분류없음

ZeroMQ 정리

효자손 효자발 2017.04.20 20:18

문서를 읽고 특징을 파악하는데 예제가 너무 간단하나, 내용이 너무 방대하여 중요 요점을 기록하기 위하여 본 페이지를 작성한다.
>> 이후 정리된 내용을 다시 깨끗하게 정리할 예정이다.

1. BSD 소켓과 마찬가지로 4개의 단계를 거쳐 연결을 수립한다.

- 소켓을 생성하고 파괴한다. 문서에선 숙명적인 소켓의 삶의 순환이라 한다. zmq_socket(), zmq_close()를 통해 수행한다.
- 옵션을 지정하여 소켓의 특성을 정의하며 필요시 조회한다. zmq_setsockopt(), zmq_getsockopt() 를 통해 수행한다.
- 네트워크 토폴로지에 소켓을 부착한다. zmq_bind()와 zmq_connect()를 통해 수행한다.
- 소켓을 통해 메시지를 송수신한다. zmq_msg_send(), zmq_msg_recv()를 통해 수행한다.

2. 고전의 TCP 소켓 연결과는 다소 다르다.

- 필요에 의하여 임의의 통신 수단을 거친다. inproc, ipc, tcp, pgm, epgm.
- 한 소켓은 다수의 수신(incomming) 그리고 송신(outgoing) 연결을 가진다.
- zmq_accept()와 같은 메서드가 없다. 소켓이 끝단(endpoint)에 바운드되면 자동적으로 연결 대기(accet connections)한다.
- 네트워크 연결은 뒷단에서 일어나며 네트워크 연결이 끊겼을 때 자동적으로 재연결을 한다.
- 응용 프로그램에서는 이러한 연결(connection)에 직접적인 제어를 할 수 없다. 이들은 모두 소켓 내부에 캡슐화되어 있다.

3. 데이터를 다룰 때의 TCP 소켓과 ZeroMQ 소켓 사이의 주요 차이는 아래와 같다.

- ZeroMQ 소켓은 UDP같이 프레임 단위로 메시지를 나른다.
- ZeroMQ 소켓은 I/O 작업을 백그라운드 스레드에서 수행한다. 메시지는 내부 수신큐와 발신큐에 각각 저장되어 오고간다.
- ZeroMQ 소켓은 소켓 타입에 따라 일대다 동작으로 구동된다.
- zmq_send()는 실제로 소켓 연결을 통해 메시지를 보내는 것이 아니며 큐에 넣을 뿐이다.

4. ZeroMQ의 동작 방식을 결정하는 패턴은 다음과 같은 것이 있다. 모호하게 구분되도록 표현이 안 되어 있으나 차이가 분명이 존재한다.

- Request-reply : 다수의 클라이언트와 다수의 서비스들을 연결하는 방식이다. 원격 프로시저 콜이며 작업 분산 패턴이다.
- Pub-sub : 다수의 퍼블리셔와 다수의 서브스크라이버를 연결하는 패턴으로, 데이터 분산 패턴이다.
- Pipeline : 노드들을 여러 단계(step)와 루프(loop)를 가질 수 있는 fan-out/fan-in 패턴으로 연결하는 것으로, 병행 작업 분산-집합(parallel task distribution and collection) 패턴이다.

효자발

임시 저장(multipart message send way etc)

5. 소켓 타입

- PUB : SUB에 데이터를 넘긴다. 수신 불가능.
- SUB : PUB으로부터 데이터를 받는다. 송신 불가능.
- XPUB : PUB을 중계한다.
- XSUB : SUB을 중계한다.
- ROUTER : REQ으로부터 패킷을 받는다. 전달 메시지는 multipart식으로 되며, 특수한 자체 프레임으로 변형된다.
- DEALER : REP에게 패킷을 보낸다. PUSH와 같이 패킷은 분배한다.
- REQ : 동기 방식의 요청용 소켓.
- REP : 동기 방식의 응답용 소켓.
- PUSH : 데이터 분배 송신용 소켓.
- PULL : 데이터 분배 수신용 소켓.
- PIAR : 내부 프로세스간 통신할 때 사용하는 소켓. inproc 통신.

6. Proxy Function 빌트 인(Built-In) 타입

- zmq_proxy(frontend, backend, capture)
- 각 인자는 셋째를 제외하고 보통 대칭이다.

7. 에러 처리는 빨리 실패하고 복구하는 정책.

- 분명한 에러 복구 전략을 명시하지 않으면 메시지는 소리 없이 유실 될 수 있다.
- 알아서 하라네..

8. SYN와 PUB의 통신 확인 가안

- REQ/REP를 이용하여 동기적인 확인(Confirmation)을 진행한다.
- 그러나 매 전송 전마다 수행하되, 해당 소켓 패턴은 동기이므로 데이터가 원활히 송수신되지 않는다면 처리가 되지 않는 문제가 있다.

9. 필요할 경우 'Zero-Copy' 방식을 취할 수 있다.

- 이는 전달할 데이터를 별도의 메시지 큐에 복사하지 않고 직접 애플리케이션 버퍼로부터 데이터를 송수신하는 방법이다.
- 다량의 매우 큰 데이터를 매우 빈번히 나를 경우에만 유용할 듯하다.

10. 프레임을 나눠 전송할 수 있으며 이 경우, 필요한 Key 값의 매칭을 실제 데이터와 분리하는 용도 및 기타 정보를 끼워 넣어 전송하는 용도로 사용할 수 있다.

11. HWM(High Water Mark) : 수위표 개념을 이용해 흐름 제어와 유사한 처리를 진행한다.

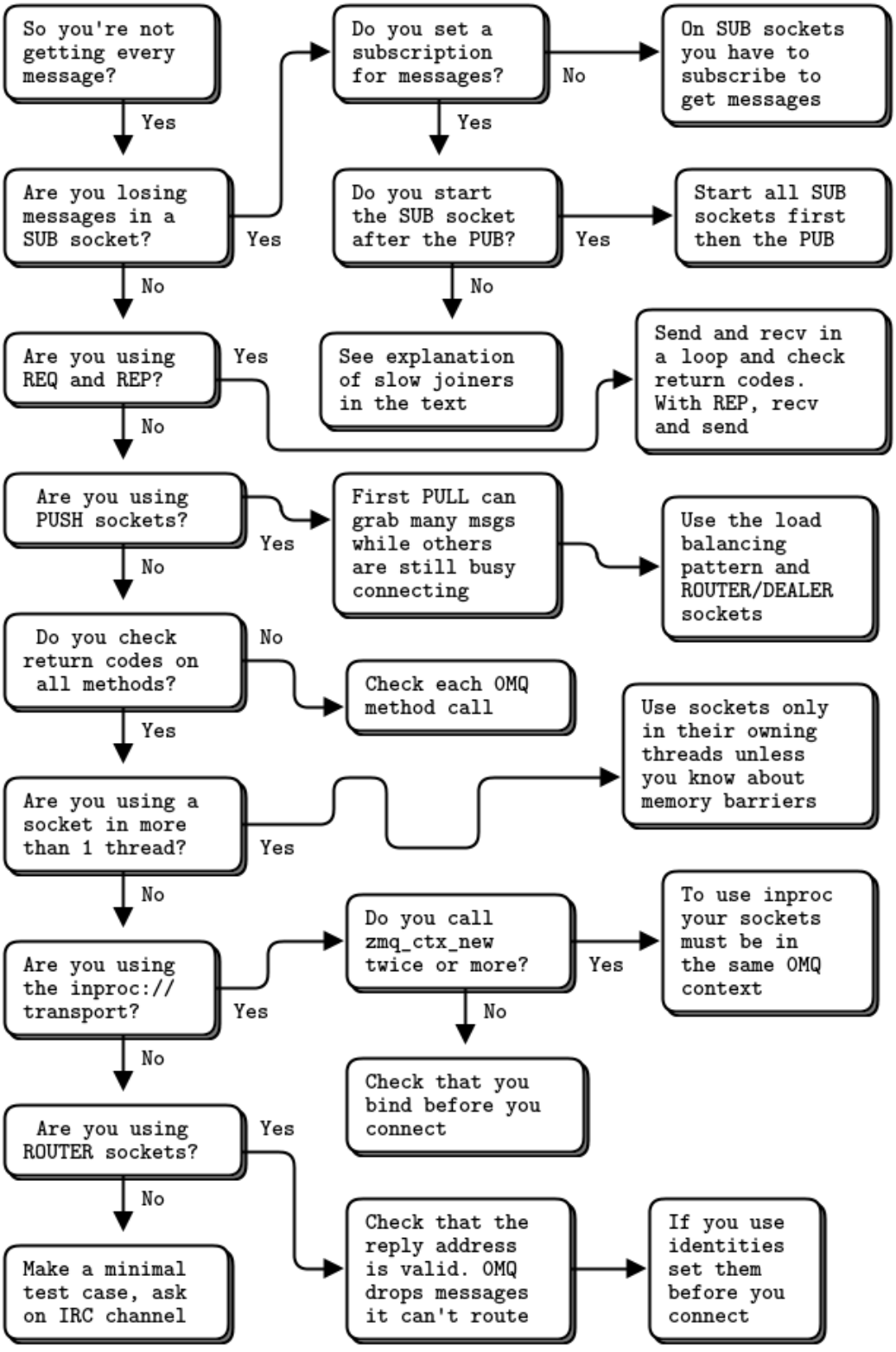
- v2.x 에서는 HWM의 값은 무한이다. 그렇기에 이 값을 설정하는 것이 필요하다. 1,000정도로.
- v3.x 에서 HWM의 값은 기본적으로 1,000으로 정의되어 있다.

효자발

CATEGORY

- 이 외는 모두 차단(block)한다.
- inproc의 경우, 송신 버퍼와 수신 버퍼를 공유하므로 HWM의 실제 값은 두 양단 버퍼의 합과 같다.
- HWM은 정확하지 않을 수 있다. 실제 버퍼 크기가 기본 HWM보다 작을 경우 문제가 된다.
- 확인과 테스트가 필요하다.

12. 메시지 유실 시 아래의 순서도를 기반으로 하여 검토해 본다.



여기까지가 Chaper 02의 내용이다.

댓글 **0**

공지사항

최근에 올라온 글

Kafka 도입부 파악 시도

복합줄 처리 & 출력 플러그인

Logstash로 로그 분석2 (필..

Logstash로 로그 분석

최근에 달린 댓글

Total

3,955

Today 21

Yesterday 5

링크

일취월장개발자

TAG more

« 2019/06 »

일	월	화	수	목	금	토
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

글 보관함

2017/09 (1)

2017/08 (6)

2017/04 (7)

2017/03 (1)

Blog is powered by Tistory / Designed by Tistory