제안한 방법의 수행 속도는 각 implementation의 runtime에 영향을 받아 소요 시간의 편차가 크고 학습에 오랜 시간이 걸리는 것을 확인