Core technology since 1996

SNMP 개념 이해

Understanding of SNMP concept

Release 1.0

2010/4/29

novo networks

목 차 (Table of Contents)

1. Audience

본 자료는 SNMP 관련 프로젝트를 통해 얻은 경험을 공유하고자 작성되었습니다. SNMP 에 대해 처음 접하시는 분들이나 개념이 익숙하지 않으신 분들이 본 자료의 대상입니다. 사실 무슨 내용이든지 처음 접근 하고자 할 때 어디서부터 시작해야 할지 난감해 지는 경우가 많습니다. 더구나 SNMP 는 주로 영문으로 작성된 RFC 규격 형태로 정보가 제공되기 때문에 전반적인 개념을 한눈에 파악한다는 것은 매우 많은 노력과 시간을 요구합니다. 따라서, 이러한 어려움에 조금이나마 도움을 드리고자 하는 것이 이 문서의 목적입니다. 아무쪼록 SNMP 에 대한 이해에 보탬이 되면 좋겠다는 바람입니다.

2. SNMP 를 설명하기에 앞서

SNMP 는 'Simple Network Management Protocol' 의 약어입니다.
그 뜻을 우리 말로 풀이해보면 '네트워크에 존재하는 다양한 장비들을 관리하기 위해 사용되는 간단한 프로토콜' 이라고 해석이 되지요.
(이 말이 바로 이해가 되셨다면, 본 절을 보실 필요가 없습니다. 다음 절로 바로 넘어가세요)

근데, 막상 해석을 해놓고 보니, 분명히 우리 말로는 되어있는데 도대체 무슨 말인지 머리에 와 닿지는 않습니다.
혹시, 지금 머릿속에서 아래와 같은 의문점이 연기처럼 몽실몽실 피어 오르시나요?

"도대체 네트워크는 뭐지?"
"장비를 관리한다는 건 또 뭐람?"
"프로토콜은 뭐 하는 거야?"

그렇다면, 일단 SNMP 는 잠시 잊어버리도록 하고, 위에서 생겨난 의문점들부터 살펴보죠.

네트워크란 간단히 '인터넷'을 생각하시면 됩니다.
인터넷 세상에서 우리는 PC 앞에 앉아 웹 브라우저를 띄웁니다. 그리고 원하는 곳을 click 하죠.
순식간에 목적지 site 에 도착합니다. 내가 보고 있는 site 가 국내에 있든지 해외에 있든지 간에 손을 뻗으면 닿는 거리에 있는 것처럼 가깝게 느끼게 되죠. 이와 같이 내가 있는 곳과 내가 원하는 곳, 그리고 그 사이를 이어주는 다양한 기기들이 그물처럼 연결되어 있는 곳을 가리켜 네트워크 라고 합니다.
인터넷은 하나의 거대한 네트워크입니다.

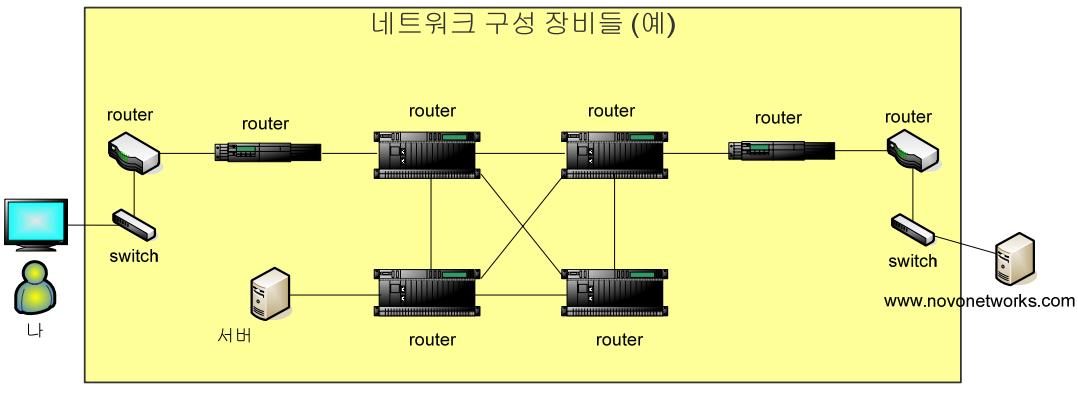
그렇다면 장비는 무엇이고 그것을 왜 관리할까요?

다시 인터넷을 예로 들겠습니다.

우리는 PC 앞에 앉아 웹 서핑을 즐깁니다. 그러나, 웹 서핑이 공짜는 아니죠. KT, SK, LG… 뭐 이런데 중에 한군데는 가입을 하여야 하고, 매달 인터넷 사용료를 내야 합니다.

이와 같이 사용료를 받고 인터넷 서비스를 제공하는 회사들을 인터넷 서비스 공급업체 (ISP, Internet Service Provider)라고 합니다. ISP 들은 가입자들에게 인터넷 서비스를 공급하는 대신, 그들이 사용하는 네트워크에 문제가 생기지 않도록 해야 할 책임이 있습니다.

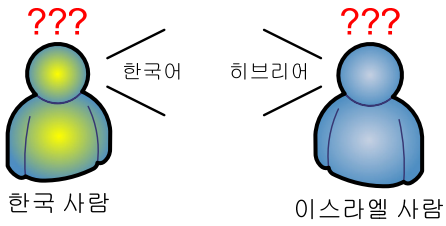
한편, 가입자들이 인터넷을 원활하게 사용할 수 있도록 하기 위해서는 생각보다 많은 장비들이 필요합니다. 대표적인 장비로는 스위치(switch), 라우터(router), 서버(server)가 있습니다. 아래 그림은 인터넷에 존재하는 장비들의 간단한 예 입니다.



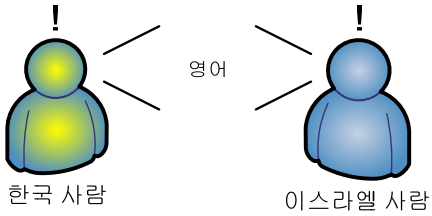
사용자들은 가급적 인터넷이 항상 정상 동작하기를 바랍니다. 혹시라도 네트워크에 문제가 생겼다면 매우 빠른 복구를 원할 거고요. 다시 말해, ISP 입장에서 네

트워크의 안정성을 보장하는 것은 그만큼 고객들에게 수준 높은 서비스를 제공하게 되는 겁니다.
결국, 네트워크의 안정성을 보장하기 위해서는 네트워크에 존재하는 장비 하나 하나에 대해 무슨 일이 일어나는 지 일거수 일투족을 알아야 하는데 이를 일컬어 '네트워크 관리'라고 합니다. 네트워크 관리를 통해 수많은 장비들 중에 어떤 장비가 죽었는지, 또 어떤 장비가 잘못 동작하는 지를 최대한 빨리 알아내어 빠른 조치를 취할 수 있습니다.

마지막으로 프로토콜입니다.
프로토콜이란 서로 다른 두 노드간에 '우리 서로 이렇게 통신을 하자' 하고 약속을 한 것을 의미합니다. 이를 어려운 말로 '표준화된 통신규약'이라고 합니다.
앞에서와 마찬가지로 간단한 예를 들어 보겠습니다.
한국 사람과 이스라엘 사람이 만날 일이 생겼습니다.
만약 이들이 만나서 각기 자기나라 말을 쓴다면 어떻게 될까요?



서로 간의 의사 소통이 안 될 것입니다.
한 사람은 한국말로 물어보고 다른 한 사람은 히브리어로 대답하니 서로 대화가 될 턱이 없지요.
그러나, 두 사람이 "우리는 만나면 언제나 영어를 사용합시다" 라고 약속한다면 서로 간의 대화가 가능해 지고, 서로에게 필요한 것을 얻을 수 있을 겁니다.
이럴 경우, 두 사람 사이의 프로토콜은 '영어'가 됩니다.



너무 다 아는 얘기라 지루하셨나요?

하하... 하지만 시작이 반이랍니다.

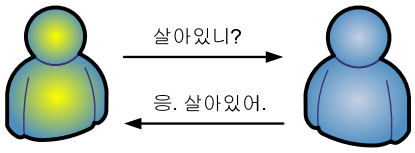
3. SNMP 의 탄생 배경

지금은 인터넷의 발달로 인터넷이 하나의 생태계로 불릴 만큼 복잡해 졌습니다. 하지만, 1970 년대만 해도 네트워크는 극 소수의 사람들만이 사용하였죠. 웹도 없었고 네트워크를 통해 할 수 있는 일이라고는 간단한 데이터를 서로 주고 받는 정도였습니다.

또한, 네트워크가 조그마하다 보니 이를 구성하는 장비들도 많지 않았기 때문에 네트워크 관리라는 개념 자체가 중요하지 않았습니다. 당시 네트워크를 사용하는 사람들의 대부분 관심사는 '내가 보낸 데이터가 목적지에 제대로 도착 했을까?' 정도였답니다. 따라서 '어떤 데이터가 목적지까지 갈수 있다 혹은 없다' 정도만 데이터를 보낼 사람에게 알려주면 충분했습니다.

이와 같은 역할을 하기 위해 태어난 프로그램이 PING (Packet Internet Groper) 입니다.

PING 은 내가 목적지의 IP address 를 아는 경우에 사용하는데 다음과 같이 동작합니다



내가 목적지로 '너 살아 있니?' 라는 질문이 담긴 메시지를 보냅니다

목적지에서 정상적으로 메시지를 받으면 '응. 나 살아있어' 라는 응답이 담긴 메시지를 내게 되돌려 보냅니다

응답 메시지를 받은 경우, 나는 목적지와 통신이 가능하다는 것을 확인한 것이 됩니다

반면에, 처음 보낸 메시지를 목적지에서 못 받았거나, 목적지의 응답 메시지를 내가 못 받을 경우 일정 시간 동안 기다립니다. 시간이 초과되면 나는 목적지와 통신이 불가능하다는 것을 확인한 것이 됩니다

1980 년대 말이 되자 인터넷의 사용 인구가 폭발적으로 증가하기 시작 했습니다. 장비는 많아지고, 네트워크는 더욱 복잡해졌지요. 네트워크에 문제가 생겨도 어느 장비에서 어떤 문제가 있는지 이제는 쉽사리 찾아 내기가 힘들어 졌습니다. 그러다 보니 문제점을 찾는데 너무 많은 시간과 노력이 필요하게 되었고요. 드디어 '네트워크 장비들을 관리' 해야 할 필요성이 나타나기 시작한 겁니다.

결국 1987 년 11 월 SGMP (Simple Gateway Monitoring Protocol) 라는 프로토콜이 생

겨 났습니다. SGMP 는 다음과 같은 3 가지 요구 사항에 초점을 맞추었습니다

- PING 보다 더 많은 기능이 있어야 한다
- 보다 직관적이고 쉽게 네트워크 문제를 찾을 수 있어야 한다
- 표준화된 프로토콜이어야 한다

그리고 SGMP 는 후에 SNMP 로 진화하게 됩니다.

4. CMIP 과 SNMP

1980 년대가 되자 유럽과 북미의 네트워크 전문가들은 앞으로 다가올 네트워크 시대를 준비해야 한다는 데 생각을 모았습니다. 각 나라들이 서로 다른 방식의 프로토콜을 사용하는 경우 전 세계를 하나의 네트워크로 묶는 데 너무 많은 비용이 든다고 판단했기 때문이죠. 그래서 이들은 하나의 국제 표준 프로토콜을 정의하기 위한 표준화 작업에 착수하게 됩니다. 이를 통해 다양한 국제 표준 프로토콜을 내놓게 되는데요. 이 들 중에 표준 네트워크 관리 프로토콜이 바로 CMIP (Common Management Information Protocol)입니다.

더 나아가 1988 년 초, 인터넷의 표준을 결정하는 기관인 IAB (Internet Architecture Board)가 다음과 같은 결정을 내리게 되면서 SNMP 는 곧 사라질 운명에 처하게 됩니다

- CMIP 이 향후 사용될 표준 네트워크 관리 프로토콜이다
- SNMP 는 단기적인 솔루션으로만 사용한다

그러나, 현실 세계에서의 이야기는 IAB 의 결정과는 정 반대 방향으로 흘러가고 있었습니다. 유럽과 북미가 자신들에게 이로운 규격을 넣으려고 서로 다투다 보니 표준화 작업은 계속 지지부진해 졌습니다. 반면에, 인터넷은 하루가 다르게 복잡해져 갔지요. 사업자들은 결국 SNMP 를 사용할 수 밖에 없었고 시간이 흐르면서 SNMP 가 네트워크 관리 프로토콜의 대세로 굳어져 갔습니다. 결국 표준화 작업이 완성되었을 때에는 이미 SNMP 가 암묵적인 표준이 되어 있었던 것입니다. 마치 TCP/IP 가 OSI 7-layer 를 기다리다가 표준이 된 것처럼요.

5. 네트워크 관리

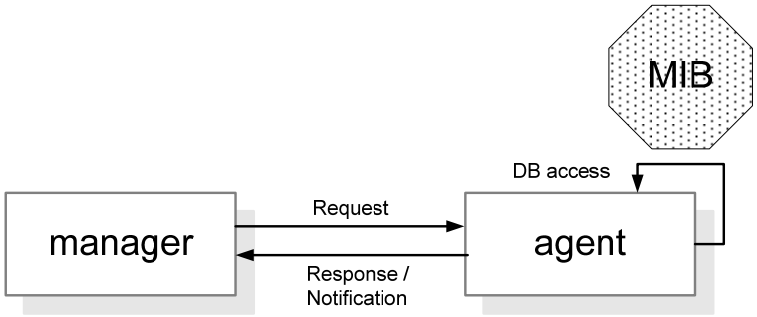
SNMP 는 앞에서 설명했듯이 네트워크를 관리하기 위한 표준 통신 규약입니다. 다시 말해 SNMP 는 표준화된 약속을 제공할 뿐이지 SNMP 자체에서 네트워크 내에 존재하는 장비를 관리하지는 않습니다.

그렇다면, 네트워크 관리를 이해하기 위해서 SNMP 말고 무엇을 더 알아야 할까요? 바로, 아래의 3 가지를 더 알아야 합니다. 그것은

- Manager,
- Agent, 그리고
- MIB (Management Information Base)*

입니다.

(* MIB 는 '밉'이라고 읽기도 하지만 여기서는 M.I.B 라고 하겠습니다).



Manager 는 필요한 정보를 agent 에 요청하는 역할을 합니다. 그러나 manager 도 소프트웨어 이므로 결국 사람의 요청을 manager 가 대신 수행한다고 보시면 됩니다. 일반적으로 하나의 manager 는 독립된 장비(PC 혹은 서버)로서 다수의 agent 를 관리합니다.

반대로 agent 는 manager 가 요청한 정보를 수집하여 제공하는 역할을 합니다. 또한, 특별한 일이 있을 경우 manager 의 요청이 없더라도 그 정보를 manager 에게 보고합니다 (이를 notification 이라고 합니다). 일반적으로 agent 는 관리 대상 장비(switch, router, 혹은 server)를 나타냅니다.

마지막으로 MIB 는 실제로 필요한 정보가 저장되어 있는 database 입니다. 이 정보는 agent 에 저장되어 있으며, manager 가 agent 에게 요청하면 agent 는 이 database 에서 필요한 정보를 얻은 후, 이를 manager 에게 알려줍니다.

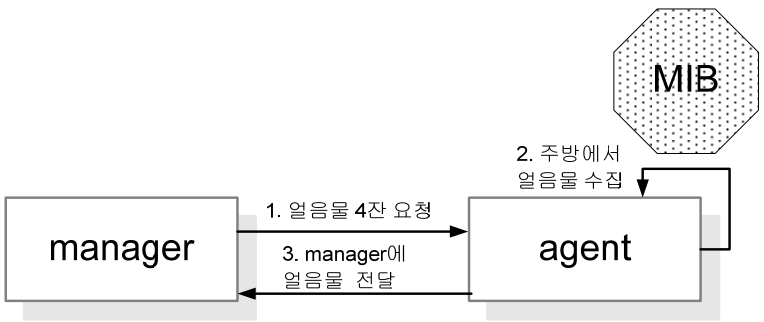
조금 어렵나요?

그럼 아래의 이야기로 예를 들겠습니다.

'시내에 노보 레스토랑이 문을 열었습니다. 음식 맛이 아주 일품이라고 합니다.
하도 소문이 자자하여 노보 레스토랑에서 저녁 식사를 하기로 했습니다.
식당 앞에 도착하니 지배인 **(manager)** 이 저를 맞이하였습니다.
"어서 오십시오. 노보 레스토랑입니다. 무엇을 도와 드릴까요?"
저는 지배인 에게 4 명이 식사를 하러 왔노라고 이야기 했습니다.
그러자 지배인은 우리를 전망이 매우 좋은 자리로 안내 했습니다.
그러고는, 예쁜 웨이트리스 **(agent)** 를 불러서 즉시 지시를 내렸습니다.
"효리씨, 이 손님들께 드릴 시원한 얼음물을 가지고 오세요."
웨이트레스 **(agent)** 는 즉시 식당 부엌 **(MIB)** 으로 가서 4 개의 컵에 시원한 얼음물을 담아서 지배인 **(manager)** 에게 가져 왔습니다. 지배인은 그 얼음 물을 일행 앞에 하나씩 내려 놓은 후 식사 주문을 받았습니다.

얼마의 시간이 지난 후, 지배인은 주문한 음식과 함께 나타났습니다.
그리고는, 매우 능숙한 솜씨로 우리에게 음식을 제공하기 시작 했습니다.
바로 그 때, 아까 본 예쁜 웨이트리스 **(agent)** 가 허겁지겁 지배인 **(manager)** 에게 다가와 중요한 사실을 전했습니다 **(notification)**.
"지배인님. 주방장님이 갑자기… 쓰러지셨습니다."
지배인은 저희에게 잠시 양해를 구하고 주방으로 달려가 주방장에게 응급 조치를 취했습니다'

위의 예에서 manager 는 agent 에게 요청 (얼음물 4 잔) 을 합니다. Agent 는 manager 의 요청을 받아 MIB 에서 필요한 정보를 가져옵니다.

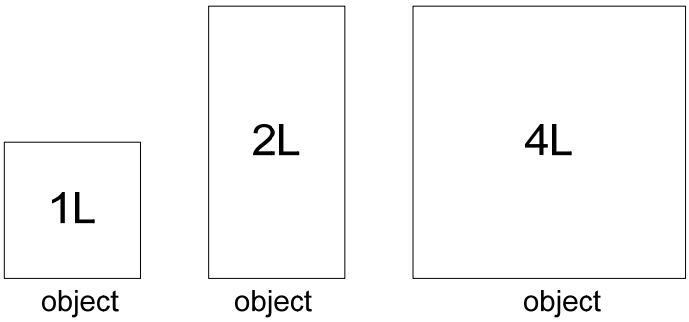


뜻하지 않을 일 (주방장이 쓰러진 일)이 일어난 경우 agent 는 manager 에게 notification 을 보냅니다.

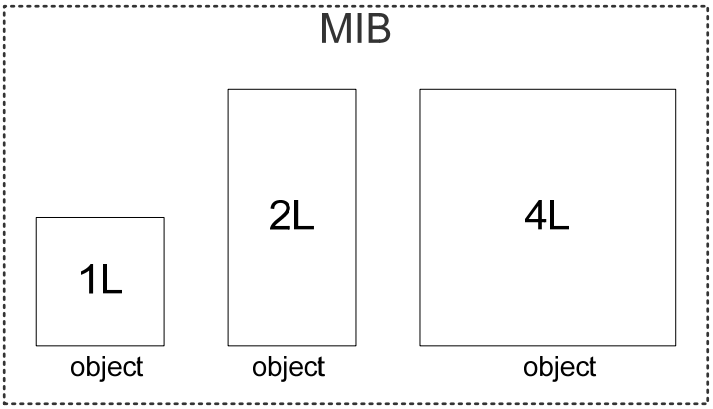
그럼 다음 절에서 MIB 에 대해 알아보죠.

6. MIB (Management Information Base)

MIB 는 agent 가 관리 대상 정보를 저장하는 방식입니다.
예를 들어, 아래 그림과 같이 3 개의 물병이 있다고 하죠

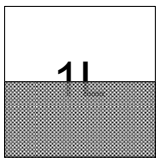


이때, 각각의 물병을 object 라고 합니다.



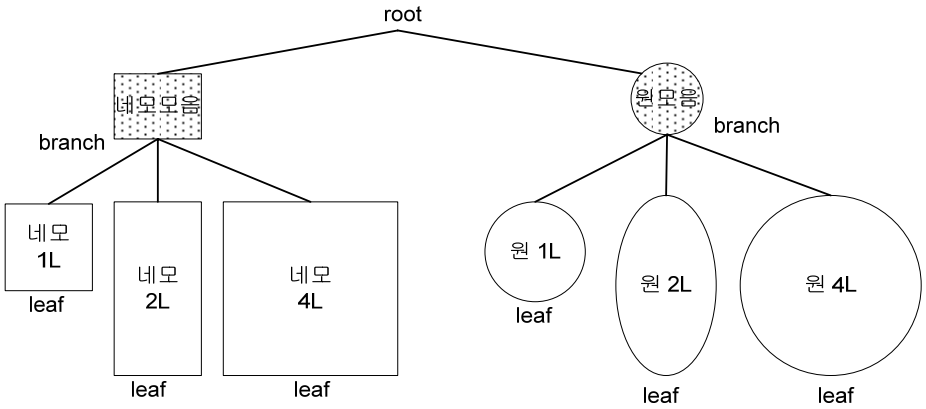
그리고 각각의 object 가 모두 모여서 MIB 가 되는 겁니다.

여기서 주의해야 할 것은, object 가 실제 값을 나타내는 것은 아니라는 점입니다. 1L 짜리 물병에 물이 항상 1L 가 있는 게 아니듯이 말이죠. 도대체 무슨 말이냐고요?
매우 중요한 부분이니 다시 한번 예를 들어 설명해 보겠습니다. 아래의 그림을 보시죠.



1L 물병에 물이 절반 차 있을 경우, 가지고 있는 물은 500mℓ 가 됩니다.
이와 같이 '1L 들이 물병'은 object 이지만, 그 안에 채워진 실제 값 (즉, 500mℓ 의 물)은 instance (혹은 instance value) 라고 합니다. 다시 말해, '1L 들이 물병'은 그 형태가 정의되어 있지만, 그 안에 채워지는 물의 양은 항상 바뀔 수 있는 거죠. 앞으로 이어질 설명들에서도 object 와 instance는 수시로 언급되게 됩니다. 그럴 때마다 '물이 절반 채워져 있는 물병'을 머리에 떠올리세요.

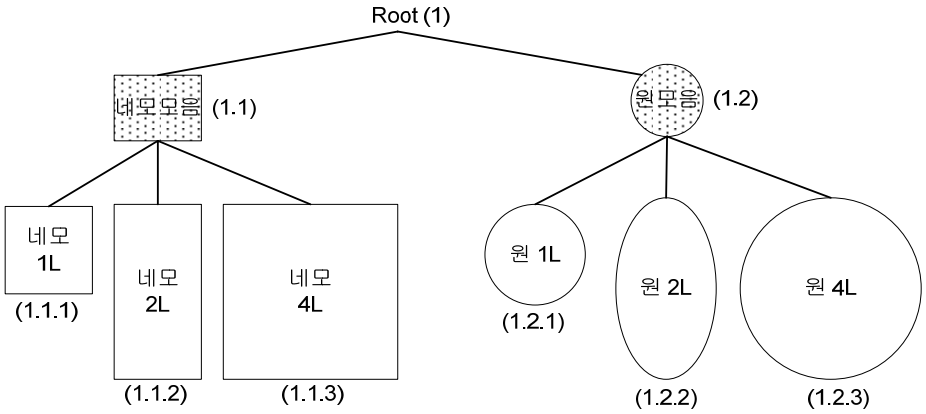
그럼 이번에는 MIB 의 구조에 대해서 살펴보도록 하겠습니다.
MIB 는 아래 그림과 같이 tree 구조로 되어 있습니다.



그림에서 MIB tree의 시작점은 root 라고 합니다.
Root 밑에는 branch node 와 leaf node 로 구성되어 있습니다.
자식을 가지고 있는 node 를 branch node 라고 합니다. 그림에서 '네모 모음'은 branch node 입니다.
자식이 없는 node 를 leaf node 라고 합니다. 그림에서 '네모 1L'는 leaf node 입니다.
Branch node와 leaf node는 모두 object 입니다. 반면에, leaf node만이 instance를 갖습니다. 즉, branch node는 leaf node를 표현하기 위한 중간 node 역할만 합니다.

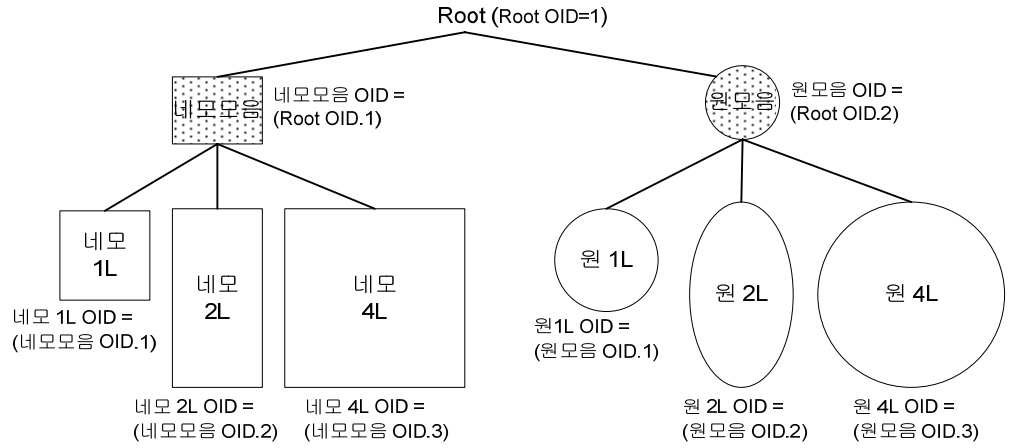
각 object 는 고유의 object name 과 object id 를 갖습니다 (object id 는 줄여서 OID 라고도 합니다).
Object id 는 root 로부터 시작하여 leaf node 에 이르기까지의 id 를 '.'으로 구분하여 나타냅니다.

또한, object id 는 모든 object 들을 유일하게 표현할 수 있는 방법입니다.
아래 그림을 보시죠.



그림에서 object name 이 '네모 1L'인 경우, object id 는 (1.1.1) 입니다.
Object name 이 '네모 모음'인 경우, object id 는 (1.1) 입니다.
또한 그림에서 알 수 있듯이, 모든 object 는 자신만의 유일한 object id 를 가지고 있습니다.

위의 그림에 나타난 object id 는 아래와 동일한 표현입니다.



7. Instance id

앞 절을 보고 나니 한가지 궁금증이 생깁니다.
앞부분에서는 object 와 instance 가 서로 다르다는 사실이 매우 중요하다고 했는데요. 막상 끝날 때는 object id 만 달랑 설명하고 말았습니다.
혹시, object id 말고 instance id 도 있어야 하는 것 아닐까요?

네 그렇습니다. 그게 바로 이번 절에서 설명할 내용이기도 하지요.
사실, object 에 대한 이해가 되지 않고서는 instance 를 제대로 이해하기가 쉽지 않습니다. 마찬가지로 object id 에 대한 이해가 없이는 instance id 에 대한 이해 역시 쉽지 않지요.
더군다나 instance id는 SNMP 내용 중에서 가장 중요하면서도 어려운 부분이니 주의 깊게 내용을 봐 주시길 부탁 드립니다.
자 이제 설명을 시작하도록 하겠습니다.

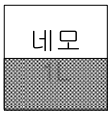
만일 manager 가 어떤 object 의 instance 를 얻고 싶다면, manager 는 그 object 의 instance id 를 알아야만 합니다.
Object 는 형태에 따라 아래와 같이 2 가지로 나뉘어 집니다

- scalar object : 하나의 값을 표현
- table object : table 형태의 값을 표현

지금까지 예를 들었던 object 들은 모두 scalar object 입니다. 다시 그림으로 예를 들어보죠
Scalar object 는 하나의 leaf node 형태입니다. 하나의 object 에 하나의 instance 가 존재하죠.

scalar object
1 object
1 instance
leaf node

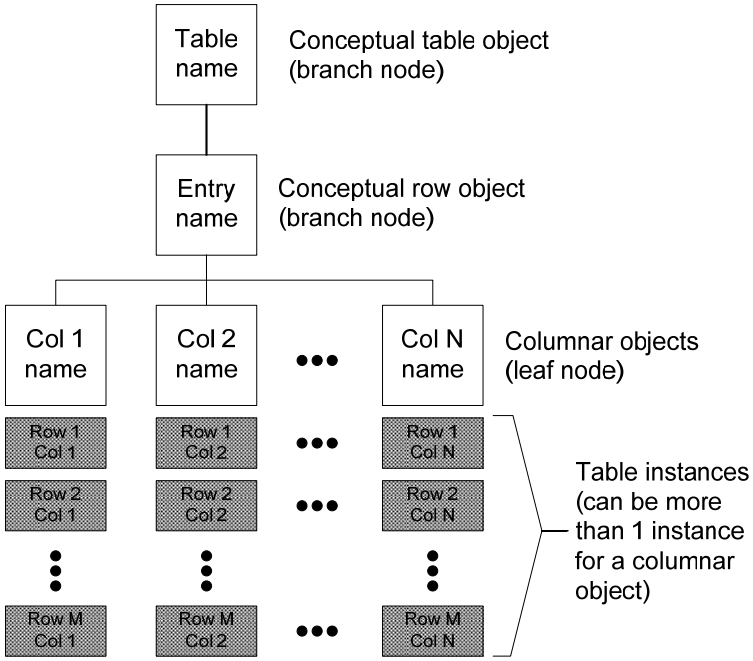
따라서, 하나의 object id 와 하나의 instance id 를 갖습니다



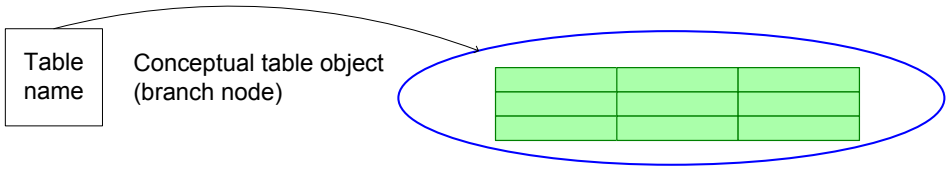
Object id = (1.1.1)
Instance id = (1.1.1.0)
Instance value = 0.5L

Scalar object의 경우 instance id는 object id + '.0' 로 표현됩니다.
(사실 scalar object 의 경우, object id 자체가 유일하므로 object id 를 instance id 로 사용하여도 되겠지만, object 와 instance 를 명확히 하기 위해 object id 와 instance id 를 서로 구분합니다.)
위의 그림에서와 같이 '네모 1L' object 의 object id 는 (1.1.1)이고 instance id 는 (1.1.1.0) 이며 instance value 은 0.5L 입니다.

반면에 table object 는 좀 복잡한 구조인데요. 다음과 같은 형태를 갖습니다



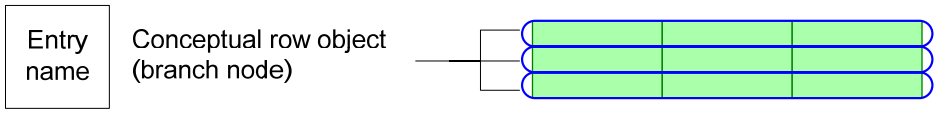
우선, '이 object 는 table 형태'라는 것을 나타내는 무언가가 필요합니다.
즉, 실제 값은 가지고 있지 않지만, table 이라는 상징적 개념을 갖는 object 가 필요하죠.
이를 conceptual table object 라고 합니다.



그 아래는 table 의 구성 요소인 row 와 column 을 나타내는 object 들로 구성되어 있습니다.

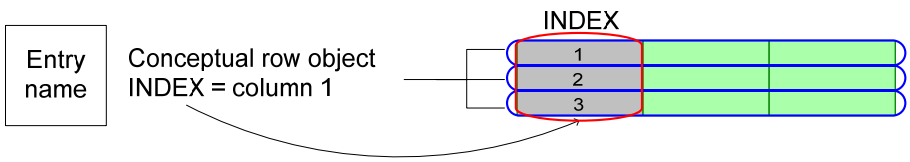
우선, conceptual row object 는 conceptual table object 와 마찬가지로 실제 값은 가지고 있지 않습니다. 대신에, 각 row 를 구분하기 위해 어떤 columnar object 를 key 로 사용할 것인가를 정의합니다.

이를 INDEX 라고 합니다.

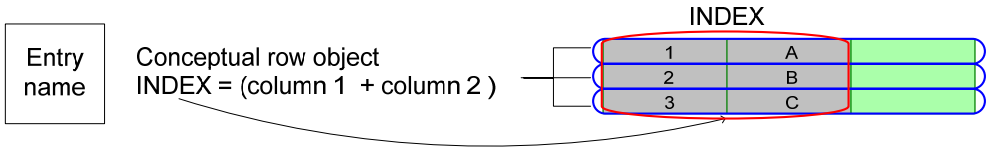


INDEX 는 columnar object 의 instance value 로 표현 됩니다.

예를 들어, 각 row 를 구분하기 위해 첫 번째 column 을 INDEX 로 사용하고자 한다면, conceptual row object 에서 이를 정의합니다.



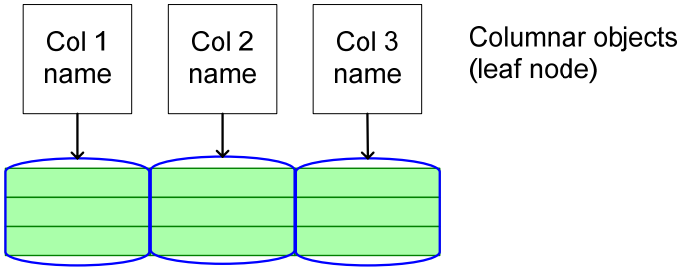
INDEX 는 여러 개의 instance value 의 조합으로도 표현이 가능합니다. 예를 들어, 첫 번째 column 과 두 번째 column 을 INDEX 로 사용하고자 한다면, 첫 번째 column 의 instance 뒤에 두 번째 column 의 instance 를 붙여서 사용합니다.



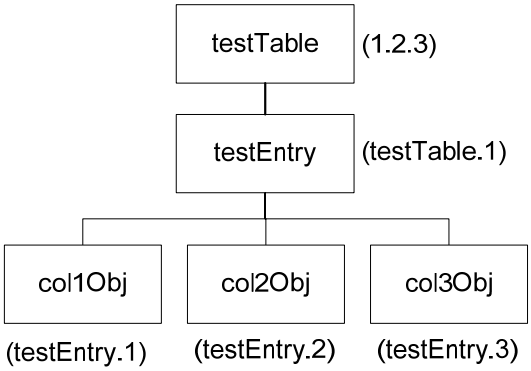
위의 그림에서 첫 번째 column 의 INDEX 는 (1.'A') 가 됩니다. 두 번째 column 의 INDEX 는 (2.'B')가 됩니다.

이와 같이 INDEX 로 각 row 를 구분할 수 있으므로, manager 는 INDEX 를 이용하여 특정 row 에 대한 접근이 가능합니다.

마지막으로 columnar object 는 table 의 각 column 을 나타내는 object 입니다. Columnar object 는 scalar object 와 마찬가지로 leaf node 이므로 instance value 를 가질 수 있습니다. 다만, scalar object 는 하나의 object 에 하나의 instance 를 가지는 구조임에 반해 columnar object 는 하나의 object 에 다수의 row instance 를 가질 수 있다는 점이 다릅니다.



자, 그럼 table object 의 object id 와 instance id 를 살펴보도록 하겠습니다.
아래의 그림을 보시죠.



위의 그림에서 보는 바와 같이 conceptual table object 'testTable' 의 object id 는 '1.2.3' 입니다.
Conceptual row object 'testEntry'의 object id 는 '1.2.3.1'이 되고요.
Columnar object 인 'col1Obj', 'col2Obj', 'col3Obj'의 object id 는 각각 '1.2.3.1.1', '1.2.3.1.2', '1.2.3.1.3'이 됩니다.

그럼 instance id 는 어떻게 표현될까요?
앞에서 설명된 table object 에 대해 아래와 같은 instance 들이 있다고 가정해 보죠.

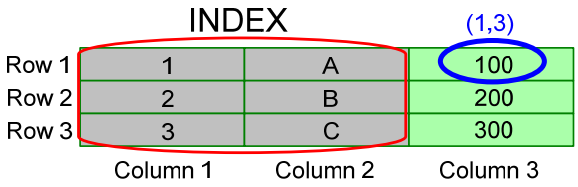


Table object 에 대한 instance id 는 다음과 같이 구성됩니다:

Instance id = 해당 column 의 columnar object id + 해당 row 의 INDEX

위의 그림에서 1 번째 row, 3 번째 column (앞으로 (1,3)으로 표현)의 instance id 는

(1,3)의 instance id = (3 번째 column 의 object id) + (1 번째 row 의 INDEX)

= (1.2.3.1.3) + (1.'A')

= (1.2.3.1.3.1.'A')

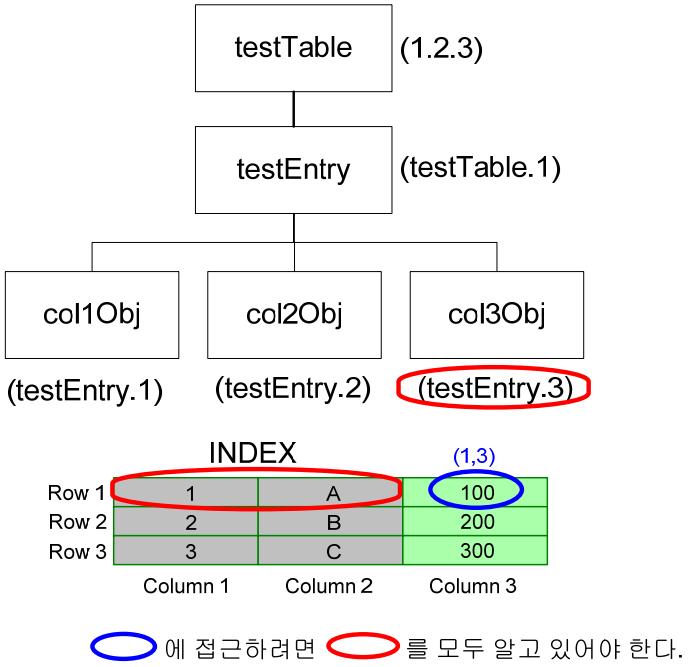
이 됩니다.

8. Lexicographical ordering

앞에서 언급했듯이, MIB 는 유일한 object id 를 갖는 object 들로 구성되어 있는 tree 입니다. 또한, MIB 내의 특정한 instance 에 접근하려면 instance id 를 알아야 합니다.

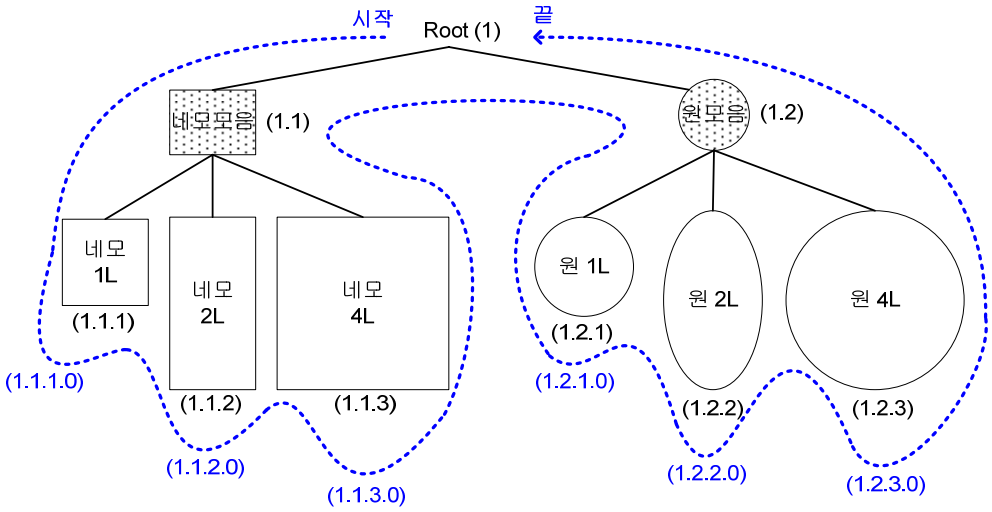
그런데 manager 입장에서 곰곰이 생각해보니 이것은 정말로 불편하기 짝이 없습니다.

Manager 가 table 같이 복잡한 구조의 instance 에 접근하려면 columnar object id 뿐만 아니라 INDEX 값까지 정확히 알고 있어야 하니까요. INDEX 가 복잡하기라도 하면 내용을 미리 알고 있기란 결코 쉬운 일이 아닙니다. 좀 단순한 방법이 어디 없을까요?



이미 짐작하셨겠지만 방법이 있습니다. 그런데 그 방법을 설명하기에 앞서 꼭 알아야 할 것이 하나 있는데, 바로 lexicographical ordering 입니다.

Lexicographical ordering 은 MIB tree 의 instance 들이 순회하는 순서를 정의하는 방식으로 'depth first search'라고도 합니다. 순서는 root 에서 시작하여 왼쪽 에서 오른쪽으로 순회합니다. 다음 그림을 보시죠.

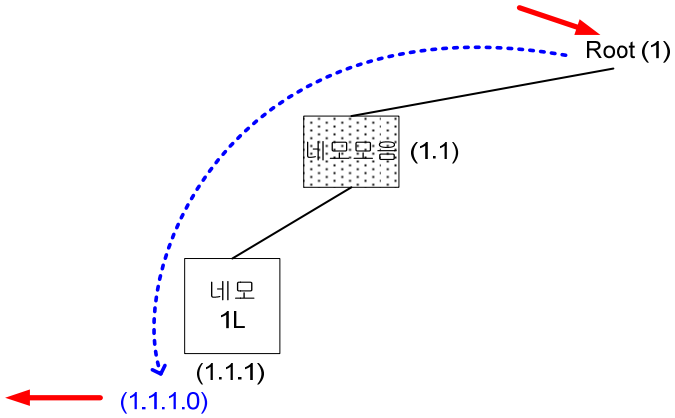


위의 그림에서 검정색 id 는 object id 이고 파란색 id 는 instance id 입니다.
Lexicographical ordering 을 이용한 instance 순회 순서를 instance id 로 나타내면 다음과 같습니다 :

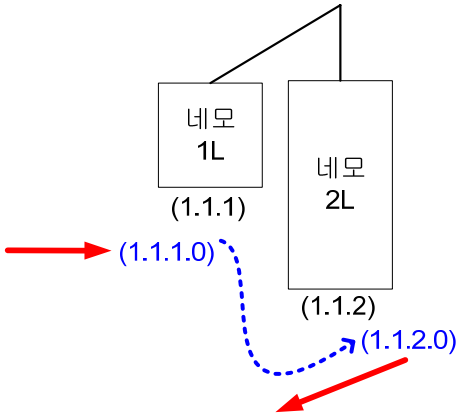
(1.1.1.0) → (1.1.2.0) → (1.1.3.0) → (1.2.1.0) → (1.2.2.0) → (1.2.3.0)

이때, 위와 같은 순회 순서가 가능하기 위해서는 모든 object 의 instance id 가 MIB tree 에 오름차순으로 정렬되어 있어야 합니다. 즉, tree 에서 항상 왼쪽이 작고 오른쪽이 커야 합니다.

Lexicographical ordering 덕분에 SNMP 에서는 manager 가 agent 에게 '내가 요청한 object id 또는 instance id 에서 가장 근접한 instance id 와 instance value 를 달라'는 요청이 가능한데요. 이를 GetNext request 라고 합니다.
그럼 GetNext request 를 했을 경우 그 결과가 어떻게 나오는지 예와 그림을 통해서 살펴 보겠습니다. 먼저, 위의 그림과 같이 scalar object 인 경우입니다.
manager 가 agent 에게 root object id (1) 로 GetNext request 를 할 경우, agent 는 이 object id 에서 가장 근접한 instance id 인 (1.1.1.0)과 해당 instance value 를 manager 에게 되돌려 줍니다



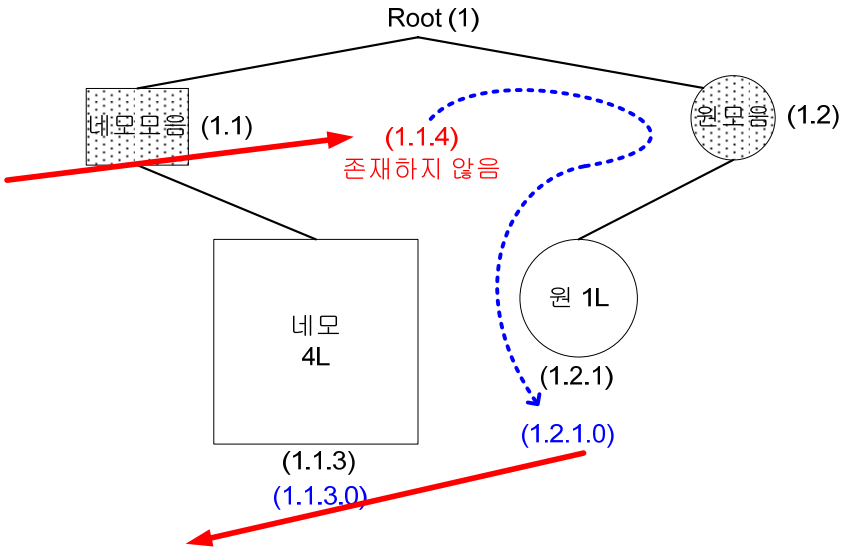
manager 가 agent 에게 instance id (1.1.1.0) 으로 GetNext request 를 할 경우, agent 는 이 instance id 에서 가장 근접한 instance id 인 (1.1.2.0)과 해당 instance value 를 manager 에게 되돌려 줍니다.



그렇다면, 만일 존재하지 않는 object id 에 대해 GetNext request 를 할 경우는 어떻게 될까요?

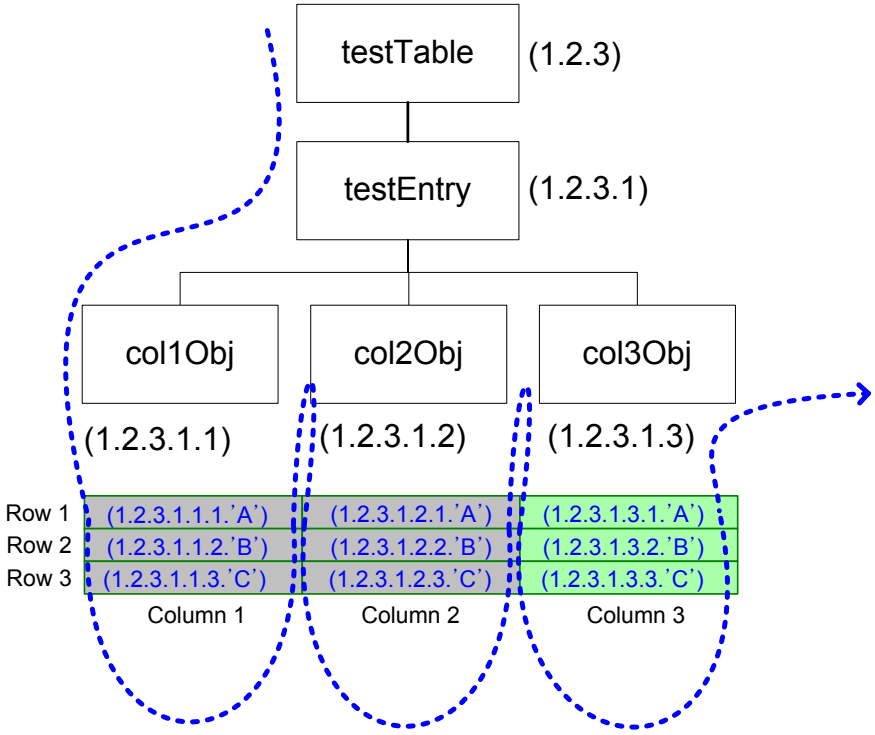
다음의 예를 보시죠.

manager 가 agent 에게 존재하지 않는 object id (1.1.4)로 GetNext request 를 할 경우, agent 는 이 object id 에서 가장 근접한 instance id 인 (1.2.1.0)과 해당 instance value 를 manager 에게 되돌려 줍니다.



다음은 table 인 경우입니다.

아래의 그림은 7 절 맨 처음 그림에 표현된 table 을 lexicographical ordering 으로 instance 순회한 결과 입니다.



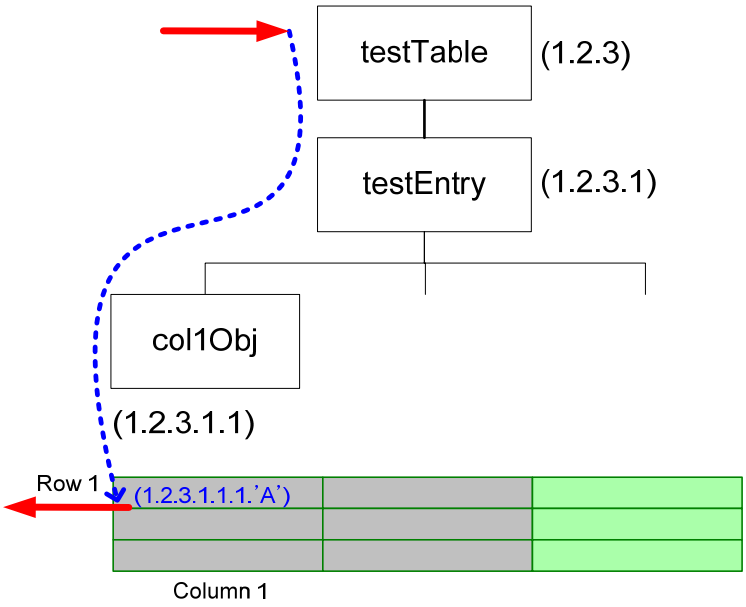
위의 그림에서 눈 여겨 보아야 할 부분은, table 의 모든 instance id 들도 scalar 의 경우와 마찬가지로 MIB tree 내에서 오름차순으로 정렬되어 있어야 한다는 사실입니다. 즉, table 안의 instance id 들도 항상 왼쪽이 작고 오른쪽이 커야 합니다.

그 이야기는 다시 말해 columnar object id 와 INDEX가 모두 오름차순으로 정렬되어 있어야 함을 의미합니다. 왜냐하면,

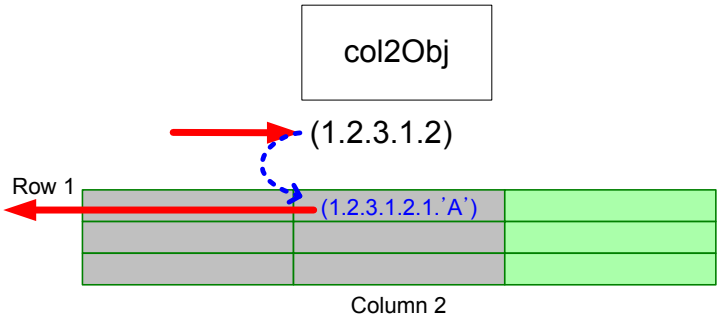
Instance id = 해당 column 의 columnar object id + 해당 row 의 INDEX

이기 때문이죠.

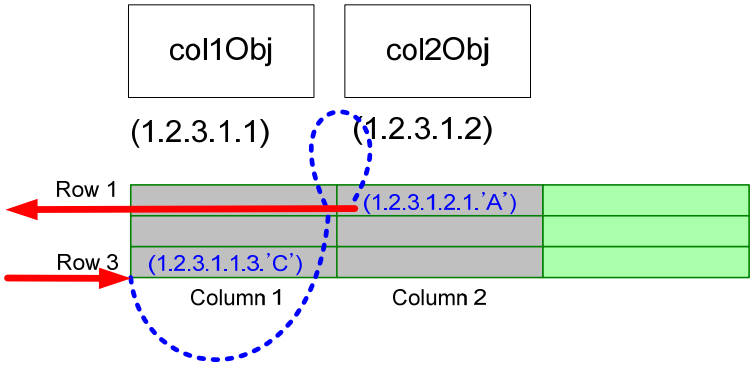
그럼 table 에서 GetNext request 를 했을 경우 그 결과가 어떻게 되는지 살펴 보죠 manager 가 agent 에게 'testTable' 의 object id (1.2.3)이나 'testEntry' 의 object id (1.2.3.1) 로 GetNext request 를 할 경우, agent 는 이 object id 에서 가장 근접한 instance id 인 (1.2.3.1.1.1.'A')와 instance value 인 1 을 manager 에게 되돌려 줍니다. 이는 row 와 column 이 (1,1)인 table 의 첫 번째 instance 입니다.



manager 가 agent 에게 columnar object 'col2Obj' 의 object id (1.2.3.1.2)로 GetNext request 를 할 경우, agent 는 이 object id 에서 가장 근접한 instance id 인 (1.2.3.1.2.1.'A')와 instance value 인 'A'를 manager 에게 되돌려 줍니다.



끝으로 manager 가 agent 에게 instance id (1.2.3.1.1.3.'C') 로 GetNext request 를 할 경우, agent 는 이 instance id 에서 가장 근접한 instance id 인 (1.2.3.1.2.1.'A')와 instance value 인 'A' 를 manager 에게 되돌려 줍니다.



결국, 위의 예제들에서 이야기 하고자 하는 핵심은 manager는 특정 instance id를 정확히 모르고서도 instance id 와 instance value에 접근할 수 있다는 점입니다.

그럼, 이제 우리가 궁금해 했던 처음 질문으로 돌아가 보죠.
6 장에서 언급했듯이 table 의 instance 를 얻기 위해서는 columnar object id 와 INDEX 가 필요합니다. 왜냐하면,

Instance id = 해당 column 의 columnar object id + 해당 row 의 INDEX

이기 때문이죠.
그러나, lexicographical ordering 을 잘 이용하면 manager 입장에서는 conceptual table object 의 object id 하나만 알고 있어도 GetNext 를 통해 table 의 모든 instance 를 얻을 수 있습니다. 또한, 특정 columnar object 의 object id 만 알아도 그 column 의 모든 instance 들을 얻을 수 있습니다.

9. SNMP 명령어

지금까지는 네트워크 관리에서 꼭 알아야 할 3 가지인 manager, agent, 그리고 MIB 에 대한 내용을 설명했습니다. 본 절에서는 이 3 가지가 SNMP 명령어와 어떻게 연결이 되어 있는지 알아보겠습니다.

우선 manager 와 agent 간에 어떤 SNMP 명령어가 존재하는지부터 살펴보도록 하죠

- Get : manager 가 agent 로부터 MIB 정보를 읽어옵니다
- Set : manager 가 agent 의 MIB 정보를 변경합니다
- Trap : agent 가 manager 에게 긴급한 정보를 알려 줍니다 (notification)

SNMP 는 이와 같은 명령어를 주고 받기 위해 다음과 같은 메시지 형태를 갖습니다. 이를 공식 용어로 PDU (Protocol Data Unit)라고 합니다. 아래는 메시지 종류와 그의 의미를 설명합니다

- GetRequest : manager 가 agent 에게 보내며 '내가 요청하는 instance id 의 instance 를 달라' 라는 의미입니다.
- GetNextRequest : manager 가 agent 에게 보내며 '내가 요청하는 object id 혹은 instance id 의 가장 근접한 instance 를 달라' 라는 의미입니다
- SetRequest : manager 가 agent 에게 보내며 '내가 요청하는 instance id 의 instance 를 변경하라' 라는 의미입니다
- Response : agent 가 manager 에게 응답으로 보내며 '너의 요청을 잘 수행했어' 혹은 '너의 요청을 수행하지 못했어' 라는 의미입니다
- Trap : agent 가 manager 에게 보내며 '지금 나한테 매우 중요한 일이 일어났어. 너도 꼭 알아야 해'라는 의미입니다

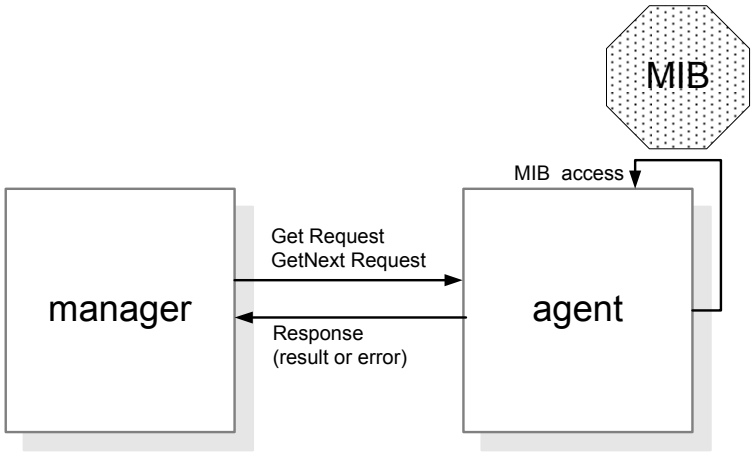
그럼, 각각의 명령어와 그에 따라 사용되는 SNMP 메시지 형태에 대해 알아보겠습니다.

먼저 get 명령어 입니다. 이 명령어는 manager 가 agent 에게 "내가 요청하는 instance 를 내게 다오" 라고 이야기 하는 것과 같습니다. Manager 입장에서는 다음의 2 가지 메시지 형태를 agent 에 전달합니다

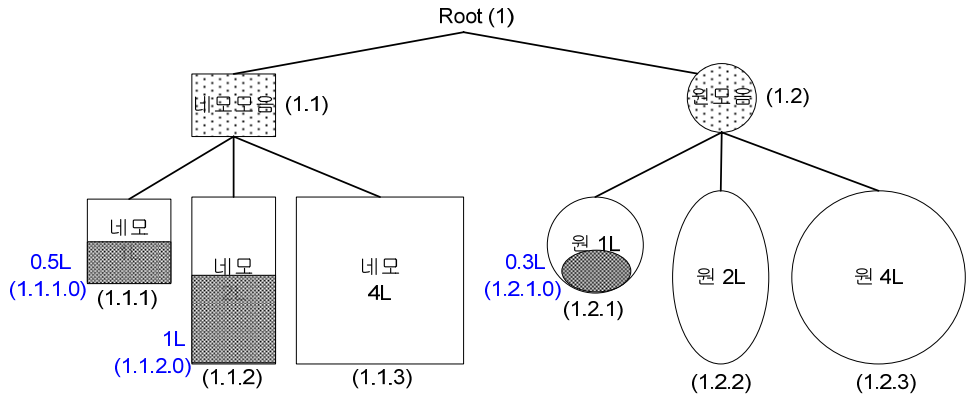
- GetRequest
- GetNextRequest

반면에 agent 는 manager 의 요청 결과를 'Response' 형태를 통해 manager 에게 전달합니다.

MIB 에 올바른 instance 가 존재할 경우 agent 는 'Response'를 통해 instance id 와 instance value 를 manager 에게 전달합니다. 반면 올바른 instance 가 존재하지 않을 경우 agent 는 'Response'를 통해 그에 상응하는 error 원인을 manager 에게 전달 합니다. 예를 들어, '네가 요청한 object 는 MIB 에 존재하지 않아' 라든지 '네가 요청한 object 는 존재하지만 instance 를 읽을 수 없어' 와 같은 이유들이죠.



그럼, 아래의 그림을 통해 예를 들도록 하겠습니다. 7 절의 예제와 함께 보시면 더욱 이해가 쉬우실 겁니다.



manager 가 instance id '(1.1.1.0)' 에 대해 GetRequest 를 할 경우, agent 는 instance id '(1.1.1.0)'의 instance 인 '0.5L' 를 MIB 로부터 얻습니다. 그런 후, 해당 instance id '(1.1.1.0)' 과 instance '0.5L' 를 Response 를 통해 manager 에 전달 합니다.

만일, manager 가 존재하지 않는 instance id '(1.1.4.0)'에 대해 Get Request 를 할 경우, agent 는 Get Response 를 통해 manager 에게 '네가 요청한 object 는 MIB 에 존재하

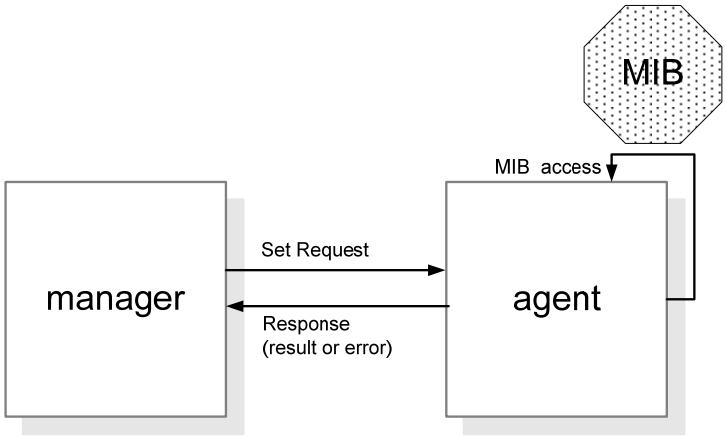
지 않아' 라는 error 원인을 알려줍니다.

manager 가 instance id '(1.1.1.0)' 에 대해 GetNextRequest 를 할 경우, agent 는 그 다음 instance id '(1.1.2.0)'의 instance 인 '1L'를 MIB 로부터 얻습니다. 그런 후, 해당 instance id '(1.1.2.0)'과 instance '1L'를 Response 를 통해 manager 에 전달 합니다.
만일, manager 가 존재하지 않는 instance id '(1.1.4.0)'에 대해 GetNext Request 를 할 경우, agent 는 가장 근접한 instance id '(1.2.1.0)'의 instance 인 '0.3L' 를 MIB 로부터 얻습니다. 그런 후, 해당 instance id '(1.2.1.0)'과 instance '0.3L' 를 Response 를 통해 manager 에 전달 합니다.

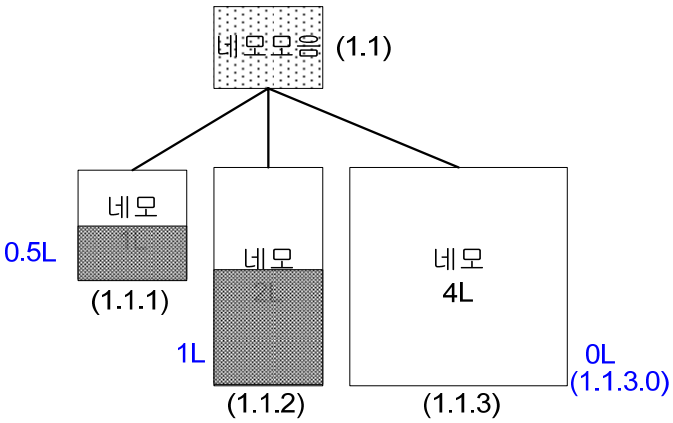
위의 예들을 표로 나타내면 다음과 같습니다.

Request	Response of Get Request	Response of GetNext Request
Object id (1.1.1)	**Error**	Instance id : (1.1.1.0) Instance value : 0.5L
Instance id (1.1.1.0)	Instance id : (1.1.1.0) Instance value : 0.5L	Instance id : (1.1.2.0) Instance value : 1L
Instance id (1.1.4.0) (MIB 에 존재하지 않음)	**Error**	Instance id : (1.2.1.0) Instance value : 0.3L

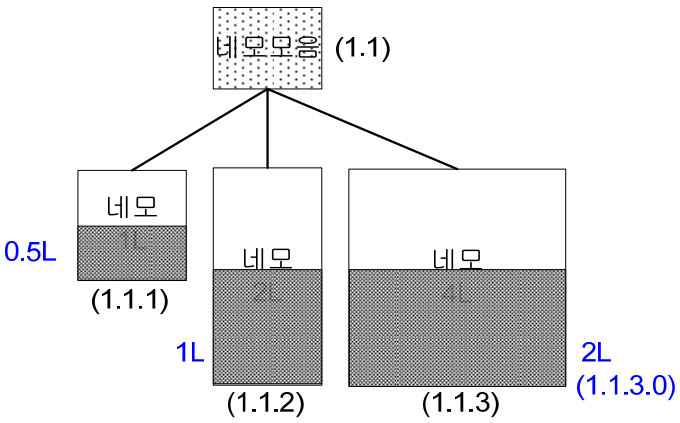
다음은 set 명령어 입니다.
이 명령어는 manager 가 agent 에게 "내가 요청한 instance 를 요청한 값으로 변경해 다오" 라고 이야기 하는 것과 같습니다. Manager 는 'SetRequest' 메시지 형태를 통해 agent 에게 요청합니다
마찬가지로 agent 는 요청 결과를 'Response' 메시지 형태를 통해 manager 에게 전달합니다.



이번에도 그림을 통해 예를 들도록 하겠습니다. 아래 그림을 보면 '네모 4L' object 의 instance 는 0L 입니다.

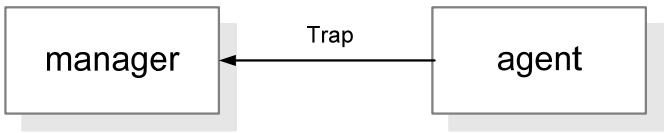


Manager 가 이 object 의 instance value 를 2L 로 바꿔달라고 agent 에게 요청을 하고, agent 가 이를 정상적으로 수행했다면 다음과 같이 변경되게 됩니다



Manager 는 이전에 설명한 get 명령어를 통해 현재 상태를 확인할 수 있습니다.

마지막으로 trap 명령어 입니다. 이 명령어는 이전에 설명된 명령어들과는 반대로 agent 가 manager 에게 보고를 합니다. 즉, agent 는 manager 에게 "지금 나한테 매우 중요한 일이 일어났어." 라고 이야기 하는 것과 같습니다. Agent 는 'Trap' 메시지 형태를 통해 manager 에게 보고 되며, manager 로부터의 응답은 없습니다.



Trap 은 규격에 정의된 것을 사용할 수도 있지만, 일반적으로는 agent 가 장비의 특성에 맞게 정의하여 사용합니다. 예를 들어, 장비가 특정한 이유로 재 시동 (rebooting) 되었다면, agent 는 규격에 정의된 'ColdStart' trap 을 발생하여 manager 에게 재 시동 사실을 알립니다. 반면에, '장비의 온도가 10 도 이하로 떨어지면 알리기'로 manager 와 agent 간에 약속 했다면, 온도가 10 도 이하로 떨어질 경우 agent 는 trap 을 발생하여 manager 에게 이 사실을 알립니다.

이로써 SNMP 개념에 대한 간단한 설명을 마치겠습니다.
더 깊은 지식이 필요하신 분들은 SNMP 관련 규격 및 관련 서적 (Appendix B 참조) 을 통해 더 많은 정보를 얻으실 수 있을 겁니다.

도움이 되셨기를 바라고, 끝까지 읽어주셔서 고맙습니다.

Appendix A. 문서정보

A.1 문서이력

■ 문서목적

SNMP 개념의 이해를 돕는다.

■ 작성자

김용석/ stone@novonetworks.com

■ 작성이력

최초작성 : 2010-04

Appendix B. References

SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 3rd Edition,
William Stallings, Addison Wesley

RFC1155
Structure and Identification of Management Information for TCP/IP-based Internets, May 1990

RFC 1157
A Simple Network Management Protocol (SNMP), May 1990

RFC 1213
Management Information Base for Network Management of TCP/IP-based Internets : MIB-II, March 1991

RFC 2578
Structure of Management Information Version 2 (SMIv2), April 1999