

Building SNMP Trap Receiver Using Apache MINA

1.0 EXECUTIVE SUMMARY

SNMP Trap is a message initiated by a Network Element to notify the Management Station of a significant event. Trap Receiver play a very crucial role in Fault Management Applications, by receiving and parsing the bytes into meaningful messages. In this article we shall discuss the implementation of a SNMP Trap receiver using Apache MINA (Multipurpose Infrastructure for Network Applications). Apache MINA is network application frameworks which help the users develop high performance and highly scalable network applications easily. It provides an abstract - event-driven - asynchronous API over various transports such as TCP/IP and UDP/IP via Java NIO.

The article focuses on developing a Trap Receiver using Apache MINA and available SNMP stacks like SNMP4J, joesnmp etc.

CONTENTS	
<u>SECTION</u>	<u>PAGE</u>
1.0 EXECUTIVE SUMMARY	1
2.0 INTRODUCING APACHE MINA	1
3.0 DESIRABLE IMPLEMENTATION ASPECTS	2
4.0 ARCHITECTURE OF HSC TRAP RECEIVER	3
5.0 SUMMARY	5
6.0 REFERENCES	5

2.0 INTRODUCING APACHE MINA

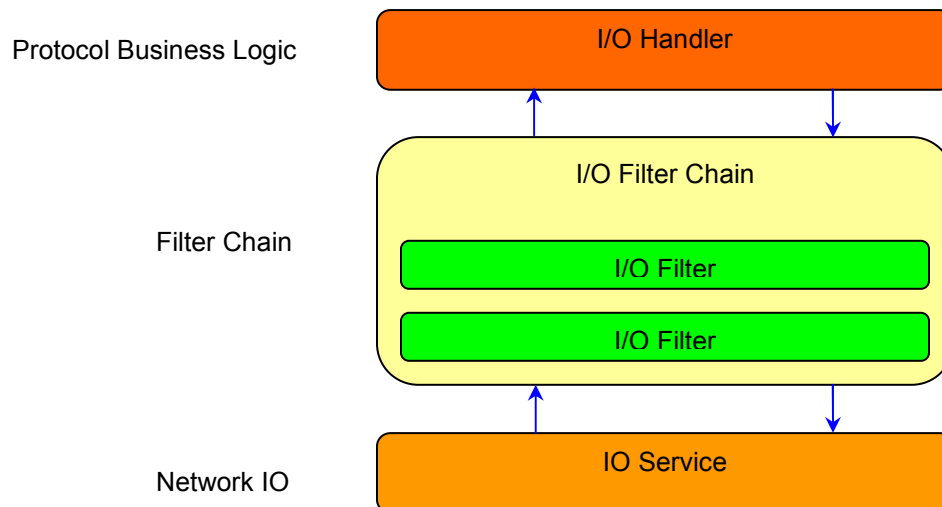


Figure 1: Apache MINA Architecture

[Apache MINA](#) is a simple, yet, full-featured network application framework which provides:

- Unified API for various transport types
- Filter interface as an extension point; similar to Servlet filters

- Low-level and high-level API got IO operations
- Highly customizable thread model
- Out-of-the-box SSL · TLS · StartTLS support using Java 5 SSLEngine
- Overload shielding & traffic throttling
- Unit testability using mock objects
- JMX manageability
- Stream-based I/O support via StreamIoHandler
- Integration with well known containers such as PicoContainer and Spring
- Smooth migration from Netty, an ancestor of Apache MINA.

3.0 DESIRABLE IMPLEMENTATION ASPECTS

A “good” Trap Receiver should be able to handle sustained high Trap rate and have virtually no dependencies towards the SNMP stack used for decoding. Such a Trap Receiver should have:

- Flexibility to use any available SNMP stack for Trap Decoding with minimal changes
- Shall use SEDA (Staged Event Driven Architecture) based Design to achieve high throughput.
- Able to handle a sustained a high Trap Rate (2000 Traps/Sec or greater)

3.1 Functional View of Trap Receiver Application

The figure below describes a high level Functional View of a Trap Receiver

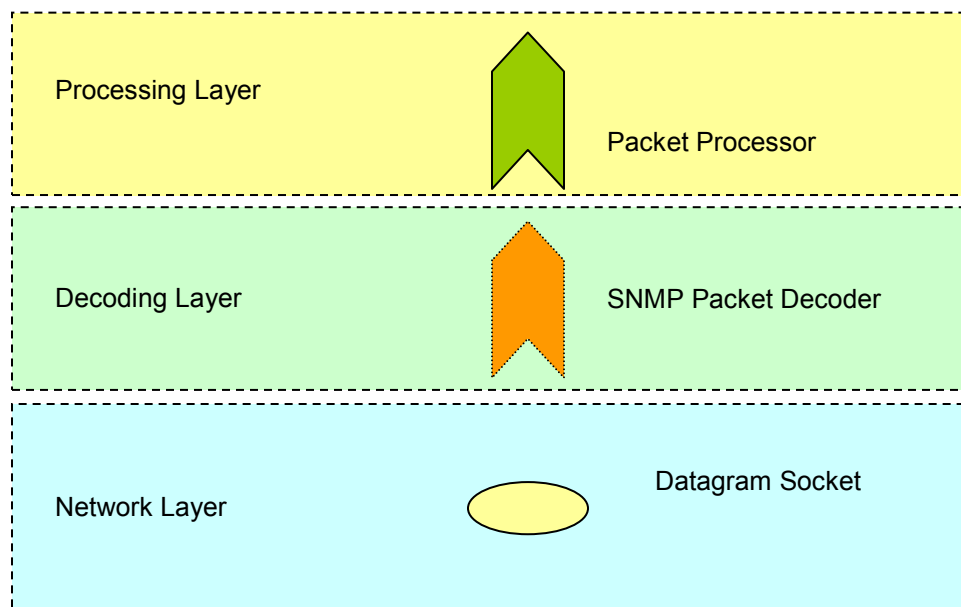


Figure 2: Functional View of Trap Receiver

The conventional Trap Receiver implementation comprises of three layers with distinct functional role at each level:

- The Network layer is responsible for receiving the Trap packet from the network and passing onto next layer
- The Decoding Layer is responsible for converting the bytes into SNMP Trap representation
- The Processing Layer is responsible for processing the SNMP Trap, like persisting in database etc

In the conventional architecture, these three layers functions are performed using a single thread and many a times, a [thread pool](#) is used for higher performance.

Design aspects unique to HSC Trap Receiver

- Upon reception of raw packet, the packet shall be inserted in a receive Queue, without processing, in order to achieve high packet reception rate
- Decoding shall be done by separate threads, by picking the packets from the receive queue
- SNMP Packet decoder shall pass the processed packets to Packet processor which shall dump them into log files

4.0 ARCHITECTURE OF HSC TRAP RECEIVER

This section describes the Architecture of HSC Trap Receiver. The architecture has been redefined to achieve high processing rate, along with minimum dependencies on any SNMP stacks. The steps briefly describe what all shall be needed:

1. Receive UDP packets on the Socket and push them into input Queue without processing, to achieve a very high
2. Thread pull the raw SNMP packet from the Queue in Step 1 and uses SNMP library to convert the packets into SNMP packets
3. The SNMP packets are placed into Queue of subsequent stages. The stages are implemented as MINA I/O filters. An example of a Stage is OSSJ Converter in Figure 3: Architecture of HSC Trap Receiver
4. The packets from Queue in Step 3 are picked up by Thread and Business processing is performed over the SNMP packets

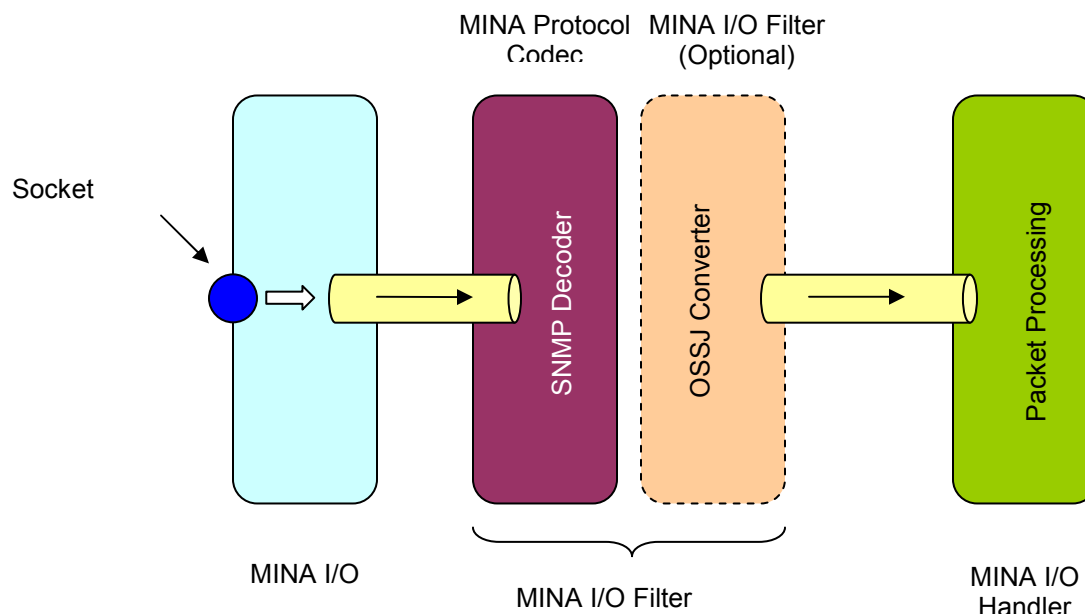


Figure 3: Architecture of HSC Trap Receiver

Apache MINA framework takes care of all IO related and housekeeping functions like creating input Queue and passing the message through the processing chain.

With the IO infrastructure in place, we can concentrate on our prime Business Logic, of decoding the byte array into an SNMP messages. An implementation of MINA's protocol Decoder shall convert bytes into SNMP object representation using SNMP4J. MINA ProtocolDecoder's being the added into the protocol chain can easily be replaced with other implementation, without changing the remaining code.

Sample SNMP4J based MINA codec is listed below

```

18 public class SNMP4JCodec extends ProtocolDecoderAdapter {
19
20     static Logger logger = LoggerFactory.getLogger(SNMP4JCodec.class);
21
22     public void decode(ioSession ioSession, IoBuffer ioBuffer,
23         ProtocolDecoderOutput protocolDecoderOutput) throws Exception {
24         ByteBuffer pduBuffer = ioBuffer.buf();
25         // Decode the bytes using SNMP4J API's
26         PDU pdu = new PDU();
27         try {
28             BERInputStream berStream = new BERInputStream(pduBuffer);
29             BER.MutableByte mutableByte = new BER.MutableByte();
30             int length = BER.decodeHeader(berStream, mutableByte);
31             int startPos = (int)berStream.getPosition();
32
33             if (mutableByte.getValue() != BER.SEQUENCE) {
34                 String txt = "SNMPv2c PDU must start with a SEQUENCE";
35                 throw new IOException(txt);
36             }
37             Integer32 version = new Integer32();
38             version.decodeBER(berStream);
39
40             // decode community string
41             OctetString securityName = new OctetString();
42             securityName.decodeBER(berStream);
43
44             // decode the remaining PDU
45             pdu.decodeBER(berStream);
46             logger.debug("PDU - "+pdu);
47         } catch (Exception ex) {
48             ex.printStackTrace();
49         }
50         protocolDecoderOutput.write(pdu);
51     }

```

With Protocol codec in place, its time to put the glue code in place

```

19     // Create Datagram Socket to listen for incoming traps
20     MIODatagramAcceptor acceptor = new MIODatagramAcceptor(Executors.newCachedThreadPool());
21     acceptor.setDefaultLocalAddress(new InetSocketAddress(162));
22
23     // Add IOHandler implementing SNMP packet processing
24     acceptor.setHandler(new DefaultIOHandler());
25
26     // Create Socket configuration
27     DatagramSessionConfig ssf = acceptor.getSessionConfig();
28     ssf.setReuseAddress(true);
29
30     DefaultIOFilterChainBuilder filterChain = acceptor.getFilterChain();
31     // Add Thread pool to implement SEDR based processing
32     filterChain.addLast("Executor", new ExecutorFilter(Executors.newCachedThreadPool()));
33     // Add SNMP4J Codec
34     filterChain.addLast("snmpCodec", new ProtocolCodecFilter(new SNMPCodecFactory()));
35     acceptor.bind();

```

SNMP Codec factory return a new instance of SNMP4JCodec. With SNMP4J codec added into Filter chain MINA uses the codec to decode the byte array into SNMP object of SNMP4J and pass onto next filter, and subsequently to IOHandler.

The SNMP4J codec can be easily replaced with joesnmp codec or Adventnet codec, by implementing a Protocol decoder using the library and returning the instance from ProtocolCodec factory class.

To use SEDA, Executors can be added in between filter's to achieve threadpool based processing, within each filter/

4.1 Advantages of the HSC Trap Receiver Architecture

- Ease of replacing the decoding function with other implementation, without changing core logic
- Easy Unit testing using MockObjects
- Adding features is easy adding filters to the Filter chain.

4.2 HSC Trap Receiver vs. Fault Management Frameworks

- Most of Fault Management frameworks come bundled with custom SNMP stacks, which limits the Users capability to use any custom stack for their applications
- Customization of FM tools is strictly governed by the extensibility of the tools. In our implementation, customization such as black listing IP's etc can be achieved by adding I/O filter in the MINA IO Filter chain.
- MINA, can be embedded in Spring and PicoContainer, thus we can take full advantages of Clustering, Dependency Injection

5.0 SUMMARY

Apache MINA eases creation of high performance Network applications. The Design of HSC Trap Receiver, based on Apache MINA, makes it easy to use any SNMP stack in the Fault Management application. Additionally, the design based on IOFilter chain, gives flexibility in adding additional functionality like blacklisting IP, managing throughput without modifying the Decoding logic or Trap processing. Under the Test environment (Intel Core2 Duo 2.2 GHz, Windows XP, 1 GB RAM), the implementation achieved a sustained rate of 3000 Traps/sec without dropping any packets.

6.0 REFERENCES

- <http://mina.apache.org/>
- <http://www.eecs.harvard.edu/~mdw/proj/seda/>
- <http://www.snmp4j.org/>
- <http://sourceforge.net/projects/joesnmp/>
- <http://www.springframework.org/>
- <http://www.picocontainer.org/>

PROPRIETARY NOTICE

All rights reserved. This publication and its contents are proprietary to Hughes Systique Corporation. No part of this publication may be reproduced in any form or by any means without the written permission of Hughes Systique Corporation, 15245 Shady Grove Road, Suite 330, Rockville, MD 20850.

Copyright © 2006 Hughes Systique Corporation

CONTACT INFORMATION:

Phone: +1.301.527.1629

Fax: +1.301.527.1690

email: whitepaper@hsc.com

Web: www.hsc.com

APPENDIX A ABOUT HUGHES SYSTIQUE CORPORATION

HUGHES Systique Corporation (HSC), part of the HUGHES group of companies, is a leading Consulting and Software company focused on Communications and Automotive Telematics. HSC is headquartered in Rockville, Maryland USA with its development centre in Gurgaon, India.

SERVICES OFFERED:

Technology Consulting & Architecture: Leverage extensive knowledge and experience of our domain experts to define product requirements, validate technology plans, and provide network level consulting services and deployment of several successful products from conceptualization to market delivery.

Development & Maintenance Services: We can help you design, develop and maintain software for diverse areas in the communication industry. We have a well-defined software development process, comprising of complete SDLC from requirement analysis to the deployment and post production support.

Testing : We have extensive experience in testing methodologies and processes and offer Performance testing (with bench marking metrics), Protocol testing, Conformance testing, Stress testing, White-box and black-box testing, Regression testing and Interoperability testing to our clients

System Integration : As system integrators of choice HSC works with global names to architect, integrate, deploy and manage their suite of OSS, BSS, VAS and IN in wireless (VoIP & IMS), wireline and hybrid networks.: NMS, Service Management & Provisioning .

DOMAIN EXPERTISE:

Terminals

- Terminal Platforms : iPhone, Android, Symbian, Windows CE/Mobile, BREW, PalmOS
- Middleware Experience & Applications : J2ME , IMS Client & OMA PoC,

Access

- Wired Access : PON & DSL, IP-DSLAM,
- Wireless Access : WLAN/WiMAX / LTE, UMTS, 2.5G, 2G ,Satellite Communication

Core Network

- IMS/3GPP , IPTV , SBC, Interworking , Switching solutions, VoIP

Applications

- Technologies : C, Java/J2ME, C++, Flash/lite, SIP, Presence, Location, AJAX/Mash
- Middleware: GlassFish, BEA, JBOSS, WebSphere, Tomcat, Apache etc.

Management & Back Office:

- Billing & OSS , Knowledge of COTS products , Mediation, CRM
- Network Management : NM Protocols, Java technologies,, Knowledge of COTS NM products, FCAPS, Security & Authentication

Platforms

- Embedded: Design, Development and Porting - RTOS, Device Drivers, Communications / Switching devices, Infrastructure components. Usage and Understanding of Debugging tools.
- FPGA & DSP : Design, System Prototyping. Re-engineering, System Verification, Testing

Automotive Telematics

- In Car unit (ECU) software design with CAN B & CAN C
- Telematics Network Design (CDMA, GSM, GPRS/UMTS)

BENEFITS:

- **Reduced Time to market :** Complement your existing skills, Experience in development-to-deployment in complex communication systems, with key resources available at all times
- **Stretch your R&D dollars :** Best Shore” strategy to outsourcing, World class processes, Insulate from resource fluctuations