Performance Analysis of SNMP over SSH

Vladislav Marinov and Jürgen Schönwälder

International University Bremen
Bremen, Germany

IFIP/IEEE DSOM 2006

Outline of the Talk

Background: IETF ISMS Working Group

- SNMPv3/USM was designed to have its own user and key management infrastructure
- Operators reported that deploying another user and key management infrastructure introduces significant costs and is a reason for not deploying SNMPv3
- The ISMS working group of the IETF was chartered in September 2004 to address this issue
- After discussing various proposals, the working group was re-chartered in September 2005 to focus on SNMP over SSH and RADIUS integration

Motivation and Research Goals

- SNMPv3/USMS uses message-based security:
  - every message contains all needed security information
  - keys are not bound to a lifetime of a session
- SNMPv3/SSH uses session-based security:
  - session establishment phase (session key negotiation)
  - no per message security information needed
  - SSH requires usage of the TCP
- Research goals:
  - understand the performance tradeoffs
  - provide "running code" experience to the working group
  - explore possible alternatives (e.g., TLS or DTLS)

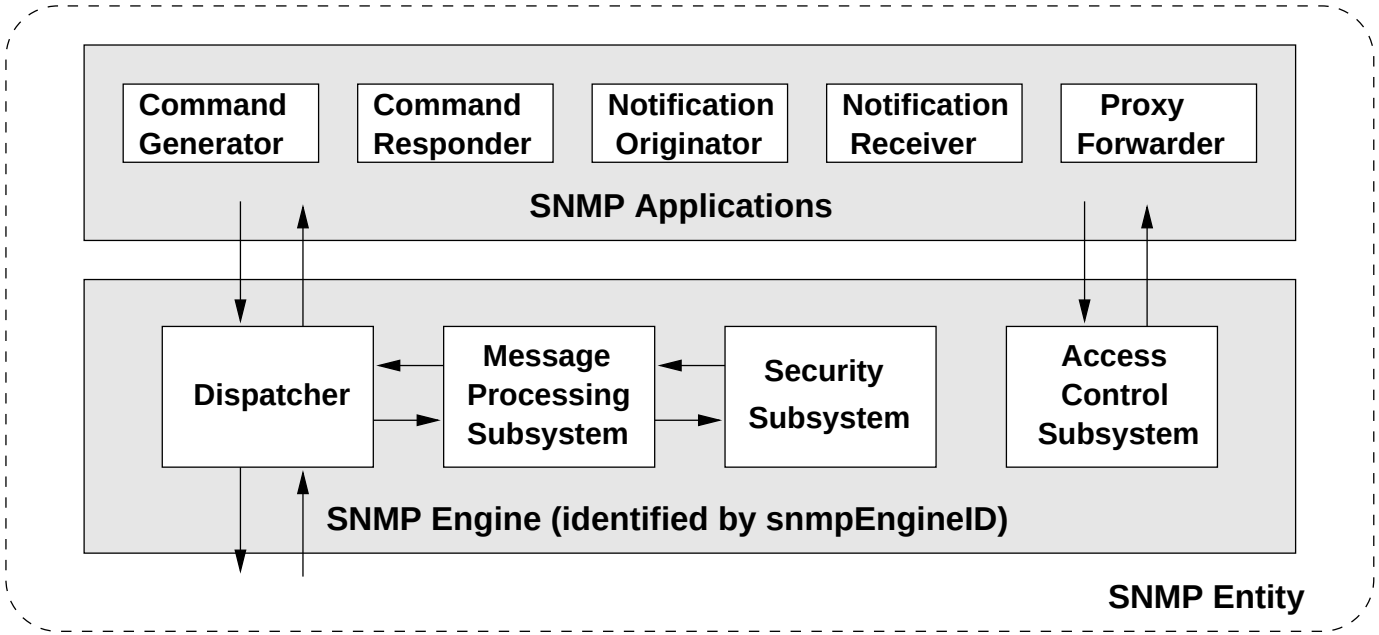
Motivation and Research Goals

- SNMPv3/USMS uses message-based security:
  - every message contains all needed security information
  - keys are not bound to a lifetime of a session
- SNMPv3/SSH uses session-based security:
  - session establishment phase (session key negotiation)
  - no per message security information needed
  - SSH requires usage of the TCP
- Research goals:
  - understand the performance tradeoffs
  - provide "running code" experience to the working group
  - explore possible alternatives (e.g., TLS or DTLS)

Motivation and Research Goals

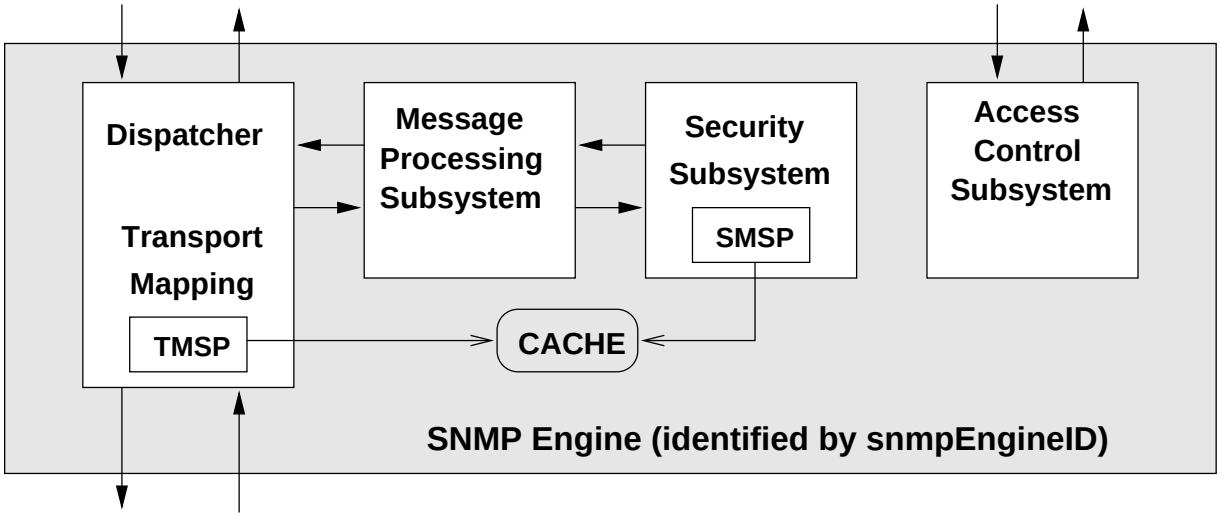
- SNMPv3/USMS uses message-based security:
  - every message contains all needed security information
  - keys are not bound to a lifetime of a session
- SNMPv3/SSH uses session-based security:
  - session establishment phase (session key negotiation)
  - no per message security information needed
  - SSH requires usage of the TCP
- Research goals:
  - understand the performance tradeoffs
  - provide "running code" experience to the working group
  - explore possible alternatives (e.g., TLS or DTLS)

SNMP Architecture [1]



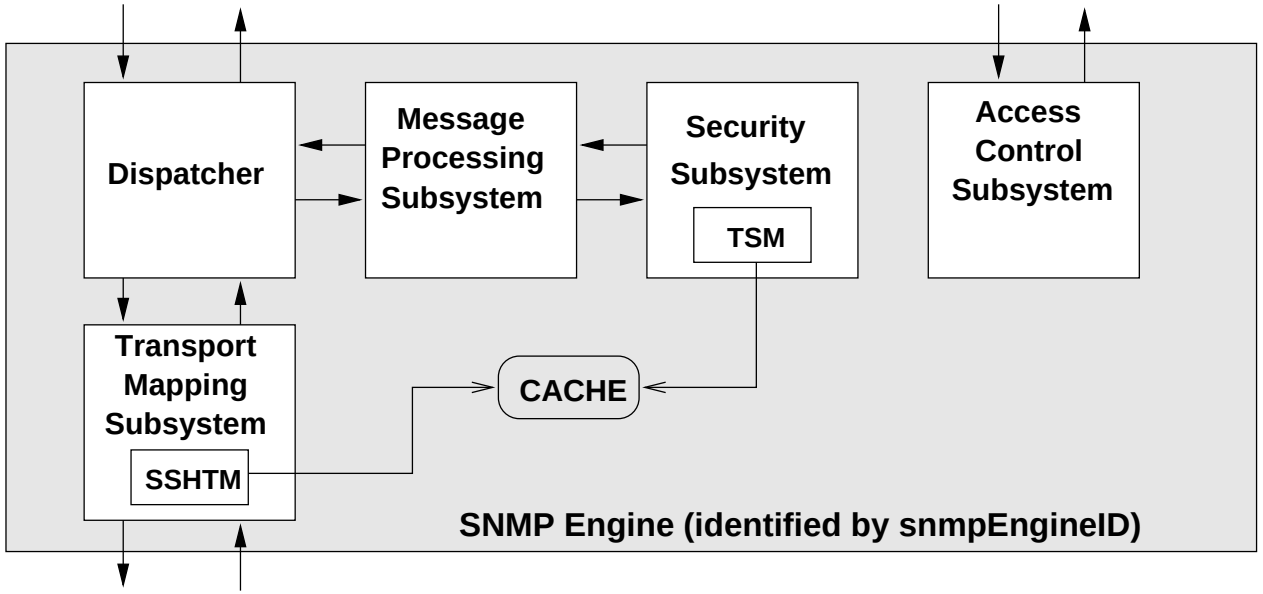
- Suitable for message-based transports (USM) - all security information is carried in every message
- Support for session-based transports (SSH, TLS, etc.) requires the addition of a transport mapping security model

Extension of the SNMP Architecture (paper)



- Transport Mapping Security Processor (TMSP) performs the actual security processing
- Security Model Security Processing (SMSP) realizes the security model required by SNMPv3
- Our focus is on the secure transport mapping

Extension of the SNMP Architecture (today)



- A separate Transport Mapping Subsystem was defined which can implement different transport models (UDPTM, TCPTM, SSHTM, TLSTM, ...) [2]
- A generic Transport Security Model (TSM) [3] hooks into a secure transport such as SSHTM [4]

Prototype Implementation

- NET-SNMP - open source C implementation of SNMP
- LibSSH - open source C library for SSH
- openssl - open source C library for TLS
- A new transport domain (SSHDomain) was defined and the transport independent functions were implemented
- An interface to the Pluggable Authentication Modules for Linux (PAM) [5] enables a system administrator to choose how credentials of a user are verified
- Implementation passes the authenticated SSH user identitiy as a security model neutral security name to the Access Control Subsystem
- Prototype consists of 1200 lines of C code (for the SSH portion)

Optimizing the Prototype

- Initial results were extremely frustrating.
- Careful analysis revealed two optimizations:
- TCP's Nagle algorithm was disabled immediately after the TCP connection establishment - improved the performance of a single `snmpget` operation from 800*ms* to 16*ms*
- SSH Window Adjustments - the `libssh` library was tuned to send window adjustment messages only when necessary - decreased the overhead and the latency for long sessions

Optimizing the Prototype

- Initial results were extremely frustrating.
- Careful analysis revealed two optimizations:
- TCP's Nagle algorithm was disabled immediately after the TCP connection establishment - improved the performance of a single `snmpget` operation from 800*ms* to 16*ms*
- SSH Window Adjustments - the `libssh` library was tuned to send window adjustment messages only when necessary - decreased the overhead and the latency for long sessions

Optimizing the Prototype

- Initial results were extremely frustrating.
- Careful analysis revealed two optimizations:
- TCP's Nagle algorithm was disabled immediately after the TCP connection establishment - improved the performance of a single `snmpget` operation from 800*ms* to 16*ms*
- SSH Window Adjustments - the `libssh` library was tuned to send window adjustment messages only when necessary - decreased the overhead and the latency for long sessions

Experimental Setup

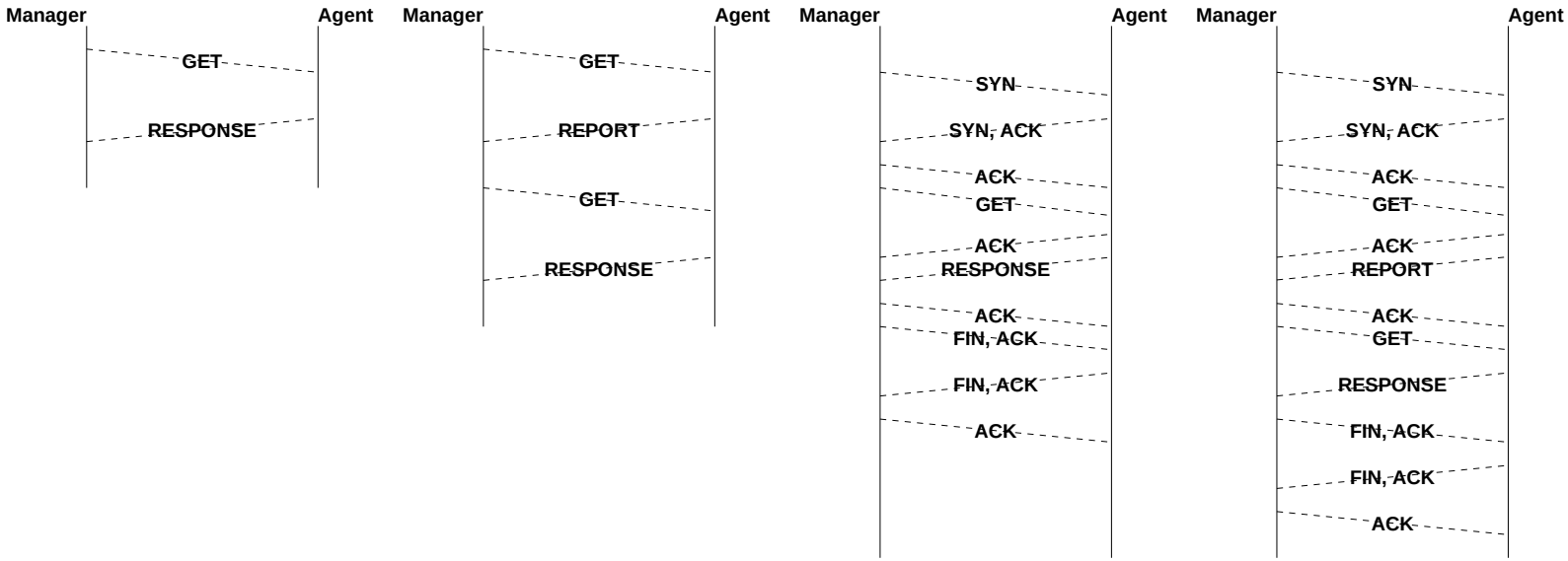
- `meat`: 2 x Intel Xeon 3GHz CPU, 2GB RAM, connected to the switched lab network via 1Gbps interface, running Debian Linux 2.6.15.1
- `veggie`: 2 x Intel Xeon 3GHz CPU, 1GB RAM, connected to the switched lab network via 1Gbps interface, running Debian Linux 2.6.12.6 XEN
- `turtle`: Ultra Sparc IIi, 128 MB RAM, connected to the switched lab network via 10Mbps interface, running Debian Linux 2.6.13
- `gettimeofday()` system call, `pmap` utility, `tc` utility, `tcpdump` utility

SNMP/SSH Session Establishment Overhead

Protocol	Time (`meat`)	Time (`turtle`)	Data	Packets
v2c/UDP	1.03 ms	0.70 ms	232 bytes	2
v2c/TCP	1.13 ms	1.00 ms	824 bytes	10
v3/USM/UDP	1.97 ms	2.28 ms	668 bytes	4
v3/USM/TCP	2.03 ms	3.03 ms	1312 bytes	12
v2c/SSH	**16.17 ms**	**91.62 ms**	**4388 bytes**	**32**
v2c/TLS	**18.00 ms**		**4109 bytes**	**16**

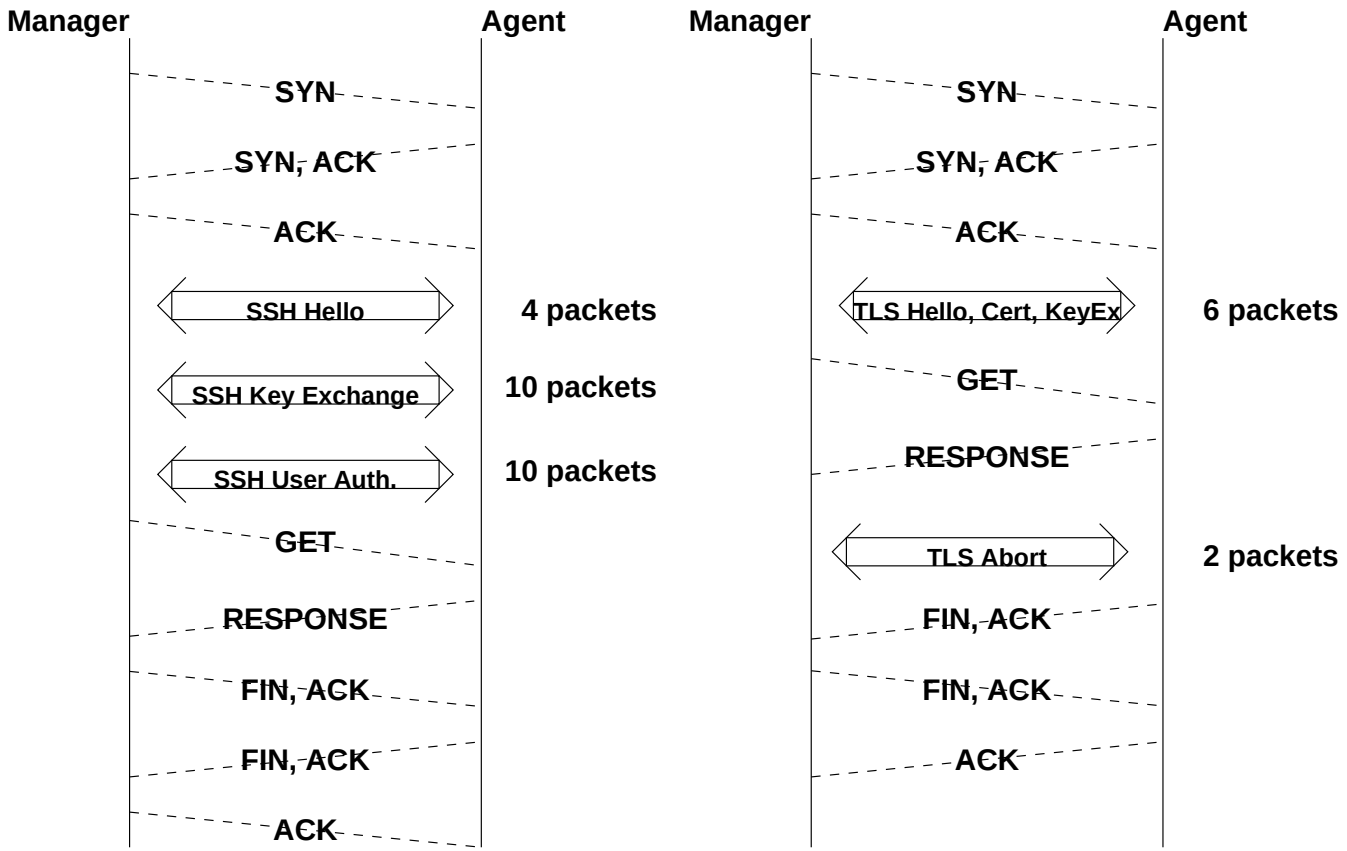
- Overhead of SSH session establishment was measured using response time of `snmpget` operation
- SNMPv2c/SSH introduces significant overhead for session establishment
- SNMPv2c/TLS uses less packets but exchanges similar amount of data
- However, overhead can be amortized over long sessions...

Time Sequence Diagrams (v2c / v3/USM)



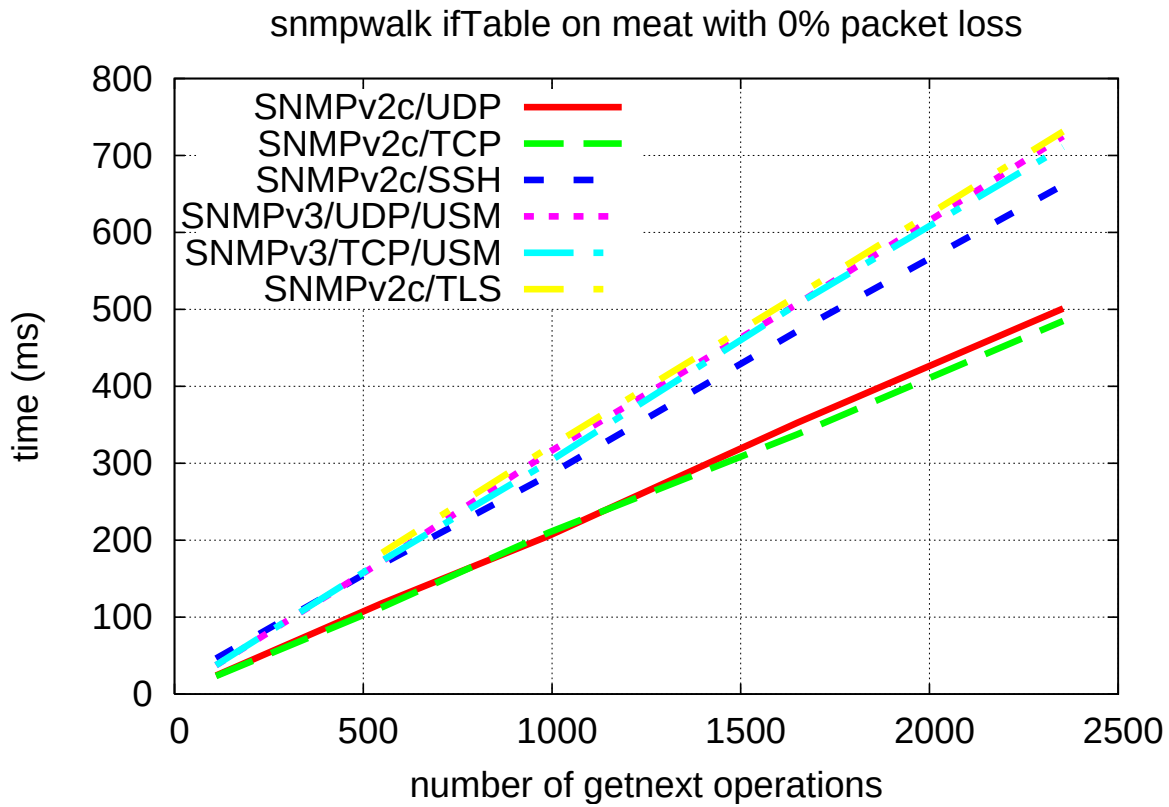
- USM requires one SNMP round trip for engineID discovery and clock synchronization (`REPORT` PDU)
- TCP adds connection establishment and teardown overhead

Time Sequence Diagrams (v2c SSH / TLS)



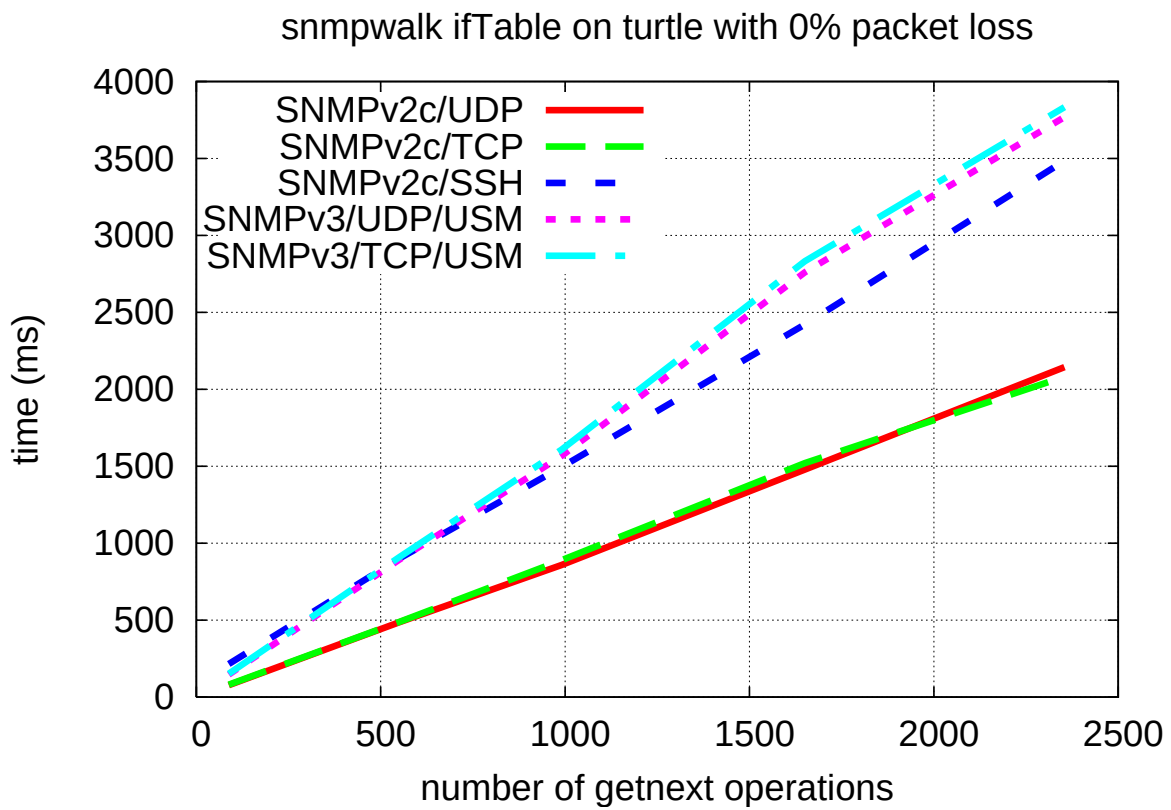
- SSH requires more messages than TLS
- TLS initiates TCP teardown from the agent

Latency Without Packet Loss (fast machine)



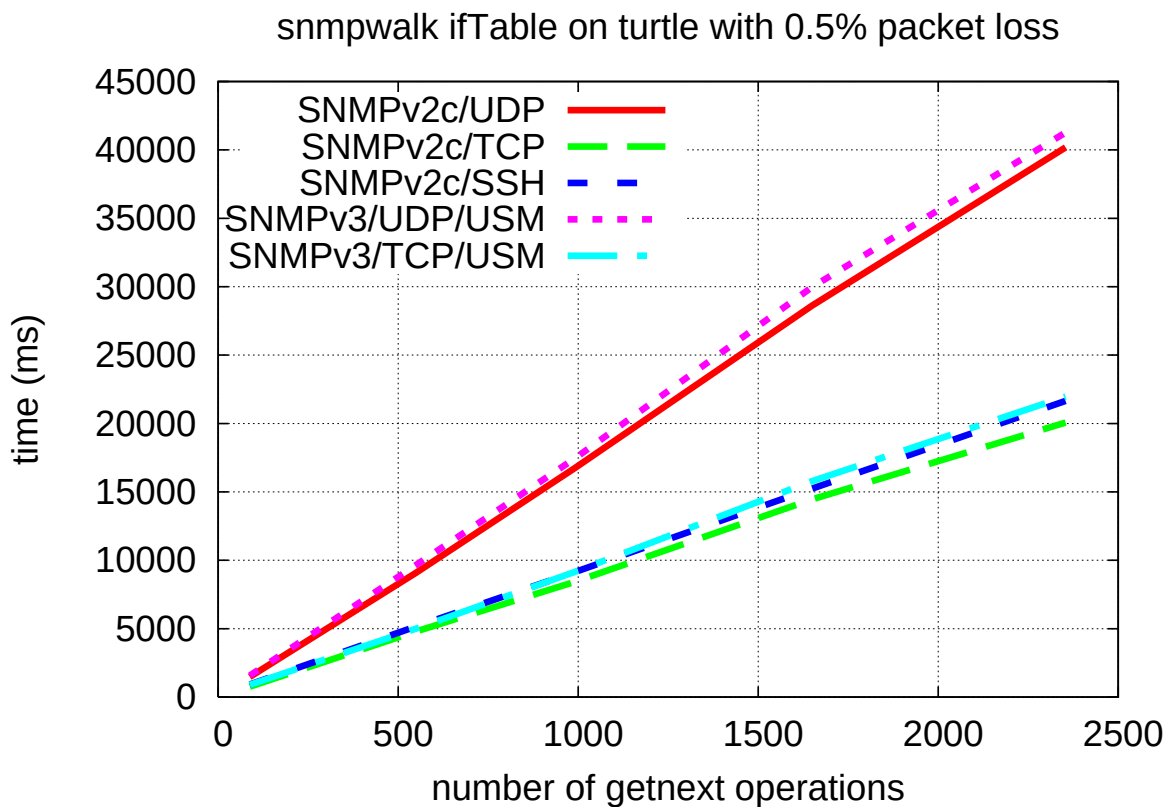
- Overhead of `snmpwalk` and `snmpbulkwalk` operations
- Marginal difference between TCP and UDP
- Initially SSH performs worse than USM due to session establishment overhead

Latency Without Packet Loss (slow machine)



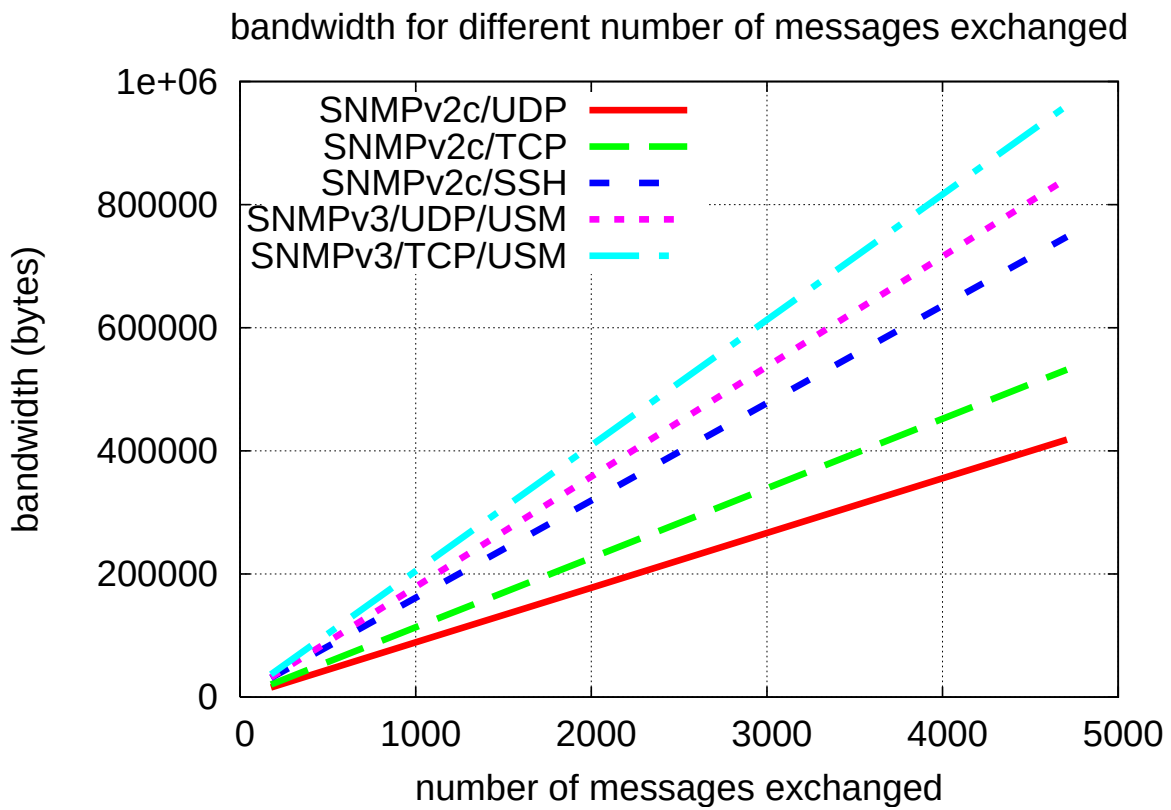
- The slope of the curves does not change on a slower machine
- The dimension of the y-axis changes significantly

Latency With Packet Loss (slow machine)



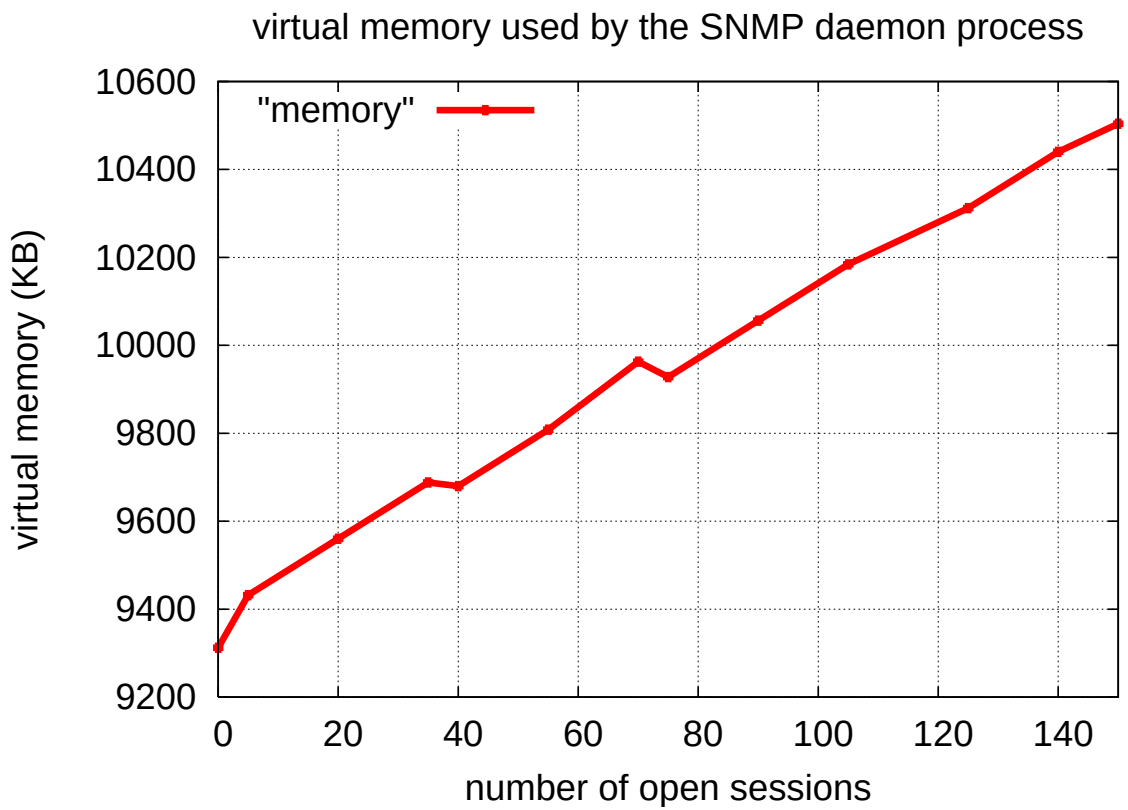
- SNMP/SSH and SNMP/TCP perform better than SNMP/UDP
- Poor retransmission algorithm of SNMP/UDP in the NET-SNMP implementation

Bandwidth Consumption



- SNMPv2/SSH requires less bandwidth compared to SNMPv3/USM because it does not carry security information in every message

Memory Usage



- Memory overhead for establishment of a session was measured using the `pmap` utility
- Virtual memory allocated to the SNMP daemon grows linearly with the increase of the number of open sessions

Conclusions

- SNMP/SSH is a new secure transport for SNMP that
  - uses session based security
  - reuses existing user/key infrastructure
  - offers better security (session keys, perfect forward secrecy)
- Performance analysis reveals
  - high overhead for short sessions
  - minimal overhead for long sessions
  - requires less bandwidth than SNMPv3/USM
  - outperforms SNMP/UDP under packet loss (NET-SNMP implementation)
- Can be used to transition from SNMPv1 and SNMPv2c to SNMPv3 more easily

Conclusions

- SNMP/SSH is a new secure transport for SNMP that
  - uses session based security
  - reuses existing user/key infrastructure
  - offers better security (session keys, perfect forward secrecy)
- Performance analysis reveals
  - high overhead for short sessions
  - minimal overhead for long sessions
  - requires less bandwidth than SNMPv3/USM
  - outperforms SNMP/UDP under packet loss (NET-SNMP implementation)
- Can be used to transition from SNMPv1 and SNMPv2c to SNMPv3 more easily

Conclusions

- SNMP/SSH is a new secure transport for SNMP that
  - uses session based security
  - reuses existing user/key infrastructure
  - offers better security (session keys, perfect forward secrecy)
- Performance analysis reveals
  - high overhead for short sessions
  - minimal overhead for long sessions
  - requires less bandwidth than SNMPv3/USM
  - outperforms SNMP/UDP under packet loss (NET-SNMP implementation)
- Can be used to transition from SNMPv1 and SNMPv2c to SNMPv3 more easily

References

D. Harrington, R. Presuhn, and B. Wijnen.

An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.
RFC 3411, Enterasys Networks, BMC Software, Lucent Technologies, December 2002.

D. Harrington and J. Schönwälder.

Transport Subsystem for the Simple Network Management Protocol (SNMP).
Internet Draft (work in progress) <draft-ietf-isms-tmsm-04.txt>, Huawei Technologies, International University Bremen, October 2006.

D. Harrington.

Transport Security Model for SNMP.
Internet Draft <draft-ietf-isms-transport-security-model-00.txt>, Huawei Technologies, October 2006.

D. Harrington and J. Salowey.

Secure Shell Transport Model for SNMP.
Internet Draft (work in progress) <draft-ietf-isms-secshell-05.txt>, Futurewei Technologies, Cisco Systems, October 2006.

A. G. Morgan.

The Linux-PAM Application Developers' Guide.
Technical report, November 1999.