

# SNMP Trace Analysis Update

Jürgen Schönwälder

Jacobs University Bremen  
Bremen, Germany

22. NMRG at 68. IETF, Prague, March 2007

# Outline of the Slide Show

- 1 Trace Characterization
- 2 Flow Analysis
- 3 MIB Object Analysis
- 4 Walk Analysis
- 5 References

# Trace Collection

<b>trace</b>	<b>description</b>	<b>start</b>	<b>hours</b>
<i>l01t02</i>	national research network	2005-07-26	162.98
<i>l01t05</i>	national research network	2006-07-10	336.00
<i>l02t01</i>	university network	2006-04-21	294.62
<i>l03t02</i>	faculty network	2006-04-27	159.21
<i>l04t01</i>	server-hosting provider	2006-04-14	4.00
<i>l05t01</i>	regional network provider	2006-04-19	580.60
<i>l06t01</i>	national research network	2006-05-14	222.08
<i>l12t01</i>	point of presence	2006-07-10	208.02

- Some more traces are being collected within EMANICS
- Additional traces are most welcome (especially from non university / research networks)

# Trace Overview and Characterization

trace	size [MB]	messages	SNMPv1	SNMPv2	SNMPv3
<i>/01t02</i>	6369	51772136	100.0%	-	-
<i>/01t05</i>	14043	40072529	-	100.0%	0.0%
<i>/02t01</i>	77789	258010521	5.5%	94.5%	-
<i>/03t02</i>	130858	871361365	95.0%	5.0%	-
<i>/04t01</i>	10	15099	35.7%	64.3%	-
<i>/05t01</i>	2898	25298667	100.0%	-	-
<i>/06t01</i>	24683	89277889	57.4%	42.6%	-
<i>/12t01</i>	312	2619884	32.3%	67.7%	-

- One trace included a few very sporadic SNMPv3 packets (someone testing SNMPv3?)
- Location */01* uses in one location 100% SNMPv1 and in a second location 100% SNMPv2c

# Protocol Operations

<b>trace</b>	Get	Next	Bulk	Set	Trap	Inform	Resp
<i>101t02</i>	0.0	50.0	-	0.0	-	-	50.0
<i>101t05</i>	0.0	-	50.0	-	-	-	50.0
<i>102t01</i>	0.1	2.4	47.1	0.0	0.7	-	49.6
<i>103t02</i>	0.3	49.8	-	0.0	0.0	-	49.9
<i>104t01</i>	32.8	3.8	22.9	-	-	-	40.5
<i>105t01</i>	50.0	0.0	-	-	0.0	-	50.0
<i>106t01</i>	12.1	31.4	6.5	-	0.0	0.0	50.0
<i>112t01</i>	1.0	49.0	-	-	0.0	-	49.9

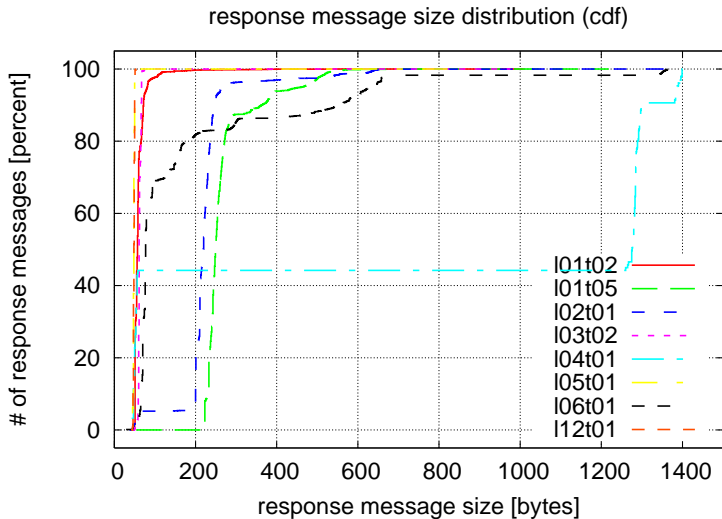
- In *101t02*, all Set operations were trying to modify sysLocation with a value of type Integer32
- In *102t01* and *103t02*, Set operations were used to trigger download of config information
- In *104t01*, there were significantly more requests than responses (system maintenance)

# PDU Parameters

trace	Get	Next	Bulk	max-reps	non-reps
<i>l01t02</i>	37.5%	99.3%	-	-	-
<i>l01t05</i>	100.0%	-	100.0%	10/50	0
<i>l02t01</i>	56.3%	99.9%	100.0%	1/10/20/25	0
<i>l03t02</i>	1.6%	99.9%	-	-	-
<i>l04t01</i>	100.0%	100.0%	100.0%	1000	0
<i>l05t01</i>	99.9%	95.6%	-	-	-
<i>l06t01</i>	8.7%	2.6%	0.0%	12	0
<i>l12t01</i>	100.0%	99.9%	-	-	-

- Except for *l06t01*, single varbind GetNext and GetBulk operations dominate
- In *l06t01*, 86.5% of the GetBulk operations contain two varbinds and the remaining 13.5% contain eight varbinds

# Response Size Distribution



## Definition

An SNMP message flow is defined as all messages between a source and destination address pair which belong to a command generator (CG) / command responder (CR) relationship or a notification originator (NO) / notification receiver (NR) relationship.

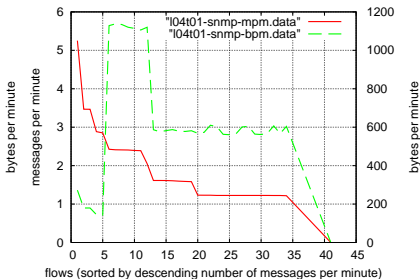
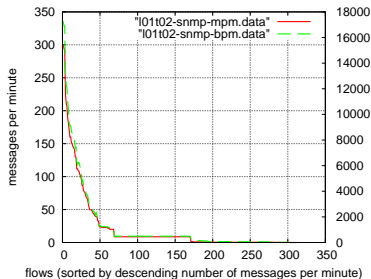
- The above definition deliberately does not consider port numbers.
- Multi-homed managers will appear with multiple flows.

# Flow Statistics

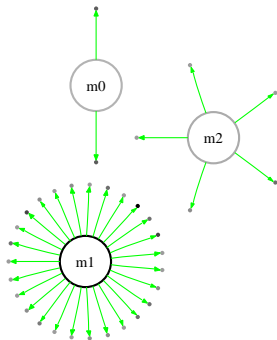
<b>trace</b>	<b>cg/cr flows</b>	<b>no/nr flows</b>	<b>cg</b>	<b>cr</b>	<b>no</b>	<b>nr</b>
<i>l01t02</i>	203	-	3	178	-	-
<i>l01t05</i>	8	-	2	8	-	-
<i>l02t01</i>	258	197	5	240	197	1
<i>l03t02</i>	42	20	25	20	17	2
<i>l04t01</i>	34	-	3	34	-	-
<i>l05t01</i>	117	2	9	99	2	2
<i>l06t01</i>	288	125	3	260	125	2
<i>l12t01</i>	30	6	5	26	6	1

- Traffic is not evenly distributed across the flows
- Most traces have very few dominating flows

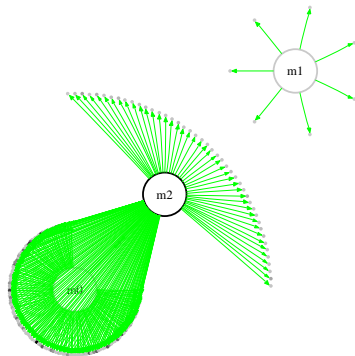
# Flow Size Distribution



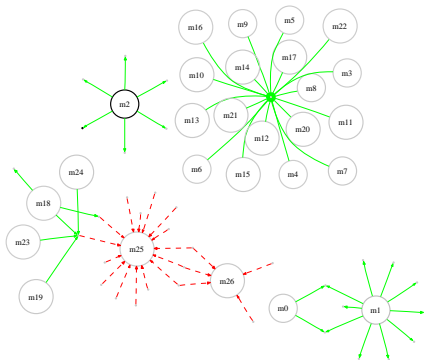
- In some traces, the number of message and bytes per flow is closely correlated
- In other traces, this is not the case (essentially due to GetBulk usage)



- Typical simple monitoring topology
- Some flows carry more traffic than other flows
- Gray-level indicates intensity (but difficult to see)

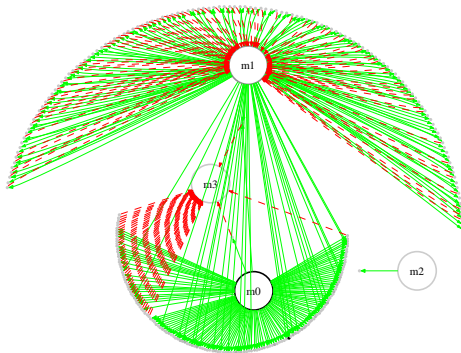


- Slightly more complex monitoring topology
- Some devices are interacting with multiple management interfaces



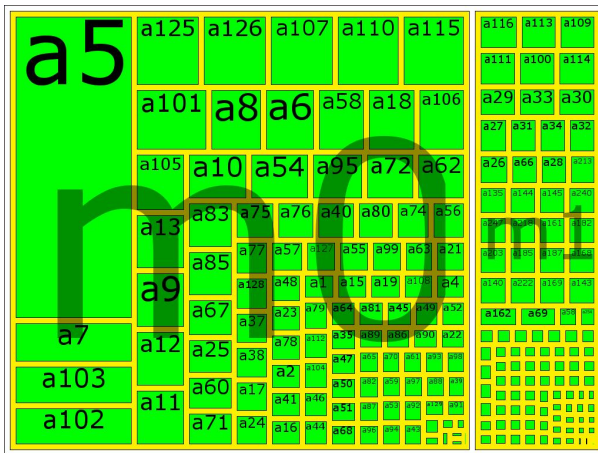
- Large number of managers talking to a single agent
- Analysis revealed that the device is a printer queried from a PC which is getting dynamically changing IP addresses

# Flow Topology *l06t01*



- Significant number of NO/NR flows
- Single pretty dark dot indicates that a single flow is dominating

# Flow Topology *106t01* (treemap)



- Treemap plots nicely visualizes contribution of the flows

<b>trace</b>	<b>int32</b>	<b>uint32</b>	<b>uint64</b>	<b>oct</b>	<b>oid</b>	<b>ip</b>	<b>null</b>	<b>exc</b>
<i>l01t02</i>	48.1	3.2	-	39.6	0.3	8.6	0.2	-
<i>l01t05</i>	13.4	21.0	52.7	12.9	0.0	0.0	-	0.0
<i>l02t01</i>	22.4	45.1	18.4	11.7	2.4	0.0	0.0	0.0
<i>l03t02</i>	2.5	95.0	-	2.4	0.1	0.0	0.0	-
<i>l04t01</i>	0.7	0.5	98.8	-	-	-	-	-
<i>l05t01</i>	2.6	80.1	-	17.0	0.0	0.0	-	-
<i>l06t01</i>	37.9	23.8	7.5	30.7	0.0	0.0	0.0	0.0
<i>l12t01</i>	48.3	51.5	0.0	0.1	0.1	0.0	0.0	-

- Strong dominance of integral types; some traces contain in addition a significant portion of octet string data
- It is unclear why read-only string objects (e.g., `ifDescr`) are retrieved over and over again

# Managed Objects

<b>trace</b>	IF	BR	BGP	HR	ENT	CIS	CP
<i>l01t02</i>	40.1	-	17.6	-	10.3	30.4	-
<i>l01t05</i>	99.7	-	-	0.0	-	-	-
<i>l02t01</i>	93.5	5.5	0.0	-	0.1	0.0	-
<i>l03t02</i>	33.3	65.1	0.0	0.0	0.0	0.1	-
<i>l04t01</i>	99.7	-	-	-	-	-	-
<i>l05t01</i>	80.1	-	-	-	-	-	17.0
<i>l06t01</i>	91.3	0.0	0.0	-	0.0	2.0	-
<i>l12t01</i>	50.4	0.0	-	47.7	-	-	-

- The IF-MIB clearly dominates in our traces
- In trace *l06t01*, 32-bit counters dominate and the discontinuity indicator is ignored
- In trace *l02t01*, all columns of the ifTable and the ifXTable are retrieved regularly

# Notifications

- In trace *102t01*, about 52.1% of the notifications are fan failure notifications that are repeated periodically. Some 42.2% are interface up/down notifications while the remaining notifications are HP and Avaya specific
- In trace *103t02*, we found that all notifications were reporting printer problems
- Trace *105t01* contains only Cisco notifications related to TCP session teardowns and configuration changes
- Trace *106t01* has 26.1% BGP and 8.1% PIM routing related notifications. Some 20.0% are sensor threshold crossing notifications while 13.2% are Cisco notifications related to TCP session teardowns
- Trace *112t01* contains 68.5% authentication failure, 14.8% cold start and 16.7% shut down notifications

# Notifications

- In trace *102t01*, about 52.1% of the notifications are fan failure notifications that are repeated periodically. Some 42.2% are interface up/down notifications while the remaining notifications are HP and Avaya specific
- In trace *103t02*, we found that all notifications were reporting printer problems
- Trace *105t01* contains only Cisco notifications related to TCP session teardowns and configuration changes
- Trace *106t01* has 26.1% BGP and 8.1% PIM routing related notifications. Some 20.0% are sensor threshold crossing notifications while 13.2% are Cisco notifications related to TCP session teardowns
- Trace *112t01* contains 68.5% authentication failure, 14.8% cold start and 16.7% shut down notifications

# Notifications

- In trace *102t01*, about 52.1% of the notifications are fan failure notifications that are repeated periodically. Some 42.2% are interface up/down notifications while the remaining notifications are HP and Avaya specific
- In trace *103t02*, we found that all notifications were reporting printer problems
- Trace *105t01* contains only Cisco notifications related to TCP session teardowns and configuration changes
- Trace *106t01* has 26.1% BGP and 8.1% PIM routing related notifications. Some 20.0% are sensor threshold crossing notifications while 13.2% are Cisco notifications related to TCP session teardowns
- Trace *112t01* contains 68.5% authentication failure, 14.8% cold start and 16.7% shut down notifications

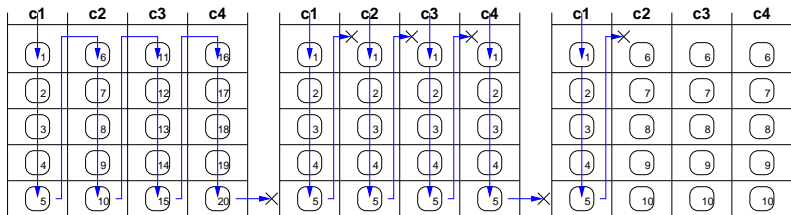
# Notifications

- In trace *102t01*, about 52.1% of the notifications are fan failure notifications that are repeated periodically. Some 42.2% are interface up/down notifications while the remaining notifications are HP and Avaya specific
- In trace *103t02*, we found that all notifications were reporting printer problems
- Trace *105t01* contains only Cisco notifications related to TCP session teardowns and configuration changes
- Trace *106t01* has 26.1% BGP and 8.1% PIM routing related notifications. Some 20.0% are sensor threshold crossing notifications while 13.2% are Cisco notifications related to TCP session teardowns
- Trace *112t01* contains 68.5% authentication failure, 14.8% cold start and 16.7% shut down notifications

# Notifications

- In trace *102t01*, about 52.1% of the notifications are fan failure notifications that are repeated periodically. Some 42.2% are interface up/down notifications while the remaining notifications are HP and Avaya specific
- In trace *103t02*, we found that all notifications were reporting printer problems
- Trace *105t01* contains only Cisco notifications related to TCP session teardowns and configuration changes
- Trace *106t01* has 26.1% BGP and 8.1% PIM routing related notifications. Some 20.0% are sensor threshold crossing notifications while 13.2% are Cisco notifications related to TCP session teardowns
- Trace *112t01* contains 68.5% authentication failure, 14.8% cold start and 16.7% shut down notifications

# Table Retrieval Algorithms



- Three basic algorithms:
  - Column-by-column retrieval
  - Row-by-row retrieval
  - Column-plus-row retrieval
- Issues to consider:
  - Dynamic tables (cells come and go while walking)
  - Holes (some cells might not be present accessible)
  - Time-filtered tables...

# Optimizations

- Several optimizations can be applied to each of the basic algorithms:
    - Index removal
    - GetBulk traversal
    - Request packing
    - Hole padding
    - Pipelined traversal
  - This leads to  $3 \cdot 2^5 = 96$  possible variations!  
(Ignoring special things like time-filtered tables)
- ⇒ We need to be very precise how we analyze traces...

## Definition (general walk)

A general walk or just walk is defined as a sequence of get-next/response or get-bulk/response operations. A mixture of get-next and get-bulk requests is not considered a walk. **At least one object identifier in the sequence of requests at the same varbind index must be increasing lexicographically while all object identifiers at the same varbind index have to be non-decreasing.** The end of a walk is detected if either a request never sees a response or there is no subsequent request where all the object identifiers match a previous response within a given timeout interval.

## Definition (strict walk)

A strict walk is defined as a sequence of get-next/response or get-bulk/response operations. A mixture of get-next and get-bulk requests is not considered a walk. **All object identifiers in the sequence of requests at the same varbind index must be strictly increasing lexicographically. In addition, the object identifiers at the same index of a response and a subsequent request must be identical.** The end of a walk is detected if either a request never sees a response or there is no subsequent request where all the object identifiers match a previous response within a given timeout interval.

# Prefix-constrained Walk

## Definition (prefix-constrained walk)

A prefix-constrained walk is defined as a sequence of get-next/response or get-bulk/response operations. A mixture of get-next and get-bulk requests is not considered a prefix-constrained walk. **At least one object identifier in the sequence of requests at the same varbind index must be increasing lexicographically and all object identifiers at the same varbind index have to be non-decreasing and all object identifiers at the same varbind index have the same object identifier prefix. This prefix is established by the first request within the walk.** The end of a walk is detected if either a request never sees a response or there is no subsequent request where all the object identifiers match a previous response within a given timeout interval.

# Properties

- 1 A strict walk is always also a general walk.
  - 2 A strict walk is not necessarily a prefix-constrained walks.
  - 3 A general walk is not necessarily a strict walk.
  - 4 A general walk is not necessarily a prefix-constrained walk.
  - 5 A prefix-constrained walk in not necessarily a strict walk.
  - 6 A prefix-constrained walk is always also a general walk.
- Questions:
    - Are the walk definitions reasonably sound and complete or do we need more?
    - Do we need to distinguish between prefix-constrained and strictly prefix constrained walks?

# Properties

- ① A strict walk is always also a general walk.
  - ② A strict walk is not necessarily a prefix-constrained walk.
  - ③ A general walk is not necessarily a strict walk.
  - ④ A general walk is not necessarily a prefix-constrained walk.
  - ⑤ A prefix-constrained walk is not necessarily a strict walk.
  - ⑥ A prefix-constrained walk is always also a general walk.
- Questions:
    - Are the walk definitions reasonably sound and complete or do we need more?
    - Do we need to distinguish between prefix-constrained and strictly prefix constrained walks?

## Definition (volume of a walk)

The volume  $v(w)$  of a walk  $w$  is the number of varbinds retrieved in all response messages included in walk  $w$ .

## Definition (start, end, duration of a walk)

Let  $w$  be a walk. Then  $s(w)$  denotes the start time of the walk  $w$  (timestamp of first request) and  $e(w)$  denotes the end of the  $w$ . The duration  $d(w)$  of the walk  $w$  is given by  $d(w) = e(w) - s(w)$ .

## Definition (concurrent walks)

Two walks  $w_1$  and  $w_2$  are concurrent at time  $t$  if  $s(w_1) \leq t \leq e(w_1)$  and  $s(w_2) \leq t \leq e(w_2)$ .

# Walk Metrics (cont.)

## Definition (concurrency level)

The concurrency level  $C_F(f, t)$  of a flow  $f$  at time  $t$  is given by current walks in flow  $f$  at time  $t$ . The concurrency level  $C_M(m, t)$  of a manager  $m$  is given by the sum of the concurrency levels  $C_F(f, t)$ , where the flows  $f$  originate at the manager  $m$ .

$$C_M(m, t) = \sum_{f \text{ originating at } m} C_F(f, t)$$

## Definition ( $\Delta$ -time serial walks)

Two walks  $w_1$  and  $w_2$  are  $\Delta$ -time serial walks if the  $s(w_2) - e(w_1) < \Delta$ .

# Current Status...

- Implementation of scripts that can extract walks (from flows) and classify them
- Usage of SQL databases to store intermediate results for further processing and analysis
- Interesting and unexpected results are caused by either
  - bugs in the code or
  - interesting message sequences in the tracesand this needs manual validation
- We will have more results to present at a future meeting...
- **You can help by getting us more traces to analyze!!**

# References



J. Schönwälder, A. Pras, M. Harvan, J. Schippers, and R. van de Meent.

**SNMP Traffic Analysis: Approaches, Tools, and First Results.**

*In Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management, May 2007.*



J. Schönwälder.

**SNMP Traffic Measurements.**

Internet Draft (work in progress) <draft-schoenw-nmrg-snmp-measure-01.txt>, International University Bremen, February 2006.