MQTT 프로토콜 기반 협업 메신저 시스템 설계 및 구현

황현천*, 박지수**, 손진곤*
*㈜한국방송통신대학교 대학원 정보과학과
**고려대학교 컴퓨터교육
e-mail:andy74.hwang@gmail.com

Design and Implementation of Collaboration Messenger System based on MQTT Protocol

Hyun Cheon Hwang*, Ji Su Park**, Jin Gon Son*,
*Dept. of Computer Science, Graduate School, Korea National Open University
**Dept. Computer Science Education, Korea University

요 약

현재 네트워크 시스템은 유선에서 무선으로 발전하면서 많은 업무가 모바일 기반의 업무로 변화되고 있다. 그 중 대표적인 것이 모바일 오피스 시스템으로 중요 구성요소 중 하나인 커뮤니케이션은 구성원간의 단문 메시지 서비스(SMS)이다. 그러나 최근 무선 환경의 발전으로 단문 메시지 전송 방식이 SMS에서 모바일 메신저로 변화하고 있으나 그에 대처하지 못한 시스템들이 있다. 이에 본 논문에서는 모바일 오피스 시스템에서 사용되고 있는 SMS를 대체하여 MQTT(Message Queuing Telemetry Transport) 프로토콜을 이용한 협업 메신저 시스템을 설계 및 구현한다.

1. 서 론

최근 무선통신의 발전으로 웹에서의 정보 공유는 데스크톱에서 모바일로 변화하고 있다. 특히 모바일 데이터의 트래픽은 2013년 전년 대비 약 81% 증가하였으며, 전 세계적으로 2018년까지 2013년 대비 약 11배인 15.9EB의 데이터 증가가 예상된다[1]. 이러한 추세로 많은 서비스들이 모바일 디바이스를 기준으로 제공되고 있으며, 그룹웨어 서비스, 현장업무 서비스와 같은 업무들이 모바일 오피스 시스템으로 이루어지고 있다. 모바일 오피스 시장의 성장력은 전체 모바일 시장의 70% 이상을 차지하고 있으며, 2017년까지 약 6조원 이상의 규모로 성장할 것으로 예상된다[2].

최근 커뮤니케이션을 위한 서비스는 SMS에서 모바일 메신저 변화되고 있다. 그러나 모바일 오피스 시스템에서는 구성원들에게 특정 메시지를 보내기 위해서 SMS를 사용한다. SMS는 단문 메시지(한글80자)만 가능하며, 장문 및 멀티미디어 정보 전송에는 어려움이 있으며, 메시지 당 과금 체계로 상대적으로 많은 비용 및 텍스트 길이 제한이 되며, 메시지에 대한 기밀성이 없는 단점을 가지고 있다. 이러한 단점으로 인터넷망을 이용한 XMPP, MQTT, HTTP, CoAP등의 프로토콜을 이용한 메시지 전송 시스템을 구축하고 있으며, 이 중 MQTT 프로토콜은 경량과 고성능의 특징으로 모바일 어플리케이션에서 사용되고 있다. 그러나 다자간 그룹 메시지 전송을 할 경우, 메시지의 전송시간이 동기화 되지 않아 도착순서가 맞지 않을 수 있기 때문에 다자간 그룹 메시지 전송 시 메시지의 도착 순서를 고려해야 한다.

본 논문에서는 모바일 오피스 시스템에서 구성원간의 커뮤니케이션을 위해 MQTT 프로토콜 기반 협업 메신저 시스템을 설계하고 구현한다.

2. 기존 사례 및 연구

2.1. 모바일 메시지 전송 프로토콜

기존의 SMS를 이용한 모바일 메시지 전송 방식은 모든 디바이스에서 작동하는 장점은 있으나, 메시지 당 과금 체계로 상대적으로 많은 비용 및 텍스트 길이 제한이 되며, 메시지에 대한 기밀성이 없다[3]. 이로 인해 SMS 대신 모바일 메시지 전송 시스템을 점차 사용하고 있다. 이러한 모바일 메시지 전송 시스템은 폴링 방식과 푸시 방식의 두 가지를 사용한다. 폴링 방식은 주기적으로 서버에 접속해서 새로운 메시지가 있는지를 확인하고 데이터를 가져온다. 푸시 방식은 모바일 서버에서 클라이언트 어플리케이션으로 새로운 데이터가 발생할 때 마다 내려 보내주는 푸시 방식이다. 폴링 방식은 주기적으로 서버에 접속해야 하는 불편함과 접속에 필요한 데이터 낭비 등으로 인해 푸시 프로토콜 방식으로 점차 변화되고 있는 추세이다.

<표 1> 모바일 메시징 시스템 간 차이점

SMS	Polling Mobile Application	Push Mobile Application
- 모든 디바이스 지원 - 메시지 당 과금 체계 - 단문 메시지로 사용상의 제한 - 메시지에 대한 기밀성이 없음	- 네트워크 데이터 사용 - 주기적으로 서버에 접속하여 메시지 확인	- 네트워크 데이터 사용 - 서버에서 메시지 발생 시 단말기로 전송 - 폴링 방식보다 저비용

2.2 MQTT 프로토콜

MQTT 프로토콜은 IBM에서 발표한 메시지 푸시 프로토콜이다[3][4]. MQTT 프로토콜은 낮은 대역폭 및 긴 지

연시간에도 신뢰적인 메시지를 전송되도록 설계되었다. 현재 Facebook 메신저 어플리케이션도 MQTT 프로토콜을 이용한다[5]. 그러나 푸시 프로토콜의 특성상 네트워크의 연결 불안정 등과 같은 경우에 전송시간이 동기화되지 않아 그룹 메시지 전송 시 메시지의 순서가 맞지 않는다[6]. 협업 메신저 시스템에서 금융 메시지등과 같은 경우에는 메시지의 트랜잭션 유지가 매우 중요하다. 트랜잭션을 유지하기 위해서는 메시지는 원자성, 성능, 순서화를 유지해야 한다[7]. 또한 협업 메신저 시스템에서는 협업을 위해 커뮤니케이션 이력 및 내용을 조회할 수 있어야 한다. 그러나 MQTT 프로토콜 기반의 메시지 전송 시스템에서는 메시지의 전달만을 목적으로 사용하게 되므로 보관하지 않는다. 따라서 신뢰적인 협업을 위한 메시지를 보관하고, 누락시 재전송을 해야 한다.

MQTT 프로토콜은 경량의 메시지 전송 프로토콜로써 publish/subscribe 방식 기반의 메시지 브로커이다. 특정 주제에 대해 다자간 채팅이나 메시지를 서로 공유하고자 하는 경우 구독자는 구독신청(subscribe)을 하고, 발행자는 발행(publish)을 하는 구조이다. 이러한 주제를 토픽(topic)이라고 한다. 토픽은 하위 토픽을 계층적으로 가질 수 있으며, 슬래시(/)를 이용해서 계층적 구조로 표현한다[4][8].

본 논문에서는 메시지를 서버에 저장 후 메시지의 순서와 전송 확인 여부를 보장하는 MQTT 프로토콜 기반 협업 메신저 시스템을 설계 및 구현한다. 모바일 협업 메신저 시스템 구현을 위해 MQTT 푸시 서버는 오픈 소스인 mosquitto를 사용한다.

3. MQTT 기반 협업 메신저 시스템 설계 및 구현

3.1. MQTT 기반 협업 메신저 프로토콜 설계

MQTT 프로토콜은 MQTT 헤더(header)와 페이로드(payload) 영역으로 구성된다. 헤더는 2byte의 고정된 크기와 구독 및 연결을 위한 추가 바이트(12바이트)로 구성된다. 페이로드 영역은 전송될 메시지이다[4][9]. 본 논문에서는 협업 메신저 시스템에서 데이터 순번을 검증 할 수 있는 페이로드 영역을 확장한다. 확장 페이로드는 다음 <표 2>와 같이 설계한다.

<표 2> 협업 메신저 시스템을 위한 페이로드 구성

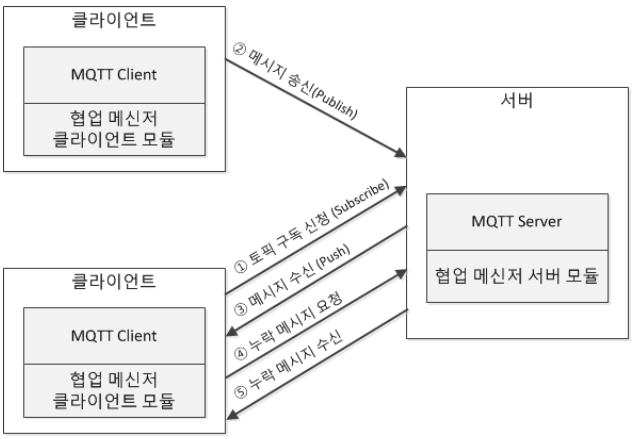
항목	ID	SEQ	QoS	원본메시지
byte	4	5	1	n

<표 2>에서 ID는 토픽을 구분한다. SEQ는 하나의 토픽 주제에서 일어나는 메시지의 순서를 나타내며, 서버에 도착한 시간을 기준으로 순번을 부여한다. QoS는 메시지의 전송 방식을 결정한다. 메시지의 순서에 따라 누락된 메시지를 먼저 수신할 것인지 또는 메시지 순서를 무시하고 도착 메시지를 즉시 수신할 것인지를 결정하는 플래그이다. 원본 메시지는 클라이언트에서 송신된 원본 메시지를 나타낸다.

3.2 협업 메신저 시스템 구성

협업 메신저 시스템은 서버 클라이언트 구조로 이루어지며, MQTT 프로토콜을 이용하여 설계 및 구축한다. 협업 메신저 시스템의 구성도는 다음 (그림 1)와 같다.

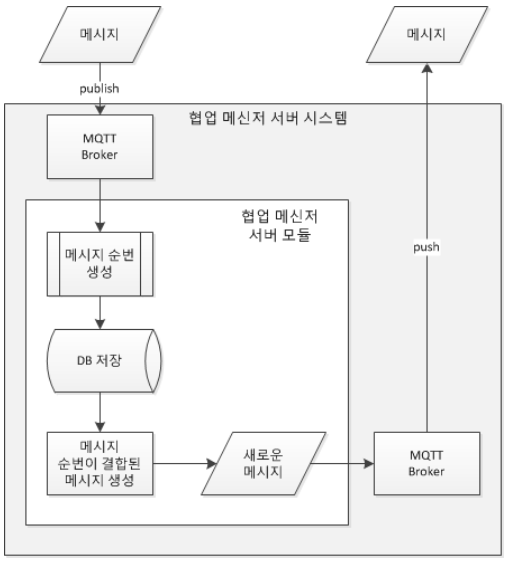
클라이언트에서는 토픽을 구독 신청하고, 클라이언트에서 토픽에 포함되는 메시지는 ID를 결합한 메시지로 구성하여 서버로 전송한다. 서버에서는 수신된 메시지와 메시지의 SEQ와 서버에 설정된 QoS를 이용하여 클라이언트에서 검증할 수 있는 메시지 형태로 재구성하여 클라이언트로 전송한다. 클라이언트에서는 도착한 메시지의 SEQ 및 QoS를 확인하고, 직전에 도착한 메시지 순번과 현재 메시지 순번을 비교하여 누락된 메시지가 존재하는지 확인하여 처리한다.



(그림 1) 협업 메신저 시스템 흐름도

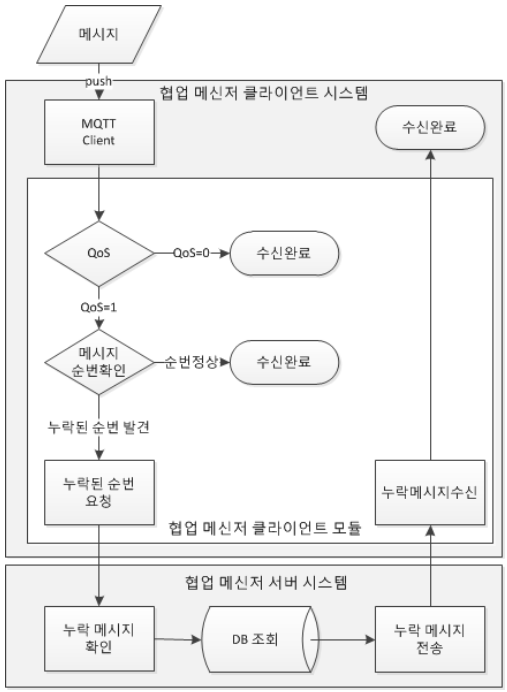
3.3 협업 메신저 클라이언트/서버 모듈 설계

협업 메신저 서버 모듈에는 클라이언트에서 발행된 메시지를 데이터베이스에 저장하고, SEQ와 QoS가 결합된 메시지로 변환 후 각 클라이언트에 푸시한다. 협업 메신저 서버 모듈에서 메시지 발생순서는 다음 (그림 2)과 같다.



(그림 2) 협업 메신저 서버 모듈 순서도

협업 메신저 클라이언트 모듈에서는 메시지를 수신시 QoS를 확인하여 값이 0이면, 메시지를 바로 보여주고, 값이 1이면 순번 확인을 한다. 또한 순번 확인 시 메시지 누락이 발견된다면, 누락 메시지를 서버에 재요청을 함으로써 누락된 메시지를 재전송 받는다. 협업 메신저 클라이언트 모듈의 메시지 처리순서는 다음 (그림 3)와 같다.



(그림 3) 협업 메신저 클라이언트 모듈 순서도

3.4. 협업 메신저 시스템 구현

협업 메신저 시스템은 python과 MQTT 라이브러리를 이용하여 서버/클라이언트를 구축한다. 또한 페이로드 영역인 메시지를 저장하기 위한 데이터베이스는 MySQL을 사용하며 테이블의 구조는 다음 (그림 4)와 같다.

```
CREATE TABLE MSG (
    idx INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID VARCHAR(255) NOT NULL,
    QOS INT NOT NULL,
    SEQ INT NOT NULL,
    MSG VARCHAR(255) NULL
    )
```

(그림 4) 협업 메신저 시스템의 데이터베이스 테이블 구조

협업 메신저 서버 모듈은 ID를 가지고 있는 토픽만 변환 및 전송한다. 해당 토픽의 수신은 getQoS()와 getSeqNumber()를 통해 QoS 및 Seq를 확인하고, QoS, Seq가 결합된 토픽으로 재생성 후, mqttc.publish()를 통해 클라이언트로 송신한다. 협업 메신저 클라이언트 모듈은 ID를 결합한 토픽 형태로 송신하고, 수신 받은 토픽에 대해서는 processCoworkClientModule()를 통해 토픽의 QoS와 Seq를 추출 및 검증할 수 있도록 구현한다.

4. 실험

4.1. 실험 환경

본 실험에서는 MQTT 프로토콜 및 협업 메신저 시스템을 성능 검증한다. 이를 위해 데이터의 응답시간, 서버의 가동률 확인 및 클라이언트 메시지가 오류 없이 전달되는 것을 실험한다. 이를 위한 서버 및 클라이언트의 실험 환경은 다음 <표 3, 4>와 같다.

<표 3> 서버 실험 환경

구 분	사 양
CPU	64bit 2.2GHz X 8 Core
Memory	0.5 Gbyte
OS	CentOS v5.1
RDBMS	MySQL v5.0.95
message broker	mosquitto v1.2.3
python	python v2.7.3

<표 4> 클라이언트 실험 환경

구 분	사 양
CPU	Intel Core i5-3210M 2.5GHz X 4 Core
Memory	4 Gbyte
OS	Microsoft Windows 7
mosquitto	mosquitto v1.2.3
python	python v2.7.3

4.2. 실험 결과 및 분석

MQTT 프로토콜의 성능을 검증하기 위해서 20byte로 구성된 메시지를 클라이언트에서 0.1초 간격으로 송신 및 수신한다. 클라이언트는 동시에 10개를 운영하여 서버의 자원 사용량을 측정 하였다.

MQTT 프로토콜의 실험에서는 서버의 CPU 점유율이 5%가 넘지 않는 안정적인 작동 결과를 보였다. 협업 메신저 시스템에서는 서버의 CPU 사용률은 1% 미만으로 줄었다. 그러나 실제 메시지를 기록하는 MySQL과 메시지의 순번을 관리하는 python의 CPU 사용률이 각각 22%, 2.6% 정도의 CPU 점유율을 보였다. 이는 메시지 순번을 처리하기 위한 협업 메신저 모듈이 자원을 소비하고 있는 것을 보여주는 것으로 이를 최적화하면 더 좋은 성능을 보여줄 수 있다.

<표 5> 각 프로세스 별 CPU 사용률 (sec)

mosquitto	MySQL	python
0.46	21.09	2.58

메시지 평균 응답 시간을 실험한 <표 6>에서는 메시지의 평균 응답시간이 1 sec 이내로 측정된다. 논문[10]에 따르면 사람이 즉각적인 반응이라고 느끼는 속도가 1초이다. 따라서 본 논문의 실험결과인 메시지 순번 처리를

위한 지연 시간(0.1x초)은 사람간의 협업을 위해 메시지 전송을 하는 환경에서는 문제시 되지 않는다.

<표 6> 메시지 평균 응답 시간 (sec)

메시지 전송 간격시간	0.1	0.2
협업 메시지 시스템 응답시간	0.37	0.32
MQTT 프로토콜 응답시간	0.20	0.23

또한 본 실험에서 서버를 통해 클라이언트로 전송된 메시지는 <표 7>과 같이 모두 누락 없이 시퀀스 번호 순서대로 전달된다.

<표 7> 메시지 누락율

메시지 전송 간격	0.1 sec	0.2 sec
메시지 개수	1,000 개	1,000 개
누락율	0 %	0 %

5. 결론

본 연구에서 협업 메신저 시스템은 MQTT 프로토콜을 기반으로 하여 빠른 전송 뿐 만이 아니라 메시지의 순번을 보장하여 실제 업무 환경에서 사용할 수 있도록 설계 및 구현을 하였다.

실험에서는 서버의 자원이 조금 더 사용되긴 하나, SMS 시스템보다는 저비용이며, SMS에서 불가능한 장문(80자 이상)의 메시지의 전송도 가능하다. 또한 메시지 전송 시, 순차적이며 긴 지연 시간 없이 메시지가 전달되는 것을 보장한다. MQTT 프로토콜만을 이용하였을 때 메시지가 휘발성으로 삭제되는 메시지와 달리 협업 메신저 시스템의 데이터베이스에 저장된 메시지를 재확인할 수 있다.

향후 연구에는 클라이언트가 특정 메시지 주제에 대한 구독 전에 발생한 과거 메시지의 수신 방법을 추가하여 모바일 디바이스에서 구현한다.

참고 문헌

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[2] 미래창조과학부, "모바일 오피스 정보보호 안내서," vol. KISA 안내·해설 제2013-16호, ed: 미래창조과학부,인터넷진흥원, 2013.

[3] 지용득, "모바일 메시징 솔루션 MQTT," ed. Impact Korea 2012: IBM, 2012.

[4] K. Tang, Y. Wang, H. Liu, Y. Sheng, X. Wang, and Z. Wei, "Design and Implementation of Push Notification System Based on the MQTT Protocol," in 2013 International Conference on Information Science and Computer Applications (ISCA 2013), 2013.

[5] Facebook Article, https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

[6] 네이버 기술전략팀 Article: http://helloworld.naver.com/helloworld/1846

[7] A. Schiper and M. Raynal, "From group communication to transactions in distributed systems," Communications of the ACM, vol. 39, pp. 84-87, 1996.

[8] 이신호, 김현우, and 주홍택, "모바일 통합 SNS 게이트웨이의 상위 구조 및 MQTT 기반의 푸시 알림 프로토콜 설계," 한국통신학회논문지, vol. 38, pp. 344-354, 5 2013.

[9] IBM MQTT Protocol Specification, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

[10] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?," Behaviour & Information Technology, vol. 23, pp. 153-163, 2004.