
System monitoring Documentation

Release 0.1

Youngmin Kim

Dec 08, 2018

1	서론	3
2	시리얼 케이블을 이용한 원격시스템 접속	5
3	원격시스템 BIOS(+GRUB) 설정	7
3.1	자동 power on	7
3.2	키보드 연결 없이 부팅하기	9
3.3	GRUB 설정 변경	9
4	원격시스템을 위한 무선이동통신	11
4.1	에그 타입	11
4.2	USB 타입	13
4.3	MINI PCI-E 타입	23
5	서버에서 원격시스템 접근	25
5.1	reverse ssh 터널 생성	25
5.2	자동 ssh 로그인	27
5.3	연결 유지용 스크립트	27
6	modbus	31
6.1	modbus 소개	31
6.2	libmodbus	36
6.3	pymodbus	38
7	로그 (syslog)	41
7.1	syslog 출력의 예	41
7.2	log level	42
7.3	셸 명령어로 syslog 출력하기	43
7.4	hostname 설정	43
7.5	C 코드에서 syslog 출력하기	44
7.6	logrotate로 최신 로그만 남기기	44
7.7	Centralized logging	46
8	nagios	53
8.1	nagios 설치	54
8.2	nagios core에 대한 간단한 소개	55
8.3	nagios 설정	61

8.4	email notification	64
9	Visualization	65
9.1	rrd	65
9.2	d3.js	65
9.3	jarmon	65

최초 작성일: 2013년 11월 25일

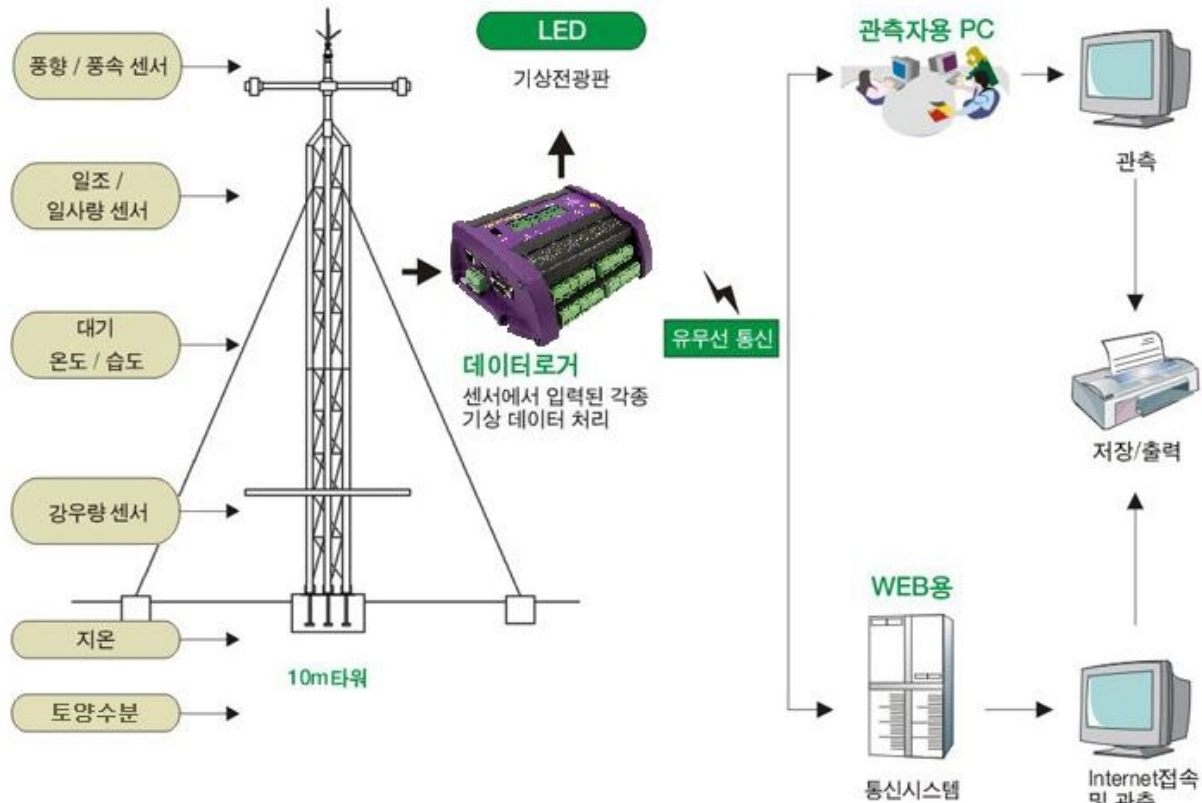
김영민 (https://github.com/ymkim92/system_monitoring)

목차:

서론

원격시스템 모니터링은 전자장비들의 수가 늘어나고 그들의 관리가 필요해지기 시작한 시점부터 수요가 발생하였다. 전자장비들의 크기는 작아지고, 가격은 낮아지고 있기 때문에 더 많은 곳에 설치되고 더 많은 데이터들을 수집하여 원격지에 위치한 서버로 전달한다.

기상 정보를 예로 들어 설명하자. 요즘도 군에서는 백엽상의 온도를 보고 손으로 이 값을 기록하는지는 모르겠으나, 적어도 필자가 근무하던 시절에는 그렇게 했었다. 하지만, 기상청에서는 좀 다르게 기상정보를 수집한다. 1986년 아시안 게임과 1988년 올림픽 게임에 필요한 기상정보를 제공하기 위해 경기장에 최초로 자동기상관측장비를 설치하였다. 하지만, 이 시점에는 우리가 생각하는 중앙 집중형 관리시스템이 존재하지는 않았다. 1996년에 들어서야 전국단위로 1분마다 정보를 수집하여 센터에서 통합정보를 볼 수 있는 기상 연속관리시스템이 구축되었다. (http://web.kma.go.kr/kma15/2004/contents/200403_05.htm) 요즘은 동단위로 예보를 하는 시대이니, 어디에 있는지는 잘 모르겠지만, 동마다 이런 관측장비가 다 설치되어 있다고 봐야 할 것이다. 아래 그림은 기상정보를 수집하는 과정이다.



기본적으로 네트워크에 연결되어 있을 수 밖에 없는 기지국, 라우터와 같은 통신 장비들은 snmp 프로토콜을 이용하여 초창기부터 원격관리가 가능하였다. snmp의 탄생적 특성상 주로 데이터의 입출력량을 모니터링할 수 있으며, UPS등의 전력시스템을 위한 전력 관련 정보들도 수집할 수 있다. NMS (Network Management System)는 snmp를 기본으로 통신장비들의 상황을 모니터링하고 제어할 수 있는 네트워크 통합 관리 시스템이다. 최근에 클라우드 서비스가 각광 받으면서 이 분야의 중요성도 부각되고 있다.

공장 자동화 분야에서도 본 글에서 논하고자하는 원격 시스템 모니터링이 사용된다. SCADA 시스템이라고 많이 불리는 이 시스템에서는 modbus, profibus 등의 단순한 프로토콜을 이용한다. 이 프로토콜들은 주로 근거리 통신을 위한 규격이지만, 원격으로 데이터를 전송할 때는 TCP/IP 상에 데이터를 실어 보내는 방법들을 제공한다.

이상에서 우리가 흔히 알고 있는 컴퓨터 급의 장비들을 관리하기 위한 snmp와 좀더 단순한 공장 자동화 장비들을 모니터링하고 제어하기 위한 modbus, profibus 프로토콜들이 사용되고 있는 것을 간단히 설명하였다. 본 글에서는 범용 컴퓨터 급이 아닌 센서 등의 단순한 컴퓨터들에서 수집되는 정보들을 어떻게 서버로 전달하고 이렇게 서버에 모아진 정보들을 운영자에게 보여줄 것인지를 설명하고자 한다. 특히 최근에 화두가 되고 있는 big data, Internet of Things, M2M 등에서도 이러한 내용들이 유용하게 활용될 수 있을 것으로 생각한다. 또한 본 글에서는 ubuntu 리눅스를 기반으로 한 소프트웨어적인 측면만을 다룬다.

이 글에서 언급하고자 하는 원격 시스템이란 유선통신 연결이 매우 어려운 지역에서 상용이동통신망을 이용하여 서버와 통신하는 장비를 의미한다. 원격시스템은 스스로 센서를 가지고 있을 수도 있으며, 근거리에서 시리얼 통신을 이용하여 데이터를 수집할 수 있다. 이렇게 수집한 데이터를 서버로 전송하고 서버에서 들어오는 제어신호를 처리할 수 있다. 원격 시스템은 ubuntu 리눅스를 OS로 사용하는 것을 가정한다.

시리얼 케이블을 이용한 원격시스템 접속

본 글에서 사용하는 원격시스템의 실제 하드웨어는 Jetway사에서 판매하는 통합 보드인 NF95A-270-LF을 이용한다. 국내에서는 [carpckorea](#)라는 업체에서 판매를 하고 있다.

Note: 이거 광고 아니고요.. 제가 carpckorea에서 물건을 좀 구매하긴 했지만, 사장님 절대 모릅니다. 사장님, 가격 너무 자주 그리고 많이 올리시는 거 아닙니까? 싸게 좀 팔아주세요~~

이 보드에 우분투를 설치할 때는 모니터와 키보드를 연결해서 하지만, 일단 설치가 끝난 경우에는 여러 대를 쌓아 놓고 시리얼로 필요한 장비에 연결하여 디버깅이나 업그레이드가 가능하다. 본 절에서는 이 방법에 대해 살펴본다.

먼저 PC와의 콘솔 연결을 위한 부팅 단계를 구분하고자 한다.

1. 부트로더로 제어권이 넘어가기 전단계 (BIOS 단계)
2. grub등의 부트로더가 시작되고 나서 로그인 프롬프트 전단계 (부팅 단계)
3. 사용자와의 인터랙션이 가능한 단계 (OS 단계)

Note: ()안의 단계 이름들은 뭐 정해진 건 아니고... 제가 그냥 붙여 봤습니다.

<http://mcchae.egloos.com/10562163>을 보면, 위 3단계별로 콘솔에서 출력을 받는 방법들이 설명되어 있다. 본 절에서는 사용자와의 인터랙션이 가능한 OS 단계에서의 출력만을 설명할 것이다.

ubuntu 6.10 이후부터 소개된 개념으로 **upstart**가 있다. 본 절에서는 upstart의 개념 자체가 중요한 것은 아니어서 자세히 설명하지는 않겠다. upstart는 부팅시에 구동되어야 하는 기능들을 기술하는 방법으로 과거 **init**가 하던 역할을 대체하고자 ubuntu를 만든 canonical에서 개발하고 밀고 있는 방법이다.

아무튼, 시리얼 콘솔 접속이 가능하도록 upstart의 스크립트를 `/etc/init/` 아래에 작성한다. 파일이름은 어떻게 해도 좋으나, 콘솔을 연결하기 위한 시리얼 포트의 이름을 붙인다. 예를 들어, `ttyS2 (COM3)`를 시리얼 콘솔 포트에 사용하려면 `/etc/init/ttyS2`를 생성하고 아래 내용을 넣는다.

```
start on runlevel [2345]
stop on runlevel [!2345]
```

(continues on next page)

(continued from previous page)

```
respawn
exec /sbin/getty -L 115200 ttyS2 vt102
```

화일을 저장한 후 `telinit 2` 명령을 수행하여 `runlevel`을 2로 해 주면 `ttyS2` 스크립트가 실행되면서 콘솔을 이용할 준비가 끝난다.

원격시스템의 `ttyS2`에 케이블을 연결한 후 콘솔에서 입출력이 가능한지를 확인하라. 콘솔의 `bps`는 115200으로 설정해야 하며 이 값은 `/etc/init/ttyS2` 에서 다른 값으로 변경하여 사용이 가능하다.

Note: 시리얼 콘솔이 정상적으로 동작하지 않는다면, 콘솔을 연결한 상태에서 다음 명령으로 데이터를 보내 케이블이나 시리얼 포트 자체가 정상동작하는지를 확인하라.

```
$ stty -F /dev/ttyS2 speed 115200 cs8 -cstopb -parenb
$ echo 'test' > /dev/ttyS2 # 콘솔로 test라는 문자열을 보낸다.
```

또는 `minicom` 을 설치하여 양방향 문자 전송이 가능한지 확인하는 것도 좋은 방법이다.

더 알아보기

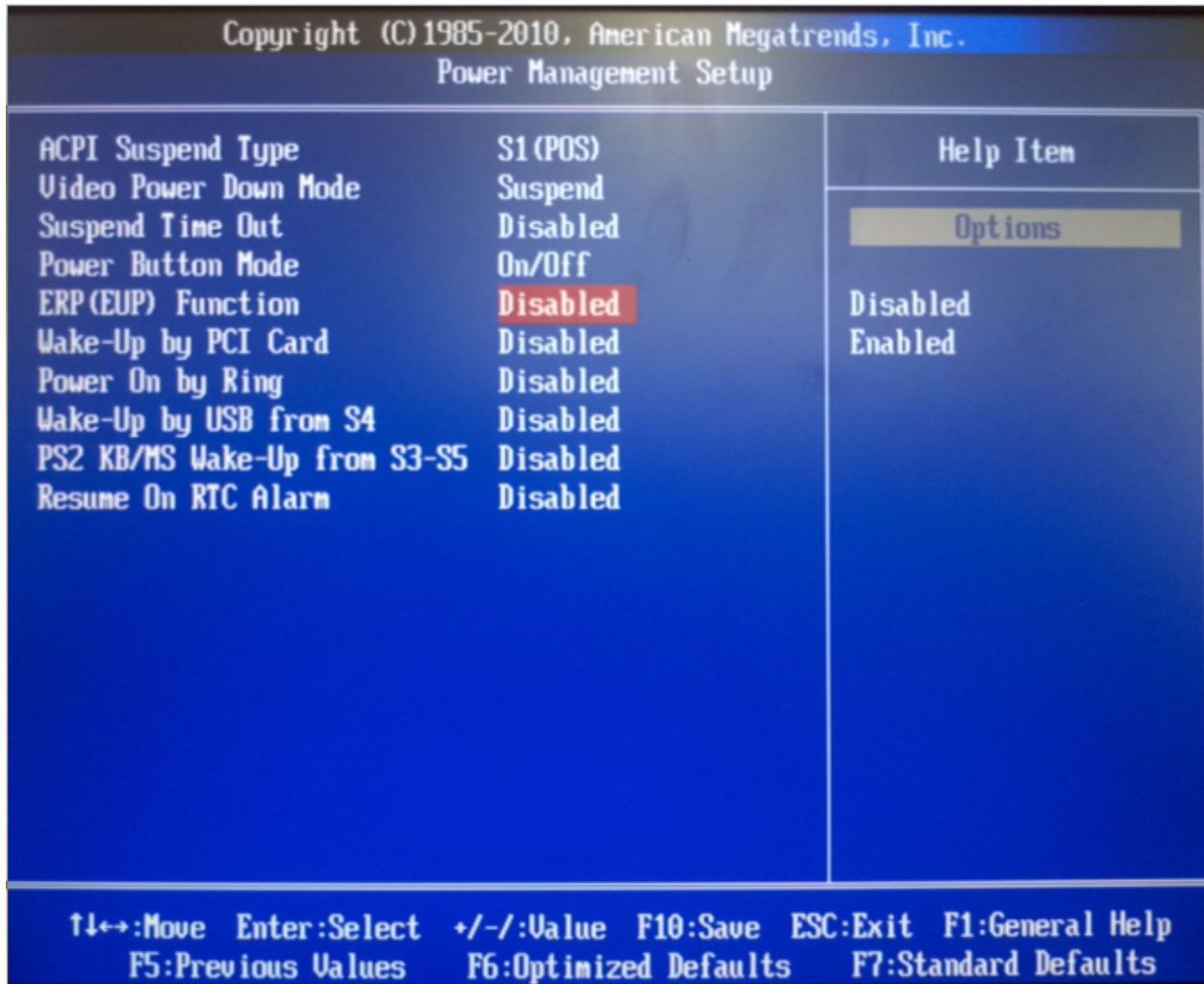
- <http://www.debuntu.org/how-to-set-up-a-serial-console-on-ubuntu/>
- <http://www.vanemery.com/Linux/Serial/serial-console.html>

원격시스템 BIOS(+GRUB) 설정

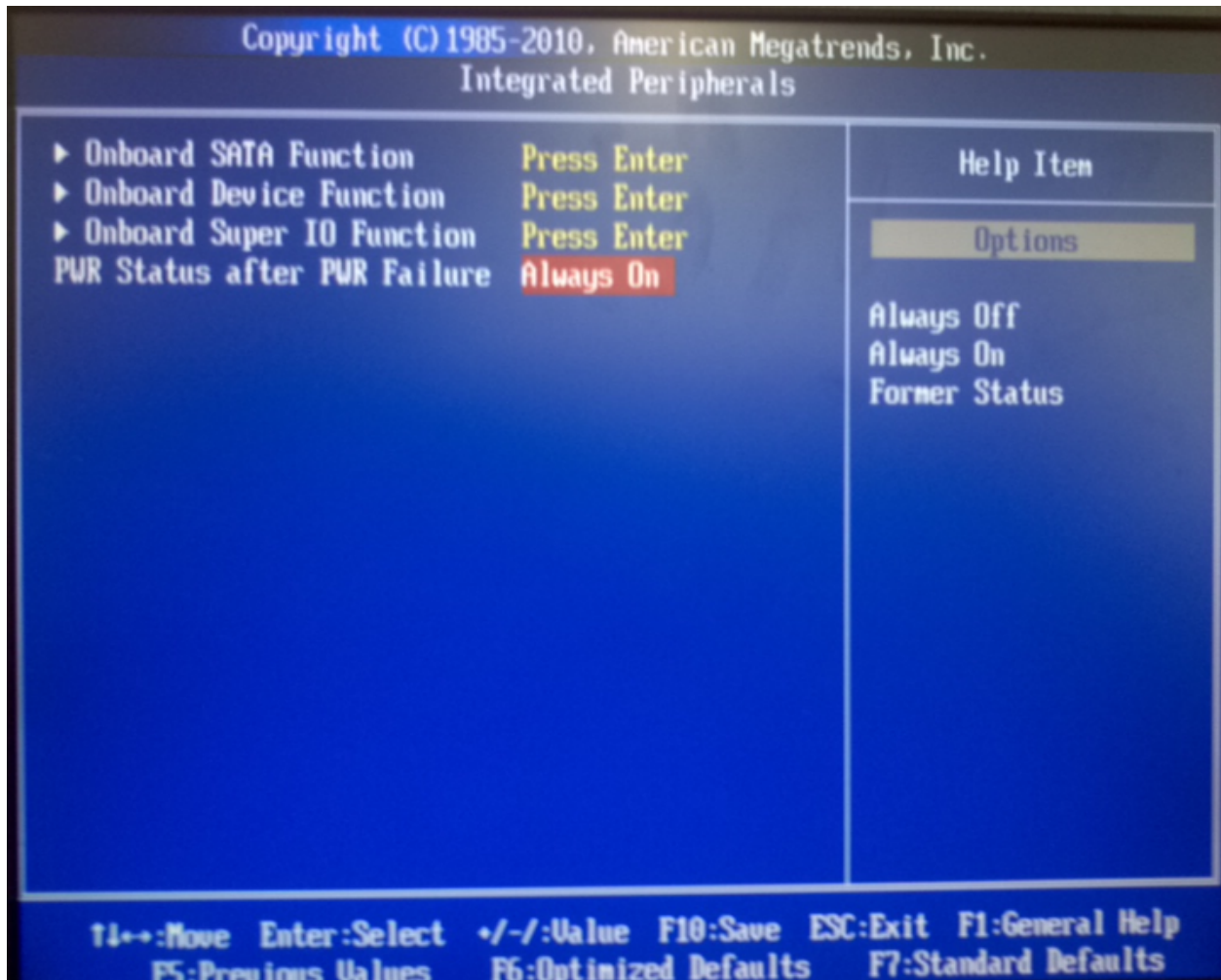
3.1 자동 power on

본 글에서 사용하는 원격시스템의 하드웨어인 NF95A-270-LF 은 꺼져있는 상태에서 전원이 공급될 경우 자동으로 켜지는 기능이 있다. 사람이 직접 접근하기 어려운 원격지에 설치된 장비에 전원이 나갔을 경우 전력이 복구된 후, 장비를 켜기 위해 사람이 투입된다면 엄청난 낭비일 것이다. 그렇기 때문에 전원만 공급되면 항상 켜지는 기능이 필요하다

CMOS에서 이를 설정할 수 있다. 먼저, ERP(EUP) Function 을 Disable 로 설정한 후



아래 그림과 같이 PWR Status after PWR Failure 을 Always On으로 설정하면 된다.



3.2 키보드 연결 없이 부팅하기

NF95A-270-LF 에서는 키보드 연결을 하지 않을 경우 부팅이 되지 않는다. 이를 방지하기 위해서 아래 설정을 해주어야 한다.

```
CMOS > Standard CMOS Features > Halt on: All errors
```

위에서 All errors 부분을 All but keyboard 로 변경한다.

3.3 GRUB 설정 변경

이 절은 CMOS에 관한 내용이 아니라 리눅스 부트로더인 GRUB에 대한 내용이다. GRUB은 부팅전에 사용자에게 어떤 OS로 부팅할 것인지, 다른 작업을 할 것인지를 선택할 수 있도록 한다. ubuntu에서는 부트로더에서 사용자 입력을 받아들이기 위해 일정시간을 대기하도록 한다. 리부팅 시간을 줄이기 위해 이 대기시간을 변경하는 방법을 알아본다.

또한, GRUB에서는 시스템 실패시 recordfail 이라는 변수에 true를 기록하고 다음 부팅시 이 값이 true이면 사용자의 입력을 받아야만 부팅이 진행되도록 하는 기능이 있다. 이 기능도 제거해야만 시스템에서 발생한 문제로 인해 다음 부팅이 불가능해지는 현상을 방지할 수 있다.

이상에서 언급한 내용들은 /boot/grub/grub.cfg 의 아래 내용 부분을 찾아 다음과 같이 수정해주면 해결할 수 있다. timeout 시간을 바꾸어 주는 것이 핵심이다.

```
if [ ${recordfail} = 1 ]; then
    set timeout=1
else
    set timeout=3
fi
```

Note: timeout 이 -1 로 되어 있으면, 자동으로 부팅이 되지 않고, 사용자의 입력이 있어야만 다음으로 진행된다.

이렇게 저장을 하고나면, 다음 번 부팅시 부터 변경된 grub 설정으로 부팅이 이루어진다.

원격시스템을 위한 무선이동통신

원격시스템은 유선전원 연결은 가능하지만, 유선통신은 불가능한 환경을 가정한다. 유선통신이 불가능하기 때문에 무선통신을 이용해야 한다. 요즘 흔히 이용할 수 있는 무선통신방법으로 무선랜과 상용이동통신망이 있다. 무선랜은 초기 설치 비용이 많이 들 수 있다. 무선랜 AP를 설치할 곳을 확보해야 하며, 인터넷 선도 끌어와야 한다. 이상한 사람들이 장비를 훼손시키지 못하도록 안전장비도 갖추어야 한다. 이런 일들은 예상외로 많은 비용이 든다. 이런 이유로 본 절에서는 상용이동통신망을 사용하는 것을 가정하고 설명할 것이다.

국내 이동통신 3사는 모두 데이터 통신이 가능한 모뎀과 서비스를 판매하고 있다. 모뎀은 크게 두 가지 형태로 나눌 수 있다. 하나는 KT에서 판매하는 에그(egg)로 유명한 독립장비로서의 모뎀이다. 이 장비는 로컬에서는 무선랜으로 클라이언트들의 접속을 허용하고 인터넷 연결을 위해서는 3G나 Wibro, LTE 망을 이용하는 형태이다. 이 제품의 특징은 하나의 장비로 여러 클라이언트들에게 서비스를 제공할 수 있다는 것이다. 또 다른 형태는 USB 타입의 모뎀이다. SKT에서 판매하는 T Login 이 대표적이며, KT에서도 iplug라는 이름으로 판매한다.

4.1 에그 타입

무선랜 AP에 접속하는 것과 동일하므로 이 절에서는 보안이 강화된 wpa 인증으로 설정된 AP에 접속하는 방법을 다루고자 한다.

무선랜(WiFi)를 이용하기 위해서는 무선랜 AP(공유기)에 접속해야 한다. 다른 사람이 사용하지 못하게 하거나, 나의 통신 내용을 다른 사람에게 보여주지 않기 위해 보안 기법을 사용한다. 예전에는 WEP라는 방법을 많이 사용했는데, 보안에 문제가 많다고 해서 요즘은 WPA/WPA2를 사용한다.

스마트폰이나 윈도우에서는 이 새로운 보안 기법을 사용하는데 큰 문제가 없어 보이는데... 리눅스에서는 조금까 다롭다고 생각된다. 물론 우분투에서도 GUI를 이용하면 WPA를 간단하게 이용할 수 있다. 하지만, GUI를 사용하지 않는 서버버전 같은 경우에는 명령어를 사용해서 WPA인증을 통과해야 하는데, 이 과정이 조금 복잡하다.

WPA인증을 하기 위한 리눅스 명령어는 wpa_supplicant 이다. 추가로 인증키를 만들기 위해서는 wpa_passphrase 명령어를 이용한다. 위 두 명령어만 있으면 WPA 인증을 이용할 수 있으며, 자세한 설명은 다음 링크에서 확인할 수 있다. (<https://help.ubuntu.com/community/WifiDocs/WPAHowTo>)

참고로 테스트 환경은 ubuntu 12.04 server 버전이다.

간단히 과정을 설명하면,

4.1.1 인증키 만들기

```
$ wpa_passphrase NetworkEssid TextPassphrase
# reading passphrase from stdin
TextPassphrase
network={
    ssid="NetworkEssid"
    #psk="TextPassphrase"
    psk=945609a382413e64d57daef00eb5fab3ae228716e1e440981c004bc61dccc98c
}
$ wpa_passphrase NetworkEssid TextPassphrase > wpa.conf
```

첫번째 명령어는 출력결과를 확인해 보기 위한 것이고, 두 번째 명령어는 이 결과를 현재 폴더에 wpa.conf라는 이름으로 저장한다. 이 화일은 wpa_supplicant 명령어의 input으로 사용되어 AP 접속에 사용된다. wpa_passphrase의 두 인자로 NetworkEssid와 TextPassphrase을 사용하며, networkEssid는 무선랜 AP 검색을 하면 나오는 이름이고, TextPassphrase는 AP에 설정한 접속 암호이다.

4.1.2 무선랜 AP에 연결하기

```
$ sudo wpa_supplicant -Dnl80211 -iwlan0 -cwpa.conf
[sudo] password for your_id:
Trying to authenticate with 00:07:ab:5d:f9:15 (SSID='NetworkEssid' freq=2452 MHz)
CTRL-EVENT-DISCONNECTED bssid=00:07:ab:5d:f9:15 reason=2
Trying to associate with 00:07:ab:5d:f9:15 (SSID='NetworkEssid' freq=2452 MHz)
Associated with 00:07:ab:5d:f9:15
WPA: Key negotiation completed with 00:07:ab:5d:f9:15 [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 00:07:ab:5d:f9:15 completed (auth) [id=0 id_str=]
```

위 예제는 연결이 성공한 경우이다. 마지막 줄에 연결되었다는 설명 (CTRL-EVENT-CONNECTED)이 나오고 더 이상의 추가적인 메시지 출력이 되지 않는다. 만약 문제가 있으면 출력문이 계속 나온다. -D 옵션은 드라이버를 정해 주는 것이고, -i 옵션은 무선랜 인터페이스 이름을 지정하는 것이며, -c 옵션에는 인증키 만들기에서 생성한 키화일을 적어주면 된다.

인터넷을 찾아보면 나오는 대부분의 사이트에서는 드라이버로 wext를 사용하면 된다고 나오지만, NF95A-270-LF에서는 wext로 무선랜 AP 접속이 실패한다. 자신의 시스템에서 사용할 수 있는 드라이버의 종류를 보는 방법은 아래 명령을 수행한 후, 중간쯤에 drivers라고 표시된 부분에 나와 있는 드라이버들을 모두 적용해 보는 것이다.

```
$ wpa_supplicant -h
...
This product includes software developed by the OpenSSL Project
for use in the OpenSSL Toolkit (http://www.openssl.org/)

usage:
  wpa_supplicant [-BddhKLqqstuvW] [-P] [-g] \
    -i -c [-C] [-D] [-p] \
    [-b] [-f] \
    [-o] [-O] \
    [-N -i -c [-C] [-D] \
    [-p] [-b] ...]

drivers:
  wext = Linux wireless extensions (generic)
  nl80211 = Linux nl80211/cfg80211
  wired = Wired Ethernet driver

options:
```

(continues on next page)

(continued from previous page)

```
-b = optional bridge interface name
-B = run daemon in the background
-c = Configuration file
-C = ctrl_interface parameter (only used if -c is not)
...
```

이 문제로 고민하는 분들에게 도움이 되길 기원한다.ㅋㅋ
드라이버가 맞지 않으면 다음과 같은 에러가 나온다.

```
"WPA: 4-Way Handshake failed"
```

4.1.3 부팅시 자동으로 WPA인증하기

생성한 wpa.conf파일을 /etc 아래에 저장하시고 /etc/network/interfaces 파일을 약간 수정한다.

```
$sudo cp wpa.conf /etc
$sudo vi /etc/network/interfaces
```

파일 내용에서 wlan0쪽을 아래와 같이 수정

```
auto wlan0
iface wlan0 inet dhcp
    wpa-driver nl80211
    wpa-conf /etc/wpa.conf
```

4.2 USB 타입

실험에 사용한 ubuntu 버전은 11.10이지만, 그 이후의 버전들에서 잘 동작할 것으로 생각한다. 본 절에서 사용할 USB 타입 모델은 2012년 초에 구입하였으며, 모델명은 아래와 같다.

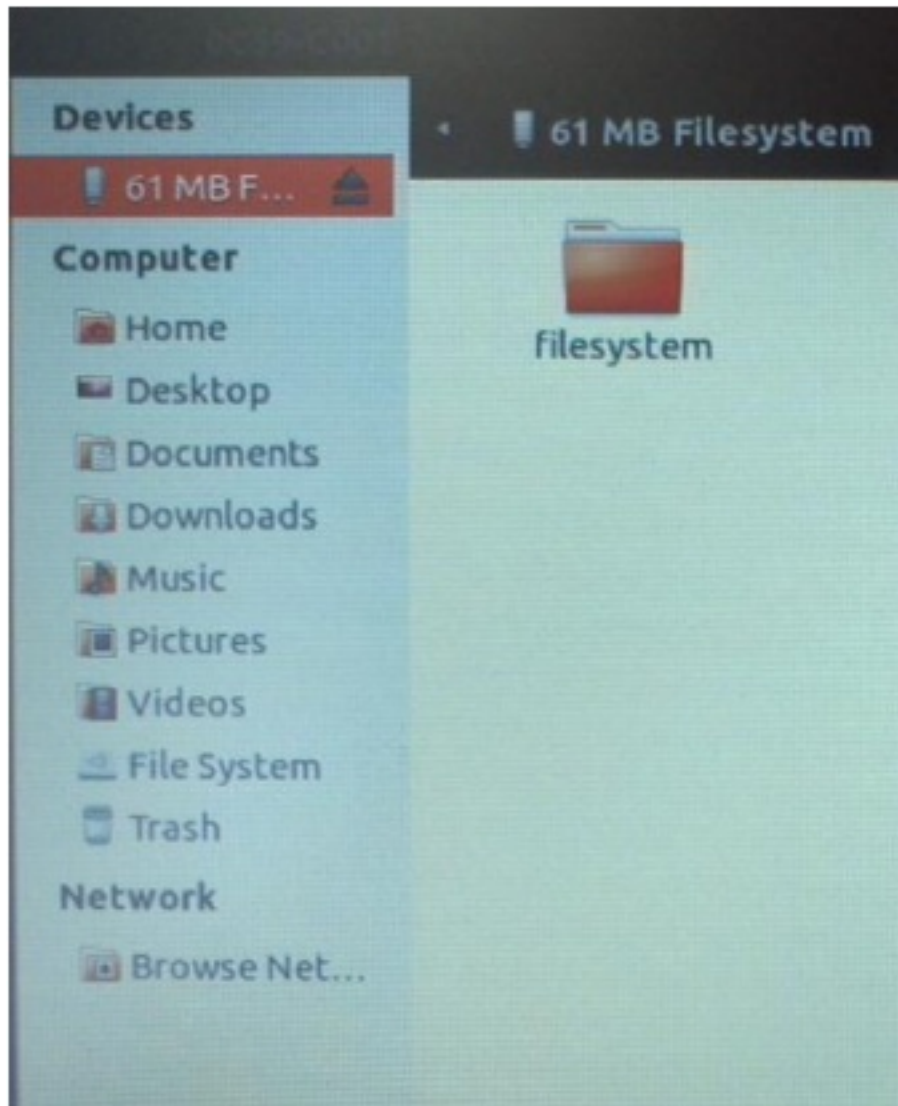
- SKT T-Login 듀얼 CBU-450D
- KT iplug CHU-629K

Warning: 통신 모뎀은 스마트폰에 비해 구입하기가 훨씬 까다로웠다. 대리점 방문전에 전화로 확인해 보거나 인터넷으로 구매처를 확인해 보라.

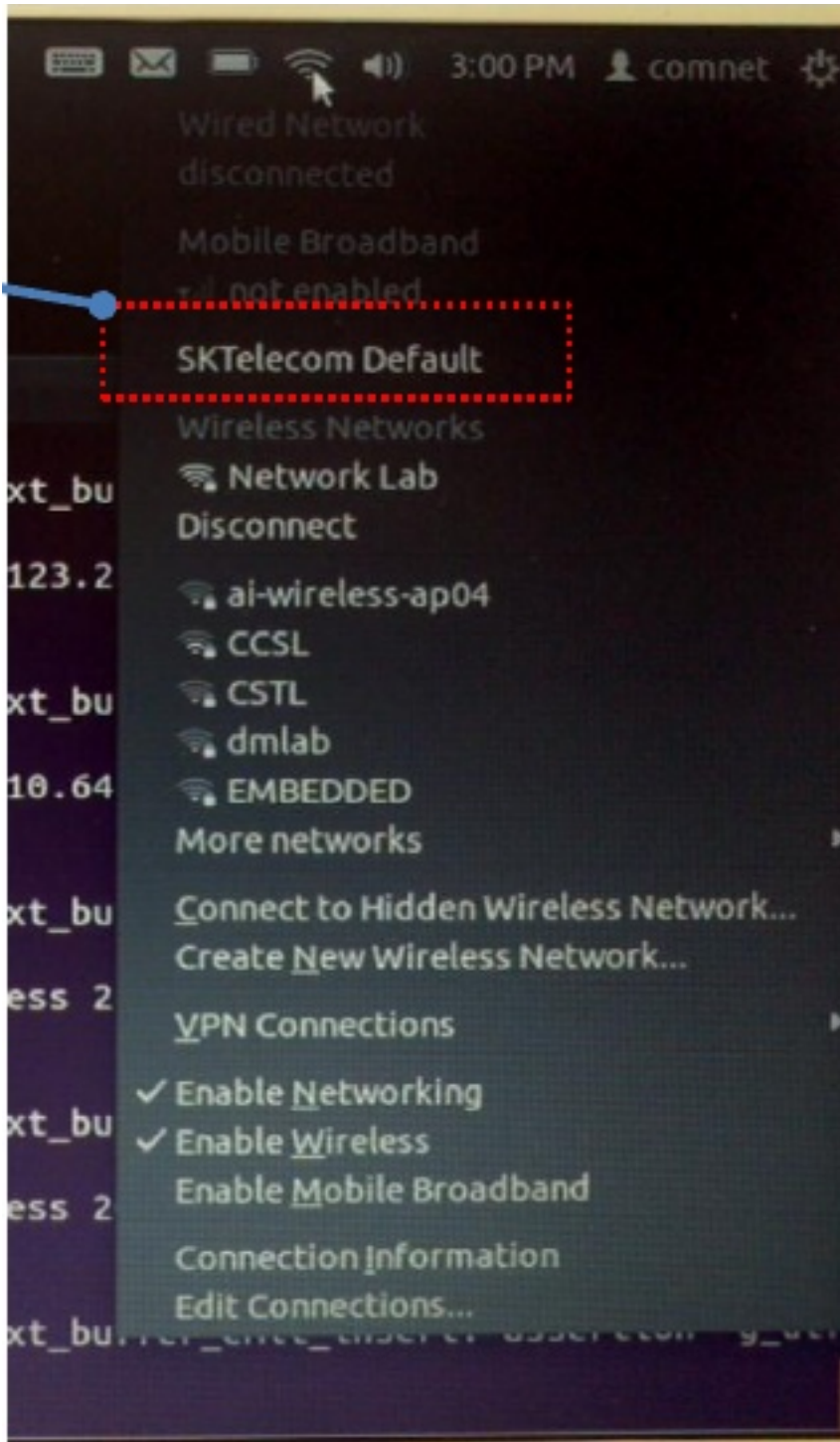
4.2.1 SKT T-Login 인식 시키기

SKT T-Login은 KT 모뎀에 비해 쉽게 설치가 가능하다. 3가지 방법으로 사용이 가능하며, 첫번째 방법은 ubuntu에서 기본적으로 제공하는 네트워크 연결 프로그램을 이용하는 것이다. 두번째 방법은 gnome-ppp 라는 프로그램을 이용하는 방법이며, 마지막으로 wvdial을 이용하는 방법이 있다. 첫 두가지 방법은 반드시 GUI 환경이 필요한 방법이며, wvdial은 명령어와 옵션만으로 실행이 가능하다.

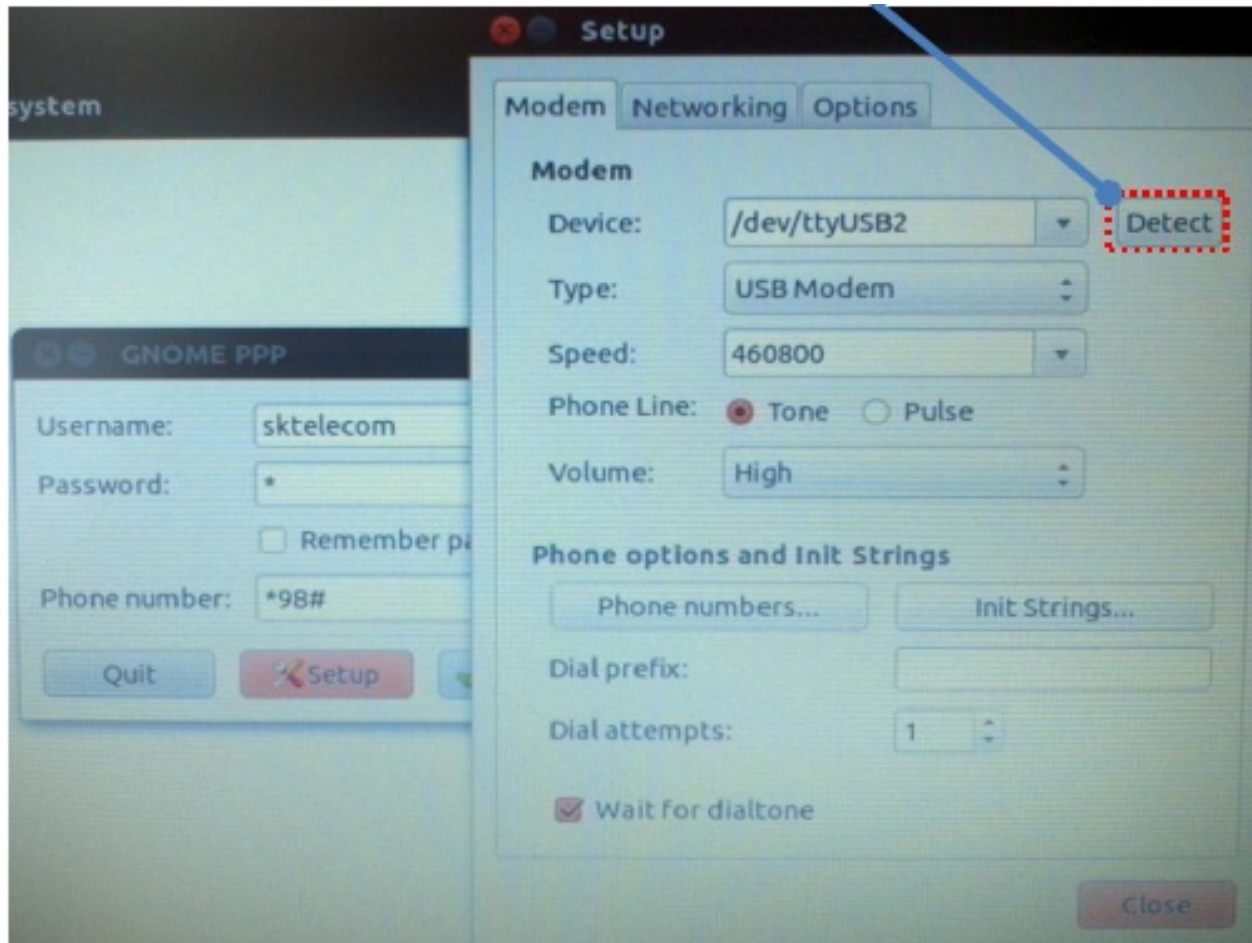
ubuntu에서는 유무선랜 연결을 관리하기 위한 프로그램을 내장하고 있으며, GUI환경에서 우상단의 트레이 아이콘을 눌러 조작할 수 있다. USB 모뎀을 연결하고 2분정도를 기다리면, Natilus 에서 다음과 같이 USB 저장장치가 잡힌 것을 확인할 수 있다. usb_modeswitch 라는 프로그램이 이 일을 자동으로 처리해 준다. 즉 usb_modeswitch는 USB 타입 모뎀을 USB 저장장치로 인식한 후 통신 기능을 활성화 시킨다. 먼저 USB 저장장치로 인식된 상태를 다음 창으로 확인할 수 있다.



이후 통신 기능이 활성화 되고 나면, 통신 아이콘을 눌렀을 때 아래와 같이 sk-telecom이라고 써지며, 이를 누르면 통신이 연결된다.



두번째 방법인 `gnome-ppp`는 별도로 설치를 해야 하는 GUI 프로그램이며, 아래에서 “detect” 버튼을 누르면 `/dev/ttyUSB2`와 같은 디바이스로 통신 모뎀이 잡힌다(참고).



마지막으로 wvdial을 이용하는 방법이다. 이 명령어는 gnompp가 수행하는 기능을 콘솔에서 수행가능하도록 한다(gnome-ppp를 설치하면 wvdial도 설치된다). 설정화일을 직접 작성하거나 아래 방법을 이용해 자동으로 생성할 수 있다. 이 명령을 수행하면 /etc/wvdial.conf 에 설정화일이 저장된다. gnome-ppp에서 detect 과정과 매칭된다고 할 수 있다.

```
$ sudo wvdialconf
```

아래는 이렇게 자동생성된 설정화일을 약간 수정한 내용이다.

```
[Dialer Defaults]
init1 = ATZ
init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
init3 = AT&F
ISDN = 0
Modem Type = USB Modem
New PPPD = yes
Phone = *98#
Modem = /dev/ttyUSB2
Username = sktelecom
Password = 1
Baud = 38400
```

다음, 아래 명령으로 T-Login을 네트워크에 연결한다.


```
$ sudo wvdial
```

사실 위에서 소개한 세가지 방법은 모두 `usb_modeswitch`에서 수행해 놓은 일 위에 숨가락만 얻는 사용자 인터페이스라고 할 수 있다. 연결된 USB 통신 모뎀을 인식하고 사용할 준비를 하는 것은 `usb_modeswitch`이다.

본 글에서 언급하는 원격시스템은 모든 일을 자동화 하여 스스로 구동할 수 있어야 하므로 마지막 방법인 `wvdial`을 사용한다.

Tip: 설정된 모뎀은 `ppp0` 라는 이름의 장치로 설정되었는데, 이 장치로 통신이 되는지를 검사해 보기 위해서는 `eth0`나 `wlan0` 등의 기존 통신 인터페이스를 꺼야 한다. 아래 명령으로 이를 수행할 수 있다.

```
$ sudo iwconfig wlan0 txpower off
$ sudo ifconfig eth0 down
```

4.2.2 KT iplug 인식 시키기

KT 모뎀을 인식시키기 위해서는 `usb_modeswitch`에 대한 지식이 필요하다. 먼저 `iplug`를 PC에 연결하고 인식과정이 어디까지 진행되었는지 살펴봐야 한다. `dmesg` 나 `lsusb -v` 명령을 이용하여 확인이 가능하며, 아래는 장치를 연결한 후 `dmesg`의 실행결과이다. 장치를 CD ROM으로 까지는 인식하였으나, 아직 시리얼 통신 장치로는 인식하지 못한 상태이다.

```
comnet@comnet-N100: ~
[ 607.556165] usb 1-1: new high speed USB device number 7 using ehci_hcd
[ 607.694847] scsi6 : usb-storage 1-1:1.0
[ 608.693559] scsi 6:0:0:0: CD-ROM          CMOTECH  Mass Storage          2.3 PQ
: 0 ANSI: 2
[ 608.705167] sr0: scsi-1 drive
[ 608.705770] sr 6:0:0:0: Attached scsi CD-ROM sr0
[ 608.706315] sr 6:0:0:0: Attached scsi generic sg1 type 5
[ 609.073526] ISO 9660 Extensions: Microsoft Joliet Level 3
[ 609.139894] ISOFS: changing to secondary root
[ 609.729410] cfg80211: Found new beacon on frequency: 2472 MHz (Ch 13) on phy0
[ 668.371113] usb 1-1: USB disconnect, device number 7
comnet@comnet-N100:~$ dmesg | tail
[ 609.073526] ISO 9660 Extensions: Microsoft Joliet Level 3
[ 609.139894] ISOFS: changing to secondary root
[ 609.729410] cfg80211: Found new beacon on frequency: 2472 MHz (Ch 13) on phy0
[ 668.371113] usb 1-1: USB disconnect, device number 7
[ 681.100178] usb 1-1: new high speed USB device number 8 using ehci_hcd
[ 681.239368] scsi7 : usb-storage 1-1:1.0
[ 682.241672] scsi 7:0:0:0: CD-ROM          CMOTECH  Mass Storage          2.3 PQ
: 0 ANSI: 2
[ 682.293192] sr0: scsi-1 drive
[ 682.293770] sr 7:0:0:0: Attached scsi CD-ROM sr0
[ 682.294255] sr 7:0:0:0: Attached scsi generic sg1 type 5
comnet@comnet-N100:~$
```

다음은 `iplug`를 연결하기 전과 후의 `/sys/bus/usb/devices` 안의 화일을 `ls`로 표시한 것이다. 1-1과 1-1:1.0이 연결 후 추가된 것을 확인할 수 있다.

`/sys/bus/usb/devices` 폴더는 `ubuntu`에서 인식한 USB 장치들의 상세정보를 볼 수 있는 곳이다. `usb1 ~ usb5`까지를 root hub라고 부르며, 숫자는 bus number이다. 즉 이 장비에는 총 5개의 USB 버스가 장착되어 있음을 알 수 있다. N-0:1.0은 root hub의 인터페이스를 나타내는 특별한 장치이다. 즉 아무런 장치도 연결되어 있지 않더라도 이 폴더는 존재한다. 이 예에서는, 1번 버스의 8번 포트에 장치가 하나 연결되어 있던 상태에서 USB 모뎀을 연결하자, 1번 포트에 인식된 상황이다.

```
comnet@comnet-N100:/sys/bus/usb/devices$ ls
1-0:1.0 1-8:1.0 2-0:1.0 4-0:1.0 usb1 usb3 usb5
1-8 1-8:1.1 3-0:1.0 5-0:1.0 usb2 usb4
comnet@comnet-N100:/sys/bus/usb/devices$ ls
1-0:1.0 1-1:1.0 1-8:1.0 2-0:1.0 4-0:1.0 usb1 usb3 usb5
1-1 1-8 1-8:1.1 3-0:1.0 5-0:1.0 usb2 usb4
comnet@comnet-N100:/sys/bus/usb/devices$
```

Note: 리눅스에서 usb 장치를 인식하는 인식하여 화일 시스템에 표시하는 자세한 내용은 <http://www.linux-usb.org/FAQ.html> 에서 /sys/bus/usb/devices 로 검색하면 찾을 수 있다.

이렇게 인식이 된 상태에서 `lsusb -v` 을 실행하면, 다음 결과를 볼 수 있다. idVendor와 idProduct 값을 주의 깊게 보아야 한다. idVendor는 0x16d8 이며, idProduct는 7003으로 되어 있다.

```
comnet@comnet-N100: ~
Device Status:      0x0000
(Bus Powered)

Bus 001 Device 010: ID 16d8:7003 CMOTECH Co., Ltd.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB                  2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor                0x16d8 CMOTECH Co., Ltd.
  idProduct              0x7003
  bcdDevice               0.00
  iManufacturer          3 CMOTECH CO., LTD.
  iProduct               2 CMOTECH CDMA Technologies
  iSerial                4 1234567890ABCDEF
  bNumConfigurations     1
Configuration Descriptor:
  bLength                9
  bDescriptorType         2
  wTotalLength           32
  bNumInterfaces         1
```

usb_modeswitch에서는 이 두 값을 이용하여 자동으로 장치를 인식하나, 위 설정값은 usb_modeswitch에 등록되어 있는 값이 아니다. 이를 등록하기 위해서는 두 가지 일을 해 주어야 한다. 첫째로, /lib/udev/rules.d/40-usb_modeswitch.rules 을 편집기로 열고나서 16d8로 검색하여 700a의 내용을 복사하여 7003을 추가로 만들어야 한다.

두번째로, /usr/share/usb_modeswitch 로 이동하면 configPack.tar.gz 화일이 존재하며, 이 화일을 풀어 16d8:700a를 찾고 이를 복사하여 16d8:7003을 생성한다.

```
$ cd /usr/share/usb_modeswitch
$ sudo tar zxvf configPack.tar.gz
$ sudo cp 16d8:700a 16d8:7003
```

이제, USB 모뎀을 제거한 후 다시 연결하면 장치를 인식할 것이다.


```

# C-motech CHU-629K
ATTRS{idVendor}=="16d8", ATTRS{idProduct}=="7003", RUN+="usb_modeswitch '%b/%k'"

# C-motech CHU-629S
ATTRS{idVendor}=="16d8", ATTRS{idProduct}=="700a", RUN+="usb_modeswitch '%b/%k'"

# C-motech CHU-629S (Variant)
ATTRS{idVendor}=="16d8", ATTRS{idProduct}=="700b", RUN+="usb_modeswitch '%b/%k'"

```

```

comnet@comnet-N100: /usr/share/usb_modeswitch
TargetClass=0xff
CheckSuccess=20
MessageContent="55534243123456782400000080000dfe524445564348473d4e444953000000"
comnet@comnet-N100:/usr/share/usb_modeswitch$ cat 16d8:700a
#####
# C-motech CHU-629S

DefaultVendor= 0x16d8
DefaultProduct=0x700a

TargetClass=0xff
CheckSuccess=20
MessageContent="55534243123456782400000080000dfe524445564348473d4e444953000000"
comnet@comnet-N100:/usr/share/usb_modeswitch$ cat 16d8:7003
#####
# C-motech CHU-629K

DefaultVendor= 0x16d8
DefaultProduct=0x7003

TargetClass=0xff
CheckSuccess=20
MessageContent="55534243123456782400000080000dfe524445564348473d4e444953000000"
comnet@comnet-N100:/usr/share/usb_modeswitch$

```

4.2.3 SKT와 KT 모뎀 동시에 인식 시키기

개별 모뎀별로 인식에 성공했다면, /etc 아래에 wvdial1.conf와 wvdial2.conf를 아래와 같이 만든다. ttyUSB#에서 번호만 해당 장치에 맞게 변경해 주면 된다.

```
comnet@comnet-N100: ~
      RX bytes:130 (130.0 B)  TX bytes:181 (181.0 B)

comnet@comnet-N100:~$ cat /etc/wvdial1.conf

[Dialer Defaults]
init1 = ATZ
init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
init3 = AT&F
Modem Type = Analog Modem
ISDN = 0
New PPPD = yes
Phone = *98#
Modem = /dev/ttyUSB2
Username = sktelecom
Password = 1
Baud = 921600
comnet@comnet-N100:~$ cat /etc/wvdial2.conf

[Dialer Defaults]
init1 = ATZ
init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
init3 = AT&F
Modem Type = Analog Modem
ISDN = 0
New PPPD = yes
Phone = *98#
Modem = /dev/ttyUSB6
Username = sktelecom
Password = 1
Baud = 921600
comnet@comnet-N100:~$
```

아래 두 명령으로 두 장치를 모두 인터넷에 연결 시킬 수 있다.

```
$ sudo wvdial -C /etc/wvdial1.conf
$ sudo wvdial -C /etc/wvdial2.conf
```

아래는 두 장치가 모두 연결된 상태를 보여준다.

ifconfig 명령으로 ppp0와 ppp1으로 각각 잡힌 것을 확인할 수 있다.


```

Terminal
comnet@comnet-N100: /etc
--> Modem initialized.
--> Sending: ATDT*98#
--> Waiting for carrier.
ATDT*98#
CONNECT 384
--> Carrier detected. Waiting
--> Don't know what to do! St
--> Starting pppd at Mon Apr 1
--> Pid of pppd: 5272
--> Using interface ppp0
--> pppd: [08][1b][04] <#[04]
--> pppd: [08][1b][04] <#[04]
--> pppd: [08][1b][04] <#[04]
--> pppd: [08][1b][04] <#[04]
--> pppd: [08][1b][04] <#[04]
--> local IP address 123.228.
--> pppd: [08][1b][04] <#[04]
--> remote IP address 10.64.64
--> pppd: [08][1b][04] <#[04]
--> primary DNS address 203.
--> pppd: [08][1b][04] <#[04]
--> secondary DNS address 211.
--> pppd: [08][1b][04] <#[04]
modpoll3.1

comnet@comnet-N100: /etc
--> Sending: ATDT*98#
--> Waiting for carrier.
ATDT*98#
CONNECT 7168
--> Carrier detected. Waiting for prompt.
--> Don't know what to do! Starting pppd and hoping for the best.
--> Starting pppd at Mon Apr 16 13:43:00 2012
--> Pid of pppd: 5299
--> Using interface ppp1
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> local IP address 10.133.25.101
--> pppd: [08] <+[03]! 8<
--> remote IP address 10.64.64.65
--> pppd: [08] <+[03]! 8<
--> pppd: [08] <+[03]! 8<
--> primary DNS address 211.246.100.20
--> pppd: [08] <+[03]! 8<
--> secondary DNS address 211.219.86.1
--> pppd: [08] <+[03]! 8<

```

```

comnet@comnet-N100: ~
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

ppp0 Link encap:Point-to-Point Protocol
inet addr:123.228.208.74 P-t-P:10.64.64.64 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:8 errors:3 dropped:0 overruns:0 frame:0
TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:182 (182.0 B) TX bytes:221 (221.0 B)

ppp1 Link encap:Point-to-Point Protocol
inet addr:10.133.25.101 P-t-P:10.64.64.65 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:7 errors:2 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:130 (130.0 B) TX bytes:181 (181.0 B)

comnet@comnet-N100:~$

```

좌하단은 KT 모뎀이고 우하단은 SKT 모뎀이다.



4.2.4 인터페이스 이중화

윗 절에서 두 개의 모뎀을 장치로서 인식시키는 지점까지는 성공하였다. 하지만, 나중에 인식된 모뎀에서 라우팅 테이블을 선점하여 설정 (default route) 하기 때문에 일반적인 통신에서는 나중에 인식된 모뎀만을 이용하여 통신이 이루어진다. 본 절에서는 두 인터페이스를 모두 이용하여 통신할 수 있는 방법을 찾을 수 있는 곳만을 소개한다.

- <http://kindlund.wordpress.com/2007/11/19/configuring-multiple-default-routes-in-linux/>
- <http://www.rjssystems.nl/en/2100-adv-routing.php>

두 개의 인터페이스를 동시에 사용하여 유선 네트워크에 비해 신뢰성이 떨어지는 무선망의 특성을 상쇄시키려는 노력으로 이중화를 시도하였다. 그러나, 아래와 같은 문제들을 아직 해결하지 못하여 이중화를 비중있게 다루지 않는다.

- /dev/ttyUSB# 을 잡을 때 순서가 바뀌는 경우가 있거나, USB 모뎀을 제외한 다른 장치가 연결되어 있을 경우에는 번호가 바뀌는 경우도 발생, ttyUSB 번호를 고정시키는 방법 필요

- 장치인식이 안되는 경우 발생

4.2.5 부팅시에 자동으로 인식시키기

/etc/rc.local 에 아래 코드를 추가하여 부팅시점에 자동으로 USB 모뎀을 인식시킬 수 있다.

```
#!/bin/sh -e
exec 2> /tmp/rc.local.debug
set -x

wvdial -C /etc/wvdial.conf &
```

2,3 줄을 추가하여 wvdial에서 출력되는 내용들을 /tmp/rc.local.debug 에 저장할 수 있다.

Note: 부팅후에 USB 모뎀을 꽂고 wvdial을 실행시키면 잘 인식이 되지만, 스크립트를 적용시키고 USB 모뎀을 꽂아 놓은 상태에서 리부팅시 모뎀을 인식하지 못할 때는 http://www.draisberghof.de/usb_modeswitch/bb/viewtopic.php?t=794&sid=74e7064df361371e772312c1813b1c20 에서 “Josh”님의 글을 참고하라. 원인 치료가 아닌 증상 치료로 상황을 넘여가기 위해서는 “Josh”님이 제시한 아래 방법을 이용하라. 다음 코드를 /etc/rc.local에 넣으면 된다.

```
modprobe -v option
echo "16d8 700b" > /sys/bus/usb-serial/drivers/option1/new_id
echo "16d8 7003" > /sys/bus/usb-serial/drivers/option1/new_id
```

4.3 MINI PCI-E 타입

최근에 잘나가고 있는 국내 업체중에 **모다정보통신** 이라는 곳이 있다. 이 곳에서는 이 글에서 다루고 있는 원격 모니터링 시스템의 통신 부분에 대한 모듈을 개발/판매하는 회사이다. **TTA** 를 통해 모니터링 시스템용 통신 모듈에 대한 표준화에도 앞장서고 있는 회사이다. 관련자료를 [여기](#) 에서 다운로드 할 수 있다.

분명한 사실은 몇몇 노트북에서 현재시점(2013년 12월)에 MINI PCI-E 타입의 WWAN 모뎀을 이용하여 이동통신망의 데이터 통신을 이용할 수 있으나, 아직까지는 표준화가 되어 있지 않아 무선랜 모듈과 같이 범용적인 모뎀은 존재하지 않는다. 하지만, 원격지 모니터링 시스템의 통신 모듈의 궁극적인 지향점은 MINI PCI-E 타입과 같이 소형의 모듈을 표준화하여 컴퓨터 내에 탈착할 수 있으며 이동통신망을 이용하여 통신하는 것이라고 생각한다.

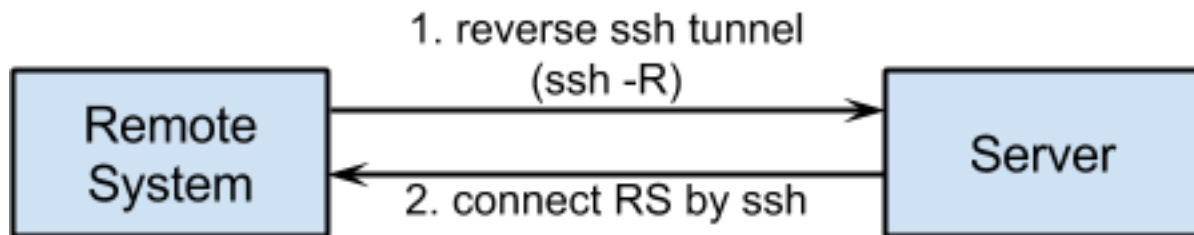
Note: 이 밖에 관련 업체로 <http://www.telit.com/> , <http://opengear.com/> 등이 있다.

이렇게 무선이동통신을 이용할 수 있으면 원격시스템에서 서버로의 접속이 가능해 진다. 그러나, 서버에서 원격시스템을 접근하려면 어떻게 해야 할까? 다음 절에서 이에 대해 알아보자.

서버에서 원격시스템 접근

원격시스템들은 원격지에 분산되어 있기 때문에 시리얼 콘솔을 이용해서 장비에 접근하는 것이 번거롭고 시간을 많이 소비하게 된다. 때문에 원격에서 시스템에 접근할 수 있는 방법을 제공하여야만 하며 이 경우, 디버깅과 업데이트가 용이해진다.

하지만, 원격시스템은 상용이동통신망을 이용하기 때문에 외부에서 직접 접근이 불가능하다. 이런 경우에 접근을 할 수 있는 방법으로 reverse ssh 터널 이 있다. 먼저 원격시스템에서 서버로 “ssh -R” 옵션을 사용하여 역방향 터널을 만든다. 이렇게 하면, 원격시스템의 특정 포트에서 ssh 접속을 대기하도록 설정한다.



5.1 reverse ssh 터널 생성

ssh는 보통 ssh 서버에 접속해 원격에서 서버를 관리할 목적으로 많이 사용된다. 클라이언트와 서버 사이에 교환되는 패킷들을 암호화하기 때문에 telnet보다 훨씬 안전하여 많이 사용되는 명령어이다. 하지만, ssh는 telnet처럼 단순한 기능만을 제공하는 것은 아니다. 본 절에서는 ssh -R 옵션을 통해 remote port forwarding을 사용하는 방법을 기술하고자 한다.

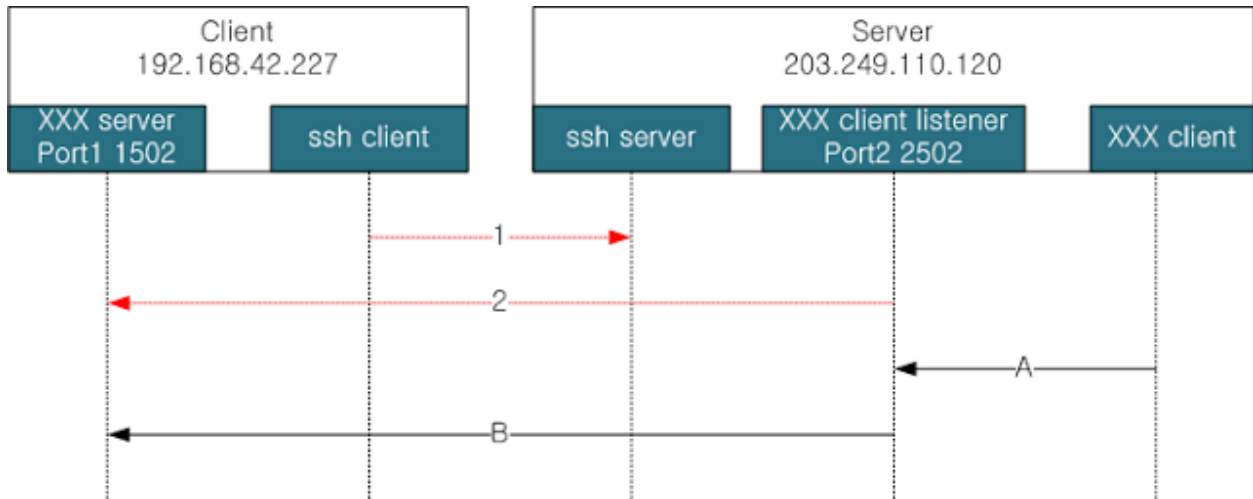
ssh 터널을 만드는 일반적인 방법에 대해서는 아래 사이트를 참고하라.

- http://www.hanb.co.kr/network/view.html?bi_id=547

두 host A와 B가 있을때 A는 공인IP를 가지고 있지만, B는 사설IP만을 가지고 있다고 할 경우, 일반적으로 B에서 A로만 접근이 가능하다. 하지만, reverse 터널을 생성할 경우 A에서 B로의 접근이 가능해 진다. 예를 들어 설명하면, 여러분 가정에 사설IP만을 가지고 있는 리눅스 장비가 있을 경우, 외부에서 집에 있는 리눅스로 접속하여 관리를 할 수 있다.

-R 옵션의 명령어 형태는 아래와 같다. 이 명령은 아래 그림의 client에서 수행하는 것이다.

```
$ ssh -R <port1>:<host_of_port2>:<port2> <id@ssh_server>
예) $ ssh -R 2502:localhost:1502 your_id@203.249.110.120
또는 $ ssh -R 2502:192.168.42.227:1502 your_id@203.249.110.120
```



위 그림과 명령어를 연관지어 설명하도록 하겠다.

1. 먼저 client에서는 server에 ssh로 접속할 수 있어야 한다. 즉, client에서 `ssh id@203.249.110.120` 으로 접속이 가능해야 한다. 그림에서 화살표 1에 해당한다.
2. 접속이 가능하다면 client에서 위 예의 "ssh -R 2502:localhost:1502 your_id@203.249.110.120" 명령어를 입력하여 그림에서 화살표 2에 해당하는 reverse 터널(remote port forwarding)을 생성한다.
3. XXX client에서 2502 포트로 접속하면, (그림에서 화살표 A에 해당)
4. reverse 터널을 통해 XXX server에 접속할 수 있다.

위의 과정을 netcat 명령어로 간단하게 검증해 보도록 하자.

1. Server에서 ssh server를 구동한다.
2. Client에서 아래 명령을 실행한다.

```
$ ssh -R 2502:localhost:1502 your_id@203.249.110.120
위 명령을 수행하면 ssh로 server에 접속한 것과 증상적으로 동일한 접속이 이루어진다.
다른 창을 열어 아래 명령을 수행하라.

$ nc -l -p 1502
위 명령으로 XXX server를 port 1502에 띄워놓는다.
```

3. Server에서 2502 port로 접속한다. 명령 수행후 아무 글자를 쳐보면 client에 동일하게 출력되어 있는 것을 확인할 수 있다.

```
$ nc localhost 2502
```

4. Server가 아닌 다른 컴퓨터에서 Sever의 2502 port로 접속하길 원한다면 아래 명령을 사용하라.

```
$ nc 203.249.110.120 2502
```

이상에서 일반적인 용도의 reverse ssh 터널에 대해 살펴보았다. 본 글에서는 원격시스템의 제어를 위해 ssh 접속을 해야 하므로 아래 명령어를 사용하여 터널을 연다. 포트번호 40122는 얼마든지 변경해서 사용할 수 있다.

```
$ ssh -fN -R 40122:localhost:22 id@server_ip_or_domain_name
-f:   백그라운드로 실행될 수 있도록 프로세스를 fork
-N:   remote 명령어를 실행하지 않음
-fN:  원격지에 로그인하지 않고 백그라운드로 터널만 생성
```

5.2 자동 ssh 로그인

윗 절에서 reverse 터널을 만드는 법을 배웠다. 시스템을 리부팅할 때마다 터널을 만들기 위해 비밀번호를 입력하는 번거로움을 감수할 관리자는 없으리라 생각한다. 이러한 이유로 본 절에서는 비밀번호 입력없이 ssh 서버에 자동으로 로그인 하는 방법을 살펴본다.

host A에서 host B의 user b로 로그인하는 것을 가정하고 아래와 같은 명령을 수행하면 A에서 B로 ssh 접속을 할 때 비밀번호를 입력하지 않아도 된다. 아래는 모두 host A에서 수행할 일들이다.

1. 인증키 생성

```
$ ssh-keygen -t rsa
-t : 생성할 Key의 type (rsa1, dsa, ecdsa, rsa(for protocol 2))
물어보는 내용에 대해 전부 엔터만 입력하면 된다.
```

2. host B에 .ssh 폴더 생성

```
$ ssh user_b@host_B mkdir -p .ssh
```

3. 인증키 복사

```
$ cat .ssh/id_rsa.pub | ssh user_b@host_B 'cat >> .ssh/authorized_keys'
```

Note: 2, 3의 과정을 한번에 수행하기

```
$ ssh-copy-id user_b@host_B
```

Note: 루트권한으로 수행하는 프로세스내에서 원격지로 ssh 접속을 시도할 경우는 루트 계정에서 위의 작업들을 수행해 주어야 한다. 즉 host A에서 어느 계정으로 위의 과정을 수행하는가도 중요하다. 루트 계정에서 자동 로그인을 하기 위해서는 아래 명령어를 먼저 수행한 후 위 과정을 실행하라.

```
$ sudo su -
```

5.3 연결 유지용 스크립트

자동으로 로그인을 할 수 있으니, 이제 ssh 터널 생성 명령어를 자동화하고 연결이 끊어졌는지를 검사하여 재연결을 시도하는 방법을 알아보자. 아래 스크립트는 인터넷에서 찾을 수 있는 코드들인

- http://www.brandanhutchinson.com/ssh_tunneling.html
- <http://blog.kxr.me/2013/02/reverse-ssh-tunnel-manager-remote-ssh.html>

를 약간 수정한 것이다.

```
#!/bin/sh

# $REMOTE_PORT is the remote port number that will be used to tunnel
# back to this system
REMOTE_PORT=${PORT_PREFIX}

# $REMOTE_HOST is the name of the remote system
REMOTE_HOST=${YOUR_HOST_NAME}

for PORT in "22" "80"
do
    # $COMMAND is the command used to create the reverse ssh tunnel
    #COMMAND="autossh -f -N -R *:$REMOTE_PORT:localhost:22 $REMOTE_HOST"
    COMMAND="ssh -fN -R ${REMOTE_PORT}${PORT}:localhost:${PORT} $REMOTE_HOST"

    # Is the tunnel up? Perform two tests:

    # 1. Check for relevant process ($COMMAND)
    pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND

    # 2. Test tunnel by looking at "netstat" output on $REMOTE_HOST
    ssh $REMOTE_HOST netstat -an | egrep "tcp.*:${REMOTE_PORT}${PORT}.*LISTEN" \
    > /dev/null 2>&1
    if [ $? -ne 0 ] ; then
        pkill -f -x "$COMMAND"
        $COMMAND
    fi
done
```

위 코드는 원격시스템의 ssh(22)와 http(80) 포트에 접속할 수 있도록 두 개의 reverse 터널을 생성한다. REMOTE_PORT 변수에 설정되는 값은 최종 포트번호가 아니며, for 문에 있는 22와 80과 합쳐져서 완성된 포트를 생성한다. 즉 \${REMOTE_PORT}\${PORT} 이 최종 포트번호이다. COMMAND 변수에 할당되는 ssh 구문을 보면 어떻게 포트가 할당되는지 이해할 수 있을 것이다.

pgrep을 이용해 명령어가 수행되고 있는지 검사하여 프로세스가 없을 경우 명령어를 수행한다. 그러므로 이 스크립트를 crontab에 등록하여 수행할 경우 cron에 의해 스크립트를 시작하기 전까지는 터널이 생성되지 않는다.

TCP 통신에서는 단순히 터널을 설정하는 프로세스가 살아있다고 해서 TCP 세션이 활성화 되어 있다는 것을 보장하지 않는다. 이는 telnet이나 ssh 접속을 한 후에 오랜동안 입출력이 없으면 연결이 끊어지는 경우가 발생하는 것으로 알 수 있다. telnet server와 telnet client는 모두 동작하고 있지만, 연결을 끊어져 있을 수 있다는 것이다.

이를 방지하기 위해서 주기적으로 클라이언트에서 서버측으로 패킷을 보내는 방법을 이용하며, ssh 자체적으로 해결하는 방법과 범용적인 방법이 있다.

ssh 에서는 설정을 통해 연결유지용 패킷을 자동으로 발생시킬 수 있다. <http://www.maketecheasier.com/keep-ssh-connections-alive-in-linux> 를 보면 해당 방법을 살펴볼 수 있다. 또는, 무식한 방법이긴 하지만, 확실한 방법이기도 한 ssh 로 원격지에 접속하여 명령어를 수행하는 방법도 있다. ssh에서는 명령어 한 줄로 이를 수행할 수 있다. 예를 들어 ssh user@host ls 를 실행하면 원격지의 ls 결과를 출력한다.

범용적으로 사용할 수 있는 방법으로 TCP KEEPALIVE 를 이용할 수 있다.

본 절에서는 두번째로 소개한 방법을 이용하여 연결을 유지하며, 추가적으로 연결이 실제로 유지되어 있는지를 검사하기도 한다. 일석이조라 할 수 있겠다. ㅋㅋ 명령어 ssh \$REMOTE_HOST netstat -an | egrep "tcp.*:\${REMOTE_PORT}\${PORT}.*LISTEN" > /dev/null 2>&1 은 원격지에 접속하여 netstat문으로 reverse ssh 터널이 살아있는지를 검사하는 것이다. 이 명령어 한 줄로 클라이언트에서 서버측으로 데이터를 보내는 효과도 있으며, 서버측에서 연결이 유효한지도 검사한다.

ssh 터널을 설정하여 서버에서 클라이언트로 접속이 가능해지면, 어떠한 형태의 모니터링도 가능해진다. 하지만, 모니터링을 자동화 하기 위해서는 몇 가지 기술들을 더 배워야 한다. 원격시스템의 정보를 서버로 전송하기 위한

modbus 프로토콜, 로그 저장 및 전송 등이다. 계속 읽어보시라~~

이제 원격시스템은 이동통신망을 이용하여 네트워크에 연결되었고 ssh 터널을 통해 서버에서 원격시스템을 제어할 수 있으므로, 원격시스템을 원격지에 설치해 두어도 된다. 하지만, 아직 모니터링 기능을 자동적으로 수행할 수는 없다. 본 절에서는 정보 수집과 제어를 자동화 하기 위해 modbus 프로토콜에 대해 알아본다.

원격시스템에서 근거리의 센서들로부터 정보를 수집하기 위해 시리얼통신에 기반한 modbus를 이용하고, 모니터링한 내용을 서버로 전송하고 서버에서 원격시스템을 제어하기 위해서는 modbus TCP를 이용한다. 실제 환경에서 원격시스템을 사용하기 위해서는 본 절에서 소개한 내용을 기반으로 소스 코드를 수정하여야 한다.

6.1 modbus 소개

본 절은 wikipedia의 [modbus](#) 를 번역하면서 약간 살을 붙인 것이다.

modbus는 1997년, 지금은 Schneider Electric인 Modicon 이라는 회사에서 만든 시리얼 통신 프로토콜이다. 제조공장이나 놀이공원의 기계들을 자동화하고 제어하는 목적으로 사용되는 Programmable Logic Controller(PLC)들과의 통신에 사용할 목적으로 만들어졌다. 프로토콜이 단순하지만, 장비 제어와 모니터링에 필요한 기능들을 수행할 수 있기에 사실상의 표준 프로토콜의 지위를 얻게 되었고, 현재까지 산업용 전자 장치들을 서로 연결하는 목적으로 널리 사용된다. 다음은 modbus가 산업용으로 널리 사용되는 이유들이다.

- 산업용 통신 프로토콜로 개발됨
- 프로토콜이 공개되어 있고 공짜
- 설치와 유지보수가 용이
- 비트단위 또는 워드(16bits) 단위로 정보조작이 용이

modbus는 약 240개의 장비들을 서로 연결할 수 있다. 예를 들면, 온도와 습도를 측정하는 여러 장비들이 모니터링 서버로 현재 상태를 보고하도록 할 수 있다. 일반적으로 서버에서 센싱 장비들에게 질의를 보내고 장비들은 이에 대해 응답하는 형태로 동작한다. Supervisory control and data acquisition (SCADA) 시스템에서도 모니터링 서버와 remote terminal unit (RTU)을 연결하기 위해 modbus를 자주 사용한다.

Schneider Electric에서는 modbus를 공개하여, 2004년 4월 이후로 프로토콜의 개발과 수정을 Modbus Organization에서 수행한다.

6.1.1 통신과 장비

modbus는 master/slave 기반 프로토콜이다. 시리얼 통신에서는 master로 설정된 장비만이 slave로 정보를 요청할 수 있는 반면, 이더넷 통신에서는 네트워크상의 어떤 노드도 정보를 요청할 수 있다. 요청정보는 읽기와 쓰기 모두 가능하다. 하지만, 대부분의 경우 master는 하나만 존재한다.

네트워크상에 연결된 모든 노드들이 요청을 받을 수는 있지만, 요청정보에 들어있는 목적주소 장비만이 이에 응답한다. 물론 목적지 주소가 브로드캐스트 주소일 때는 예외이다. 목적지 주소를 0으로 설정하면, 수신한 모든 노드에서 요청을 처리하지만, 응답은 보내지 않는다. 또한, modbus 요청 정보에는 통신 오류를 검출하기 위한 코드를 포함한다.

modbus를 지원하는 모뎀과 게이트웨이들이 많이 존재한다. 주로 serial-to-IP 제품들로 master측으로는 WiFi나 이 동통신등 무선통신을 지원하기도 한다.

6.1.2 프레임 포맷

통신 네트워크의 종류와 요청 정보의 형식에 따라 여러가지 프레임 포맷이 존재한다. Modbus RTU는 시리얼 통신망을 이용하며, 프레임에서는 이진정보를 사용한다. Modbus Ascii도 시리얼 통신망을 이용하지만, 프레임에 ascii 문자를 넣어 통신한다. 즉 0을 넣으면, 실제 프레임에는 0의 ascii 코드인 48이 한 바이트를 차지하게 된다.

Table 1: Modbus RTU Frame Format

Name	Length (bits)	Function
Start	28	at least 3 1/2 character times of silence (mark condition)
Address	8	Station Address
Function	8	Indicates the function code, eg read coils / inputs
Data	n * 8	Data + length will be filled depending on the message type
CRC	16	Error checks
End	28	at least 3 1/2 character times of silence between frames

Table 2: Modbus ASCII Frame Format

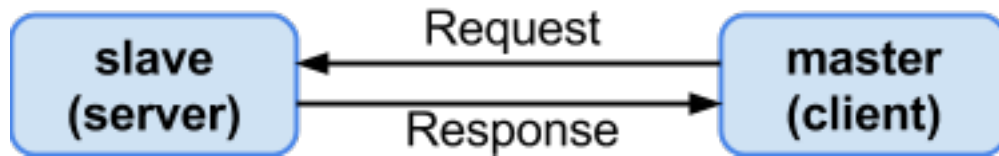
Name	Length (char.)	Function
Start	1	starts with colon (:) (ASCII hex value is 0x3A)
Address	2	Station Address
Function	2	Indicates the function codes like read coils / inputs
Data	n	Data + length will be filled depending on the message type
LRC	2	Error checks
End	2	carriage return – line feed(CR/LF) pair (ASCII values of 0x0D & 0x0A)

Table 3: Modbus TCP Frame Format

Name	Length (bytes)	Function
Transaction Identifier	2	For synchronization between messages of server & client
Protocol Identifier	2	Zero for Modbus/TCP
Length Field	2	Number of remaining bytes in this frame
Unit Identifier	1	Slave Address (255 if not used)
Function code	1	Function codes as in other variants
Data bytes	n	Data as response or commands

Modbus/TCP의 unit identifier는 modbus 게이트웨이 내부의 RTU 장비들을 지칭할 목적으로 사용되며, 이런 경우가 아니라면 IP 주소를 사용해 목적지를 인식할 수 있으므로 사용되지 않는다. 바이트 순서는 빅 엔디언을 사용한다. 즉 첫번째 바이트가 MSB이다.

프레임 포맷의 Function code와 data부분을 특별히 Protocol Data Unit (PDU)라고 부르며, 위 프레임 포맷 표에서 Name 부분에 푸른색으로 표시된 부분이 PDU이다. PDU는 Request PDU, Response PDU, Exception Response PDU의 3가지 종류가 정의되어 있으며, 아래는 요청(Request)과 응답(Response)의 과정을 해당 주체와 함께 간단히 표시한 그림이다.



요청을 받은 slave에서 해당 메시지를 해석할 수 없거나, 명시된 주소를 찾을 수 없는 등의 오류 발생시에는 요청의 function code에 0x80을 더한 값을 오류 코드로 설정하여 Exception Response로 응답한다.





6.1.3 지원되는 기능 코드들

읽기와 쓰기를 비롯한 다른 몇몇 기능들이 아래 표에 정리되어 있다. 프로토콜이 만들어진 초창기부터 있었던 기능들은 굵은 글자로 표시되어 있으며, 1~6까지의 기능 코드를 갖는다.

Function type	.	.	Function name	Function code
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	2
.	.	Internal Bits or Physical Coils	Read Coils	1
.	.	.	Write Single Coil	5
.	.	.	Write Multiple Coils	15
.	16-bit access	Physical Input Registers	Read Input Register	4
.	.	Internal Registers or Physical Output Registers	Read Holding Registers	3
.	.	.	Write Single Register	6
.	.	.	Write Multiple Registers	16
.	.	.	Read/Write Multiple Registers	23
.	.	.	Mask Write Register	22
.	.	.	Read FIFO Queue	24
.	File Record Access	.	Read File Record	20
.	.	.	Write File Record	21
Diagnostics	.	.	Read Exception Status	7
.	.	.	Diagnostic	8
.	.	.	Get Com Event Counter	11
.	.	.	Get Com Event Log	12
.	.	.	Report Slave ID	17
.	.	.	Read Device Identification	43
Other	.	.	Encapsulated Interface Transport	43

modbus는 센서나 LED등을 모니터링하고 제어하기 위한 특수한 data type들을 제공한다. 다음 표에 data type들을 정리하였다. 이 표는 <http://jamod.sourceforge.net/kbase/protocol.html> 에서 가져온 것이다. java 로 구현한 modbus 에 대한 문서이며, 볼 만한 내용들이 있다.

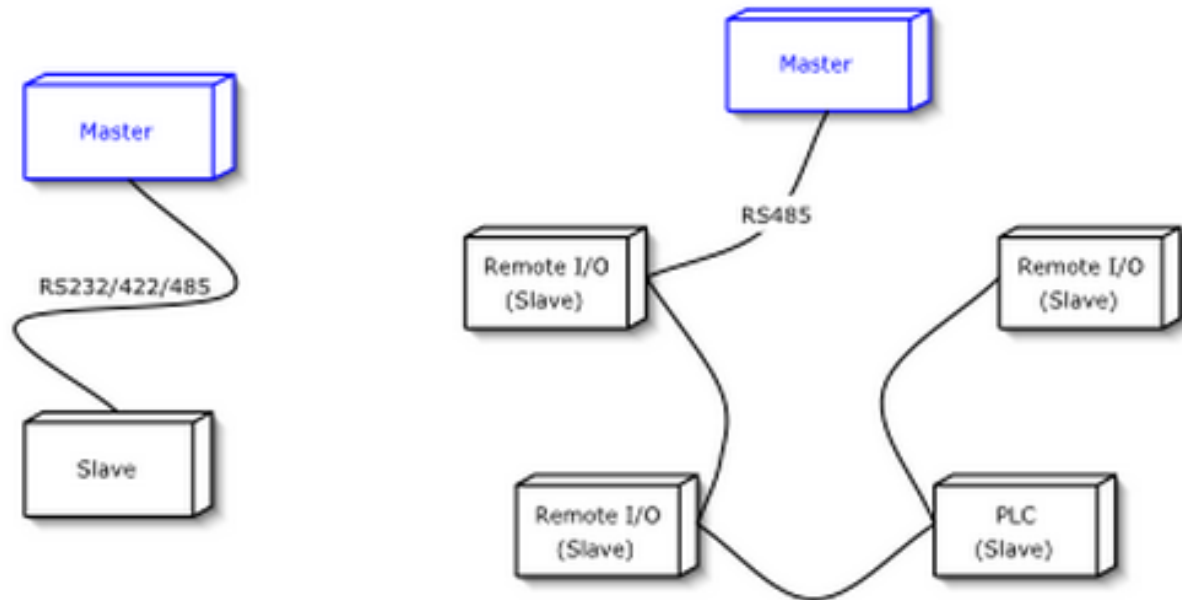
<Modbus Data Type>

Name	Type	Access	Visual
Discrete Input	single bit	read-only	
Discrete Output (Coils)	single bit	read-write	
Input Registers	16-bit word	read-only	
Holding Registers (Registers)	16-bit word	read-write	

6.1.4 구현들

modbus의 구현은 시리얼 통신에 사용할 수 있는 modbus와 IP 통신망에서 사용할 수 있는 modbus가 존재한다.

시리얼 통신에 사용되는 대표적인 규격은 RS-232와 RS-485가 있다. RS-232는 가까운 거리에 놓인 두 장비를 1:1로 연결하는 용도로 사용된다 (RS-422은 RS-232에 양방향 통신기능을 추가한 것이다). RS-485는 여러 장비들을 하나의 망으로 묶을 수 있으며, 하나의 master에서 여러 개의 slave들과 통신할 수 있다.



시리얼 전송 모드는 데이터의 인코딩 방식에 따라 아스키와 RTU로 다시 나누어진다. 아스키 모드에서는 바이트는 두 개의 아스키 문자로 기록되어 데이터 전송효율은 떨어지지만, 디버깅등에서 사람이 데이터를 읽기에는 더 편리한 점이 있다. RTU 모드에서는 이진 데이터를 그대로 전송에 이용한다.

IP 기반의 modbus 구현은 TCP와 UDP 모두 가능하며, IETF에서 502 포트를 할당받았다.

대부분의 modbus 구현들은 필요에 의해 프로토콜을 약간씩 변형하여 구현하였다. 그러므로 변형된 부분에 대해서는 서로 호환이 되지 않을 수 있다. 다음은 변형된 부분들에 대한 내용이다:

- Data types
 - Floating point IEEE
 - 32-bit integer
 - 8-bit data
 - Mixed data types
 - Bit fields in integers
 - Multipliers to change data to/from integer. 10, 100, 1000, 256 ...
- Protocol extensions
 - 16-bit slave addresses
 - 32-bit data size (1 address = 32 bits of data returned.)
 - Word swapped data

6.1.5 제한

- modbus는 1970년대 말에 만들어졌기 때문에 그 당시 사용되던 PLC 장비들에서 사용되던 data type들만을 지원한다. 이전 절의 변형된 data type들을 참고하라.
- data object는 오직 주소에 의해서만 결정되며, 설명을 넣을 수 없다.
- Master의 요청 없이 slave에서 특정 이벤트에 의해 master로 메시지를 전달할 수 있는 방법이 없다(Modbus TCP에서는 가능은 함).
- 시리얼 네트워크에서는 247개의 노드까지만 네트워크에 연결할 수 있다 (Modbus TCP에서는 IP 주소가 허용하는 범위에서 할당 가능).
- 데이터는 연속적으로 전송해야 한다. 버퍼링을 위해 하나의 메시지를 중간에 끊어서 전송할 수 없다.
- 통신 보안을 제공하지 않는다.

6.2 libmodbus

modbus를 구현한 라이브러리는 대표적으로 libmodbus를 뽑을 수 있다. c로 구현된 라이브러리로 리눅스, 맥, 윈도우 등 대부분의 OS에서 사용할 수 있으며, 꾸준히 업데이트가 되고 있다.

본 글에서는 Modbus RTU의 예제를 libmodbus를 이용하여 살펴보고, Modbus TCP에 대해서는 pymodbus를 사용하여 예제를 구동시킬 것이다. 테스트 환경은 ubuntu 12.04와 13.04이다.

6.2.1 libmodbus 설치

<http://libmodbus.org/download/> 에서 최신버전을 다운받아 적당한 곳으로 옮긴 후 아래 명령을 이용하여 압축을 푼다.

```
$ tar zxvf libmodbus-3.0.5.tar.gz
```

libmodbus-3.0.5 폴더로 이동한 후 아래와 같이 컴파일한다.

```
$ cd libmodbus-3.0.5/
$ ./configure
$ make
$ sudo make install
```

libmodbus는 TCP 상으로도 데이터를 전송할 수 있지만, 본 글에서는 modbus RTU의 동작만을 살펴본다.

6.2.2 Modbus RTU 구동 예제

테스트를 위해 시리얼 케이블을 이용하지 않고 가상 시리얼 loopback을 만들어 사용할 수 있다. ubuntu에서는 socat을 이용하여 다음과 같이 만들 수 있다.

```
$ sudo apt-get install socat
$ socat -d -d pty pty
2013/10/30 15:24:38 socat[13104] N PTY is /dev/pts/7
2013/10/30 15:24:38 socat[13104] N PTY is /dev/pts/8
2013/10/30 15:24:38 socat[13104] N starting data transfer loop with FDs [3,3] and [5,
↪5]
```

USB-to-Serial 케이블을 이용한다면, ls 명령을 이용하여 /dev/ttyUSB0 와 같은 식으로 장치가 잡히는 것을 확인할 수 있다.

libmodbus-3.0.5/tests 로 이동하여 화일들을 보면 여러 형태의 테스트 화일들을 볼 수 있다. 본 글에서는 unit-test-server.c와 unit-test-client.c 만을 이용하여 modbus RTU의 간단한 구동을 살펴볼 것이다. 먼저 아래 두 줄을 위 socat 에서 나온 가상 디바이스로 각 화일에서 수정해 준다. 시리얼 케이블을 사용하는 경우 해당 장치의 이름을 적어주면 된다.

<unit-test-server.c>

```
ctx = modbus_new_rtu("/dev/pts/7", 115200, 'N', 8, 1);
```

<unit-test-client.c>

```
ctx = modbus_new_rtu("/dev/pts/8", 115200, 'N', 8, 1);
```

make로 컴파일을 해 준 후, 아래 명령으로 서버를 실행시킨다.

```
libmodbus-3.0.5/tests$ ./unit-test-server rtu
```

다음으로 아래 명령을 이용하여 client를 실행시킨다.

```
libmodbus-3.0.5/tests$ ./unit-test-client rtu
```

다음은 server와 client의 출력이다.

<server>

```
Opening /dev/pts/7 at 115200 bauds (N, 8, 1)
Waiting for a indication...
<11><05><00><13><FF><00><7F><6F> # addr:0x11, func_code:5.. 수신한 메시지
[11][05][00][13][FF][00][7F][6F] # 응답 메시지
Waiting for a indication... # 데이터 수신을 기다림
<11><01><00><13><00><01><0E><9F>
[11][01][01][01][94][88]
Waiting for a indication...
<11><0F><00><13><00><25><05><CD><6B><B2><0E><1B><10><35>
[11][0F][00][13][00][25][67][45]
Waiting for a indication...
```

<client>

```
Opening /dev/pts/8 at 115200 bauds (N, 8, 1)
** UNIT TESTING **

TEST WRITE/READ:
[11][05][00][13][FF][00][7F][6F] # 요청 메시지
Waiting for a confirmation... # 응답을 기다림
<11><05><00><13><FF><00><7F><6F> # 응답 메시지
1/2 modbus_write_bit: OK
[11][01][00][13][00][01][0E][9F]
Waiting for a confirmation...
<11><01><01><01><94><88>
2/2 modbus_read_bits: OK
[11][0F][00][13][00][25][05][CD][6B][B2][0E][1B][10][35]
Waiting for a confirmation...
<11><0F><00><13><00><25><67><45>
1/2 modbus_write_bits: OK
```

modbus 명세서 를 참고하여 각 메시지의 값들을 비교해 보라.

6.3 pymodbus

본 글에서는 pymodbus 를 이용하여 python 코드로 modbus를 사용하는 방법을 살펴본다. 테스트 환경은 ubuntu 12.04와 13.04이며, python 2.7 버전을 이용한다. pymodbus는 시리얼 쪽 구현이 온전하지 않고 오류가 발생한다. 하지만, TCP쪽은 안정적으로 동작하는 것을 확인하였다.

6.3.1 pymodbus 설치

아래 명령을 통해 간단히 설치가 가능하다. pymodbus는 asynchronous 모드를 지원하기 위해 twisted를 이용한다.

```
$ sudo apt-get install python-setuptools
$ sudo apt-get install python-twisted
$ sudo easy_install -U pymodbus
```

문제가 발생한다면, <https://github.com/bashwork/pymodbus> 를 참고하라.

6.3.2 Modbus TCP 구동 예제

pymodbus는 server와 client 모두에 대해 synchronous 및 asynchronous 모드를 지원한다. 먼저 synchronous 모드의 TCP 예제를 살펴보자. 소스 코드를 다운로드 받으면, pymodbus/examples/common 위치에 synchronous-client.py 화일을 아래와 같이 수정한다. input register는 센서 값에서 읽어오는 것이기 때문에 쓸 수가 없다. 예제에 이렇게 되어 있는 것이 좀 의아하다.

```
-- synchronous-client.py 2013-10-31 12:35:38.149374926 +1000
+++ synchronous-client-tcp.py 2013-10-31 12:37:18.317372691 +1000
@@ -55,7 +55,7 @@
#
#     client = ModbusClient('localhost', retries=3, retry_on_empty=True)
#-----#
-client = ModbusClient('localhost', port=502)
+client = ModbusClient('localhost', port=5020)
#client = ModbusClient(method='ascii', port='/dev/pts/2', timeout=1)
#client = ModbusClient(method='rtu', port='/dev/pts/2', timeout=1)
client.connect()
@@ -83,7 +83,8 @@
assert(rr.bits == [True]*8)           # test the expected value

rq = client.write_coils(1, [False]*8)
-rr = client.read_discrete_inputs(1,8)
+#rr = client.read_discrete_inputs(1,8)
+rr = client.read_coils(1,8)
assert(rq.function_code < 0x80)       # test that we are not an error
assert(rr.bits == [False]*8)         # test the expected value

@@ -93,7 +94,7 @@
assert(rr.registers[0] == 10)         # test the expected value

rq = client.write_registers(1, [10]*8)
-rr = client.read_input_registers(1,8)
+rr = client.read_holding_registers(1,8)
assert(rq.function_code < 0x80)       # test that we are not an error
```

(continues on next page)

(continued from previous page)

```
assert(rr.registers == [10]*8)      # test the expected value

@@ -104,7 +105,7 @@
    'write_registers': [20]*8,
}
rq = client.readwrite_registers(**arguments)
-rr = client.read_input_registers(1,8)
+rr = client.read_holding_registers(1,8)
assert(rq.function_code < 0x80)     # test that we are not an error
assert(rq.registers == [20]*8)      # test the expected value
assert(rr.registers == [20]*8)      # test the expected value
```

동작을 확인하기 위해서는 옵션 없이 synchronous-server.py 를 실행시킨후, 다른 창에서 synchronous-client.py 를 실행시키면 된다.

로그 (syslog)

모니터링 시스템에서 로그 보다 더 중요한 것이 있을까? 나는 없다고 생각한다. 자신만의 로그를 만들어 사용할 수도 있겠지만, 사실상의 로그 표준으로 사용되고 있는 syslog, 그 중에서도 ubuntu에서 기본으로 사용하는 rsyslog를 이용하여 로그를 출력하고자 한다. 편의상 rsyslog를 syslog로 부르도록 하고 꼭 rsyslog 로 구분하여 나타낼 필요가 있을 때만 rsyslog 를 사용할 것이다.

syslog를 이용하면, 원격시스템 내부에서는 물론이고 로그의 내용을 서버로 전송하여 화일이나 DB에 저장하여 볼 수 있다. 단순한 기능 같지만, 원격시스템 모니터링에 있어서 가장 중요한 기능 중 하나라고 할 수 있다. 빈번하게 변화하는 값들은 modbus 프로토콜을 이용하여 모니터링하고, 중요한 이벤트나 디버깅 정보등은 syslog로 관리하면 편리하다. syslog는 특정 이벤트가 발생하는 시점들을 관리할 수도 있고, 시스템에 문제가 발생했을 때 원인을 분석하는 디버깅 용도로도 매우 유용하다.

Note: syslog는 IETF의 RFC 5424 로 등록되어 있다. RFC 5424에는 syslog 메시지를 인터넷 상으로 전달하는 방법을 기술한다.

7.1 syslog 출력의 예

다음은 /var/log/syslog 의 내용 일부이다. syslog가 설치된 시스템에서는 기본적으로 시스템에서 발생하는 로그들을 이 화일에 출력한다.

```
Dec  5 10:52:25 ymkim-AO756 anacron[1058]: Job cron.daily terminated
Dec  5 10:52:25 ymkim-AO756 anacron[1058]: Normal exit (1 job run)

^^^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^ ^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
DATE      TIME      hostname      process name  message (log content)
```

위의 예에서 보인바와 같이 syslog의 출력은 날짜와 시간으로 시작한다. 이후 호스트 이름과 프로세스 이름을 출력하며, 여기까지는 syslog에서 자동으로 출력해 주는 부분이다. 콜론(:) 이후의 내용은 로그를 찍는 이유를 설명하는 메시지 부분이다.

Note: apache와 같은 프로그램은 syslog의 기준을 따르지 않는 독자적인 로그를 별도의 로그 파일에 기록한다. ubuntu의 경우 /var/log/apache2/ 아래에 access log와 error log를 별도로 저장한다.

7.2 log level

본격적으로 syslog를 설명하기 앞서서 기본적인 내용을 하나 공부하고 넘어가야 한다. 바로 메시지의 특성을 정의하는 facility와 severity에 대한 내용이다. facility는 메시지를 발생시킨 프로그램의 타입을 나타내는 값이며, severity는 메시지의 성격 또는 중요도를 나타낸다. syslog에서는 이 값에 따라 로그 메시지를 어느 화일에 기록할지, 누구에게 이 사실을 알릴 것인지를 결정한다.

- facility : auth, authpriv, daemon, cron, ftp, lpr, kern, mail, news, syslog, user, uucp, local0, ... , local7

Facility Number	Keyword	C code	Facility Description
0	kern	LOG_KERN	kernel messages
1	user	LOG_USER	user-level messages
2	mail	LOG_MAIL	mail system
3	daemon	LOG_DAEMON	system daemons
4	auth	LOG_AUTH	security/authorization messages
5	syslog	LOG_SYSLOG	messages generated internally by syslogd
6	lpr	LOG_LPR	line printer subsystem
7	news	LOG_NEWS	network news subsystem
8	uucp	LOG_UUCP	UUCP subsystem
9	clock	LOG_CRON	clock daemon
10	authpriv	LOG_AUTHPRIV	security/authorization messages
11	ftp	.	FTP daemon
12	.	.	NTP subsystem
13	.	.	log audit
14	.	.	log alert
15	cron	.	clock daemon
16	local0	LOG_LOCAL0	local use 0 (local0)
17	local1	LOG_LOCAL1	local use 1 (local1)
18	local2	LOG_LOCAL2	local use 2 (local2)
19	local3	LOG_LOCAL3	local use 3 (local3)
20	local4	LOG_LOCAL4	local use 4 (local4)
21	local5	LOG_LOCAL5	local use 5 (local5)
22	local6	LOG_LOCAL6	local use 6 (local6)
23	local7	LOG_LOCAL7	local use 7 (local7)

- severity : Emergency, Alert, Critical, Error, Warning, Notice, Info or Debug

Code	Severity	Keyword	C code	Description
0	Emergency	emerg (panic)	LOG_EMERG	System is unusable.
1	Alert	alert	LOG_ALERT	Action must be taken immediately.
2	Critical	crit	LOG_CRIT	Critical conditions.
3	Error	err (error)	LOG_ERR	Error conditions.
4	Warning	warning (warn)	LOG_WARNING	Warning conditions.
5	Notice	notice	LOG_NOTICE	Normal but significant condition.
6	Informational	info	LOG_INFO	Informational messages.
7	Debug	debug	LOG_DEBUG	Debug-level messages.

facility와 severity에 따라 어떤 화일에 로그를 쓸지에 대해, rsyslog의 경우 /etc/rsyslog.d/50-default.conf 에 정의되어 있다.

7.3 셸 명령어로 syslog 출력하기

시스템에서 발생하는 로그가 아닌 직접 생성한 로그를 만드는 가장 쉬운 방법은 logger 명령어를 이용하는 것이다. man logger 에서 모든 옵션에 대한 설명을 찾을 수 있으며, 여기서는 몇 가지 예만을 보도록 한다.

```
$ logger log test..
$ tail -f /var/log/syslog
```

tail 명령어 출력 결과의 마지막에서 아래와 같은 내용을 볼 수 있을 것이다.

```
Dec  5 15:30:37 ymkim-AO756 ymkim: log test..
```

log level을 입력으로 넣기 위해서는 -p 옵션을 사용한다. -p 옵션 뒤에 facility.severity 를 입력한다. -p 옵션을 사용하기 전에 local0 facility의 새로운 출력화일을 지정하기 위해 /etc/rsyslog.d/50-default.conf 의 제일 아래 줄에 다음 내용을 추가하자.

```
local0.*      /var/log/test.log
```

Note: rsyslog의 메인 설정화일은 /etc/rsyslog.conf 이다. 이 화일에서 /etc/rsyslog.d/50-default.conf 을 불러와 추가적인 내용을 설정한다. 설정화일에 대한 자세한 내용은 man rsyslog.conf 를 확인하라.

변경내용을 저장한 후 아래 명령으로 rsyslog를 재시작한다.

```
$ sudo restart rsyslog
```

이제 아래 명령을 실행시키면 /var/log/test.log 와 /var/log/syslog 에서 입력한 로그 메시지를 확인할 수 있다.

```
$ logger -p local0.info log test 2..
```

/var/log/syslog 로는 로그를 쓰지 않도록 하기 위해 /etc/rsyslog.d/50-default.conf 설정화일에서 아래 내용을 찾아서,

```
*.*;auth,authpriv.none      -/var/log/syslog
```

다음과 같이 local0.none을 추가하라

```
*.*;auth,authpriv.none,local0.none      -/var/log/syslog
```

이 줄의 의미는 모든 로그(*.*)를 /var/log/syslog에 기록하지만, 세미콜론(;) 이후의 facility-들인 auth, authpriv, local0 은 제외(none)하라는 것이다. 화일이름 앞의 - 은 로그를 화일에 바로 쓰지 말고 메모리에 로그를 가지고 있다가 디스크에 입출력 여유가 있을 경우 쓰라는 의미이다 (<http://shallowsky.com/blog/linux/rsyslog-conf-tutorial.html> 의 Rules Section을 보라).

7.4 hostname 설정

ubuntu 설치시 사용자 아이디와 컴퓨터 이름을 조합해 자동으로 hostname을 만들어 준다. 참 편리한 기능이다. 하지만, 동일한 장비에 ubuntu를 설치하면 모든 장비의 hostname이 같아지는 현상이 발생한다. syslog는 원격으로 로그를 보내 통합하여 관리하는 기능이 있으므로 hostname을 다르게 설정해 주는 것이 중요하다.

`/etc/hostname` 과 `/etc/hosts` 를 열어 자동으로 설정된 `hostname`을 변경한다. 예를 들어 `hostname`의 마지막에 컴퓨터마다 다른 숫자를 넣어 컴퓨터들을 구별할 수 있다. 로그 아웃후 다시 로그인을 하면 변경된 `hostname`이 적용된 것을 아래 명령으로 확인할 수 있다.

```
$ hostname
```

7.5 C 코드에서 syslog 출력하기

`joinc`의 `syslog` 에서 잘 설명되어 있으며, 여기서는 간단한 따라하기를 소개한다.

`facility level`에서 `local0`에서 7까지 총 8개의 `facility`를 사용자의 목적에 맞게 사용할 수 있다. 로그를 발생시키는 프로그램별로 `facility`를 할당할 수도 있고, 이용목적에 따라 여러 프로그램에서 하나의 `facility`에 값을 쓰도록 할 수 있다. 물론 같은 `facility` 내에서도 `severity`에 따라 다른 화일에 로그를 저장할 수도 있다.

위에서부터 쪽 따라온 분은 `/etc/rsyslog.d/50-default.conf` 화일 아래에 다음 줄이 있을 것이다.

```
local0.*      /var/log/test.log
```

아직 없다면 추가하고 `sudo restart rsyslog` 를 수행하라.

c 코드에서 다음과 같이 `local0`에 쓰도록 하면 `/var/log/test.log` 에 로그가 기록된다.

```
#include <syslog.h>

int main()
{
    syslog(LOG_INFO | LOG_LOCAL0, "write your log message");
    return 0;
}
```

실행결과는 아래와 같이 확인할 수 있다.

```
$ gcc tt.c
$ ./a.out
$ tail /var/log/test.log
Dec  9 12:14:55 ymkim-SD550 a.out: write your log message
```

7.6 logrotate로 최신 로그만 남기기

하나의 화일에 계속 로그가 쌓이다 보면 화일이 커져서 로딩하기 위해 걸리는 시간도 길어지고 디스크의 용량도 많이 차지하게 된다. 이러한 문제를 해결하기 위해 일정시간 단위로 로그화일을 구분하여 저장하고 오래된 로그 화일은 지우는 방법이 있다. `logrotate`가 하는 일이 바로 이것이다.

`/etc/logrotate.d` 는 `logrotate`의 설정화일들이 위치하는 곳이다.

```
$ ls
apport      cups-daemon  ppp          speech-dispatcher  upstart
apt         dpkg         rsyslog      ufw
consolekit  pm-utils    samba        unattended-upgrades
```

위의 화일들중에서 `dpkg` 을 이용해 설명하고자 한다. 화일을 보면, 앞 부분에 이런 내용이 있다.

```

1 /var/log/dpkg.log {
2     monthly
3     rotate 12
4     compress
5     delaycompress
6     missingok
7     notifempty
8     create 644 root root
9 }

```

다른 화일이나 이 화일의 나머지 부분도 대략 이런 형식을 가지고 있다. 첫 줄에는 화일 이름이 있다. 즉 이 화일에 대해 설정을 하겠다는 뜻이다. 여러 화일에 동일한 설정을 적용하고자 할 경우, 화일 이름을 한 줄씩 적고 { 을 열 수 있다.

```

/var/log/mail.info
/var/log/mail.warn
/var/log/mail.err
/var/log/mail.log
/var/log/daemon.log
/var/log/kern.log
/var/log/auth.log
/var/log/user.log
/var/log/lpr.log
/var/log/cron.log
/var/log/debug
/var/log/messages
{
    rotate 4
}

```

또는 아래와 같이 정규 표현식을 사용할 수도 있다.

```

/var/log/mail.*
{

```

두번째 줄의 `monthly`는 `cron`에 의해 처리되는 부분이며, 한달 단위로 로그화일을 잘라서 관리하겠다는 의미이다. 즉 `cron`에 의해 정해진 시간에, 첫줄에 명시된 화일을 화일명.1로 변경하고 새로운 빈 화일명의 화일을 생성한다. 이 때 이전 로그들의 마지막 숫자도 전부 1씩 증가한다.

Note: `/etc/cron.daily/logrotate` 화일이 `cron`에 등록되어 매일 수행되며, `/etc/crontab`에는 어떤 시간에 `daily cron`이 실행되는지 적혀있다. `ubuntu`에서는 오전 6시 25분이다.

```

25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /
→etc/cron.daily )

```

세번째 줄의 `rotate 12`는 총 12개의 화일, 즉, 화일명, 화일명.1, ..., 화일명.11을 유지한다는 의미이며, 이 예에서는 1년치 로그를 보관한다.

네번째와 다섯번째 줄은 마지막에 숫자가 붙은 로그들을 압축하여 관리하는 것을 뜻하며, 여섯번째 줄의 `missingok`는 연속적으로 번호가 부여되어 존재해야 하는 로그 파일 중에서 빠진 부분이 있더라도 무시하고 넘어가겠다는 의미이다.

`notifempty`는 로그 화일이 비어 있을 경우에는 `logrotate`를 수행하지 않도록 한다. 마지막 `create`는 생성할 로그의 권한을 설정한다.

Note: 화일크기에 따라 로그를 구분하기 위해서는 <https://www.digitalocean.com/community/articles/>

[how-to-manage-log-files-with-logrotate-on-ubuntu-12-10](#) 을 보라.

새로운 logrotate를 추가하기 위해서는 /etc/logrotate 아래에 임의의 화일을 생성하고 위의 내용을 참고하여 목적에 맞게 내용을 추가하면 된다.

7.7 Centralized logging

원격시스템의 상태를 확인하기 위해 원격시스템에 접속하여 개별 로그를 확인하는 방법은 확실한 방법이지만, 원격시스템의 수가 많을 경우, 어려운 방법이 될 수 있다. centralized logging 이란 중앙에서 관리해야 할 중요한 정보들에 대해 원격시스템에서 서버로 로그를 전송한 후, 서버에서 이 로그들을 관리하는 방법을 의미한다.

원격시스템에서 서버로 로그를 보내는 것은 비교적 간단하게 설정할 수 있지만, 서버에서는 로그를 화일에 저장할지, DB에 저장할지, 저장된 내용을 어떻게 보여줄지 등은 좀 더 복잡한 설정이 필요하다.

Note: 본 절은 <http://www.linuxjournal.com/content/centralized-logging-web-interface> 을 참고하여 작성하였다.

7.7.1 시간 동기화

원격시스템들의 시간이 일치하지 않는다면, 한 곳에 모아진 로그의 의미를 이상하게 만들 수도 있다. 즉 정확한 시간 동기화를 해야 한다는 의미이다. 아래 명령을 이용하면 간단하게 인터넷에 연결된 시간서버로 부터 시간을 동기화한다.

```
$ ntpdate ntp.ubuntu.com
```

한 번 맞추어 놓은 시간도 시간이 지남에 따라 조금씩 차이가 생기기 마련이다. 이를 방지하기 위해서 cron에 등록하여 하루에 한 번씩 ntpdate를 수행하도록 한다. /etc/cron.daily/ntpdate 파일을 생성하고 위의 명령어인 ntpdate ntp.ubuntu.com 을 입력하고 저장한다. 실행권한을 주기 위해 아래 명령을 수행하라.

```
$ chmod 755 /etc/cron.daily/ntpdate
```

이제 매일 정해진 시각마다 시간동기화를 실행한다.

7.7.2 서버에서 원격시스템의 로그 받기

원격시스템에서 보내는 로그를 서버에서 받아들이기 위해서는 /etc/rsyslog.conf 화일에서 아래 부분을 찾아 주석을 해제하여야 한다. TCP를 사용할지 UDP를 사용할지에 따라 선택적으로 주석을 해제할 수도 있다.

```
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
# provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

rsyslog에서 통신 모듈의 작명방법은 input module일 경우, im으로 시작하고 output module일 경우 om으로 시작하며, 그 뒤에 프로토콜의 이름이 붙는다. imudp는 UDP input module 이라는 의미이다. 이상의 설정에서 514 TCP/UDP 포트를 열어 원격시스템의 로그를 받아들일 수 있다. 아래와 같이 rsyslog를 재시작하여 설정을 적용한다.

```
$ sudo restart rsyslog
or
$ sudo service rsyslog restart
```

Note: 서버로 전송하는 로그의 양이 엄청나게 많고 원격시스템의 수도 많다면 서버에서는 넘쳐나는 로그를 처리하지 못할 수도 있다. 이런 상황에서도 모든 로그를 처리하도록 하기 위해 RELP (Reliable Event Logging Protocol) 라는 프로토콜이 제안되었으며, 로그를 버퍼에 저장하는 방법과 함께 활용하여 신뢰성을 향상시킬 수 있다. <http://www.linuxjournal.com/content/centralized-logging-web-interface> 을 참고하라.

7.7.3 서버로 syslog 출력 보내기

/etc/rsyslog.d/50-default.conf 에 추가했던 아래 내용을 다시 보자.

```
local0.*      /var/log/test.log
```

UDP로 로그를 보내고자 한다면, 아래처럼 수정하고

```
local0.*      @your_server_name_or_ip_address
```

TCP를 통해 로그를 보내고자 한다면, 아래처럼 수정한다.

```
local0.*      @@your_server_name_or_ip_address
```

서버에서 대기포트번호를 변경하였을 경우에는 아래와 같은 형식으로 514 포트번호를 바꿀 수 있다.

```
local0.*      @your_server_name_or_ip_address:new_port
ex) local0.info @monitor.com:10514
```

Note: 본 절의 내용은 이 문서 를 참고하라(18쪽 중간쯤)

전송한 로그는 서버의 /var/log/syslog 에 기록된다.

Note: 원격시스템에서 로그 메시지를 전송할 시점에 통신이 불가능하였을 경우 통신이 복구되고 나서 다음 로그 메시지를 전송할 때 이전에 전송되지 못한 메시지도 함께 전송된다.

7.7.4 웹에서 syslog 결과보기 (Log Analyzer)

Log Analyzer 를 개발하고 있는 Adiscon 사는 RFC 5424 (The syslog protocol) 을 제출한 회사이다. 즉 syslog 전문 회사로 웹에서 syslog를 분석할 수 있는 툴을 제공한다.

Log Analyzer는 php로 작성된 프로그램으로 웹서버인 apache와 mysql 데이터베이스를 이용하여 사용자에게 최종 화면을 제공한다.

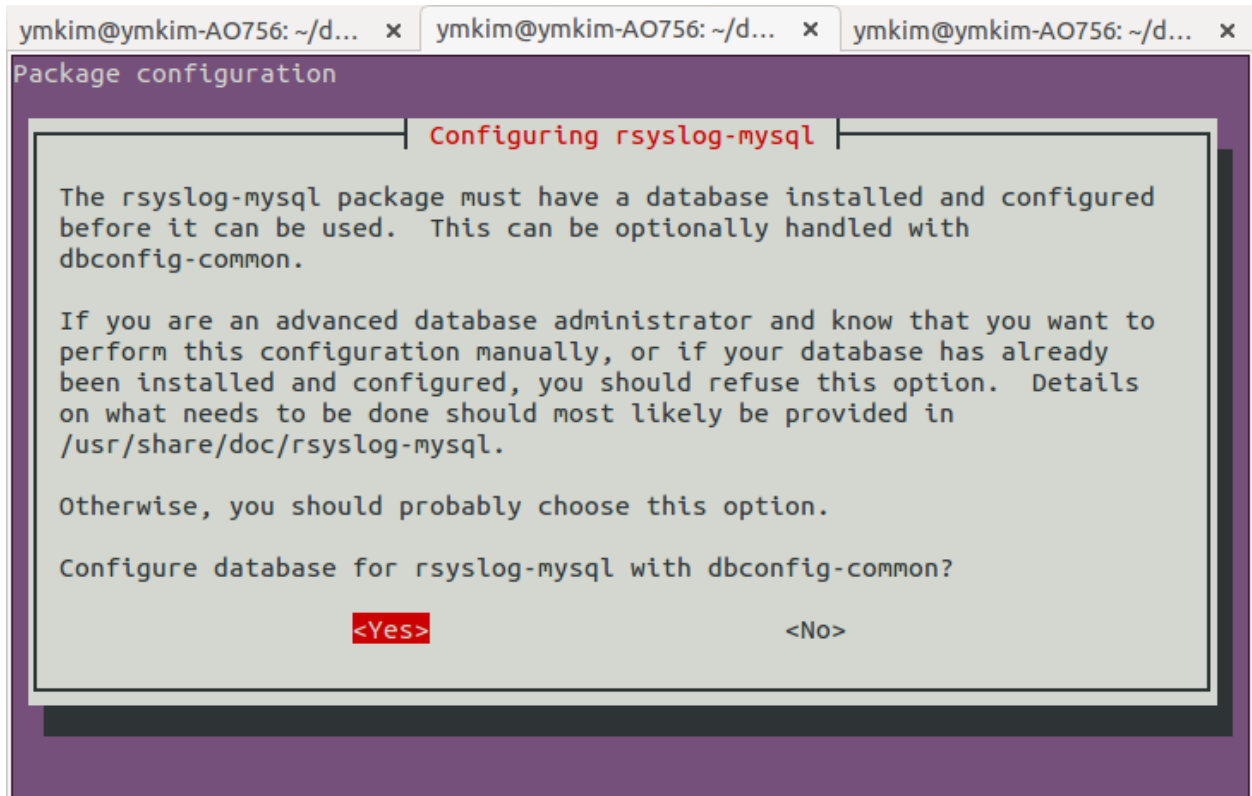
Note: ubuntu에서는 taskel 이라는 명령어를 이용하여 LAMP 를 선택하면 쉽게 Apache + Mysql + Php 환경을 구축할 수 있다. 추가적으로 phpmyadmin 을 설치하면 웹에서 mysql 을 편리하게 관리할 수 있다.

```
$ sudo apt-get install phpmyadmin
```


apache와 mysql, php가 설치되어 있다는 가정하에 이하 내용을 진행한다. syslog의 로그를 화일이 아닌 데이터베이스에 저장하면, 검색어를 이용해 보고 싶은 로그만을 간추려 보기에 편리하다.

```
$ sudo apt-get install rsyslog-mysql
```

위의 명령어를 입력하여 syslog의 로그를 mysql로 입력하는 프로그램을 설치한다. 설치중에 새로운 데이터베이스와 테이블을 만들 것을 요구하는 화면이 나온다. Yes를 눌러 데이터베이스를 만들어라.



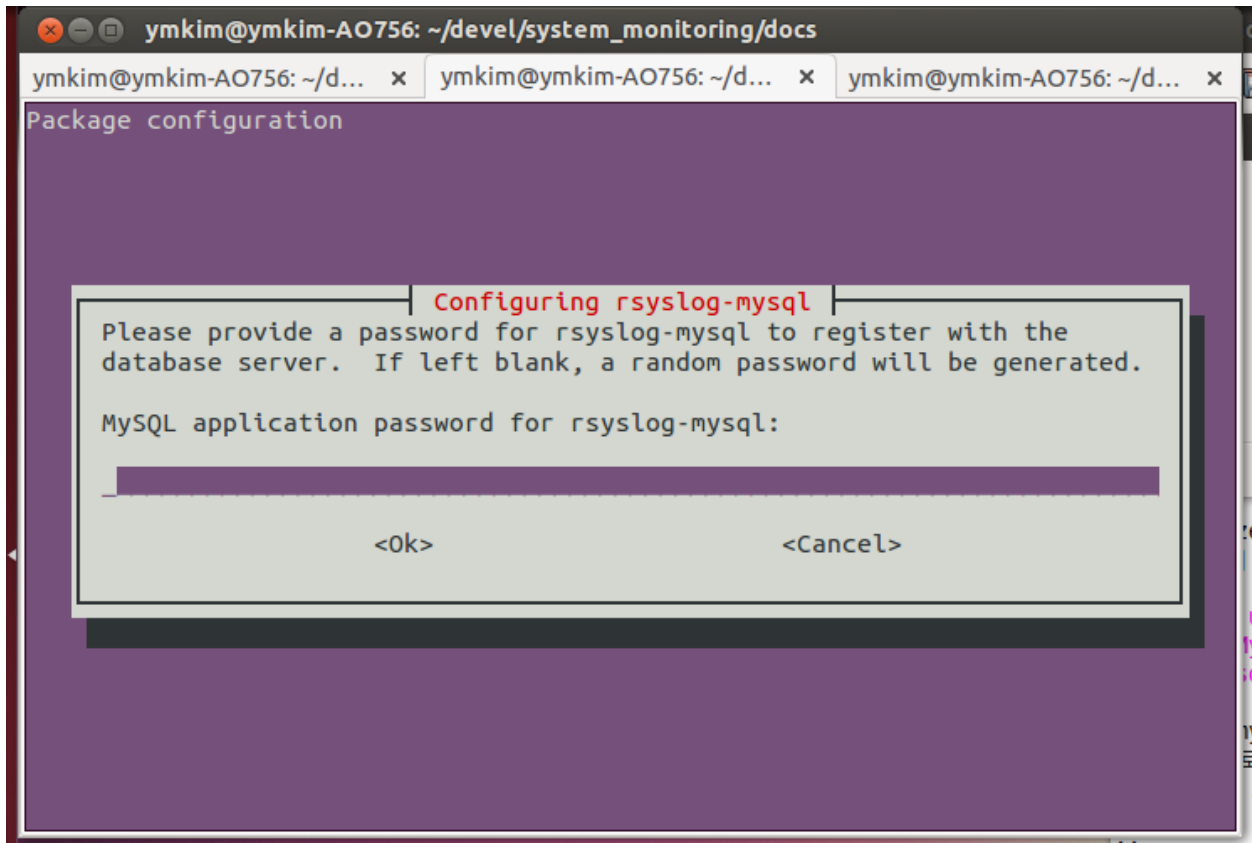
다음은 mysql 관리자(root)의 비밀번호를 묻는다. mysql 설치시 또는 이후 변경한 관리자 비밀번호를 입력한다.

그 다음으로 rsyslog-mysql 사용자를 위한 새로운 비밀번호 입력을 요구한다. 확인을 위해 한 번 더 비밀번호를 입력하라 (이 부분은 phpmyadmin을 설치할 때도 동일하게 묻는 것이다).

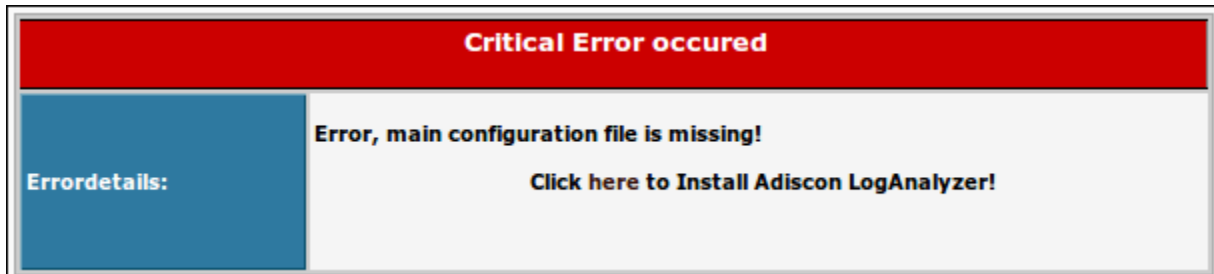
rsyslog-mysql의 설치가 완료되면 /var/log/syslog 뿐만 아니라, mysql 데이터베이스에도 로그 메시지들이 저장된다.

이제 데이터베이스에 대한 준비는 완료되었다. 본격적으로 Log Analyzer를 설치해 보자. <http://loganalyzer.adiscon.com/downloads> 에서 최신 버전을 다운받으라. 다운 받은 폴더로 이동한 후 아래를 차례대로 실행하라.

```
$ tar zxvf loganalyzer-3.6.5.tar.gz
$ cd loganalyzer-3.6.5/
$ sudo mkdir /var/www/logs
$ sudo cp -R src/* /var/www/logs
$ sudo cp -R contrib/* /var/www/logs
$ cd /var/www/logs
$ sudo chmod +x configure.sh secure.sh
$ sudo ./configure.sh
```

이제 브라우저에 `http://localhost/logs` 를 입력하면, 아래와 같은 오류 화면이 나온다. 오류가 나오는 것이 정상이므로 놀라지 말라~~



Click here to Install Adiscon LogAnalyzer! 에서 here 를 누르면 설정화면으로 이동한다. 다음 내용을 참고하여 설정을 진행하라.

Note: 설치 후에도 웹에서 아래 설정들을 변경할 수 있다.

- Number of syslog messages per page = 50 (default)
 - 페이지당 출력되는 메시지의 수를 정의한다. 웹 인터페이스에서도 이 값을 변경할 수 있으므로 우선은 default를 선택하라.
- Message character limit for the main view = 80 (default)
 - 로그 메시지를 몇 글자까지 표시할 지 결정한다. 잘린 부분은 마우스를 가져다 대면 볼 수 있다.
 - 이 값을 0으로 설정하면, 전체 메시지를 모두 출력하므로 많은 사람들이 사용하는 값이다.

- Show message details popup = yes (default).
 - 팝업을 싫어하시는 분은 no를 사용하라.
- Enable User Database = no (default).
 - 화일이 아닌 데이터베이스에 로그를 저장하였을 경우 yes로 변경하고 그 아래에 정보들을 입력한다.
 - Database Name = Syslog

Frontend Options	
Number of syslog messages per page	50
Message character limit for the main view	80
Character display limit for all string type fields	30
Show message details popup	<input type="radio"/> Yes <input checked="" type="radio"/> No
Automatically resolved IP Addresses (inline)	<input checked="" type="radio"/> Yes <input type="radio"/> No
User Database Options	
Enable User Database	<input checked="" type="radio"/> Yes <input type="radio"/> No
A MYSQL database Server is required for this feature. Other database engines are not supported for the User Database System. However for logsources, there is support for other database systems.	
Database Host	localhost
Database Port	3306
Database Name	Syslog
Table prefix	
Database User	root
Database Password	••••
Require user to be logged in	<input checked="" type="radio"/> Yes <input type="radio"/> No
Authentication method	Internal authentication ▼

- Step 6 - Creating the Main Useraccount
 - loganalyzer 웹 화면에 접근할 때 물어보는 아이디와 비밀번호를 설정한다.
- Step 7

First Syslog Source	
Name of the Source	My Syslog Source
Source Type	MYSQL Native ▼
Select View	Syslog Fields ▼
Database Type Options	
Table type	MonitorWare ▼
Database Host	localhost
Database Name	Syslog
Database Tablename	SystemEvents
Database User	rsyslog
Database Password	••••
Enable Row Counting	<input type="radio"/> Yes <input checked="" type="radio"/> No

정상적으로 설정을 마쳤다면 Step 6에서 입력한 아이디와 비밀번호로 로그인 후 아래와 같은 결과를 볼 수 있다. search 버튼을 누르면 다양한 검색 옵션을 사용할 수 있다.

ANALYSIS & REPORTING

Select Source

My Syslog Source

Select View

Syslog Fields

Search

Show Events

Statistics

Reports

Help

Search in Knowledge Base

Admin Center

Logout

Logged in as "log"

Search (filter):

Search

I'd like to feel sad

Reset search

Highlight >>

Advanced Search

(sample: facility:local0 severity:warning)

Recent syslog messages

Set auto reload:

Auto reload d

Records per page:

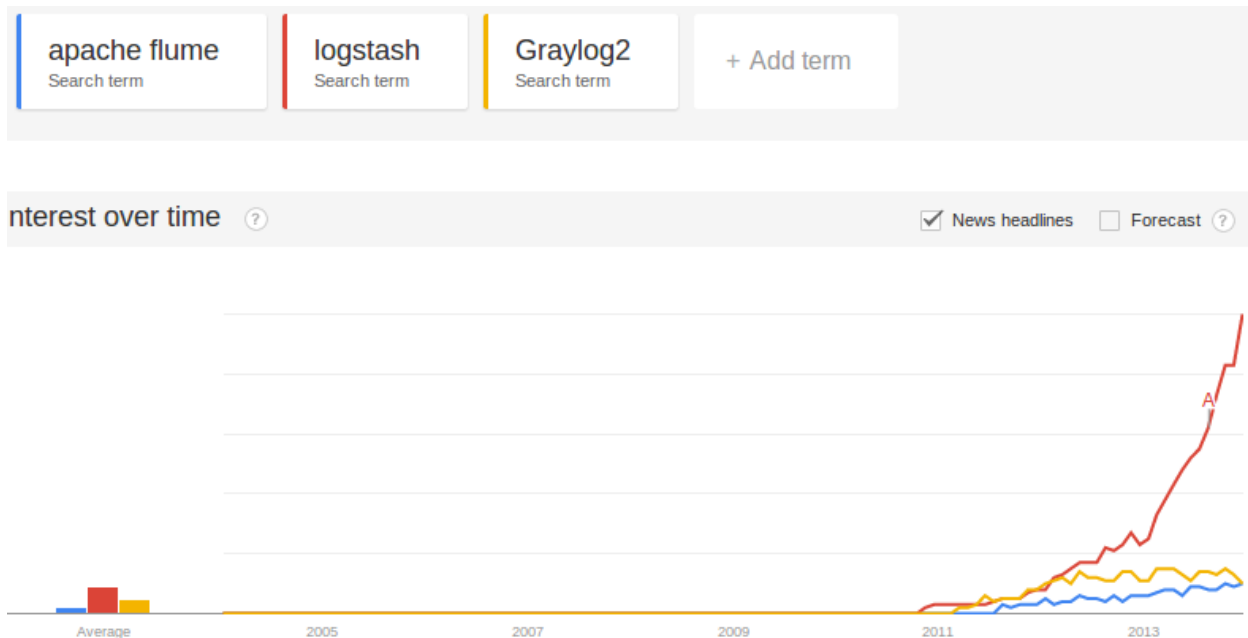
Preconfigure

Pager:

Date	Facility	Severity	Host	Syslogtag	ProcessID	Message	Message
Today 15:10:58	DAEMON	INFO	ymkim-AO756	NetworkManager[970]:		<info> (wlan0): supplicant interface state: associating -> completed	
Today 15:10:58	KERN	INFO	ymkim-AO756	kernel:		[3089.019674] wlan0: associated	
Today 15:10:58	KERN	INFO	ymkim-AO756	kernel:		[3089.019677] wlan0: RX AssocResp from 00:26:66:72:6d:2e (capab=0x1 status=0 ai ...	
Today 15:10:58	DAEMON	NOTICE	ymkim-AO756	wpa_supplicant[1014]:		wlan0: CTRL-EVENT-CONNECTED - Connection to 00:26:66:72:6d:2e completed (reauth ...	
Today 15:10:58	DAEMON	NOTICE	ymkim-AO756	wpa_supplicant[1014]:		wlan0: Associated with 00:26:66:72:6d:2e	
Today 15:10:58	DAEMON	INFO	ymkim-AO756	NetworkManager[970]:		<info> (wlan0): supplicant interface state: authenticating -> associating	
Today 15:10:58	KERN	INFO	ymkim-AO756	kernel:		[3089.014658] wlan0: associate with 00:26:66:72:6d:2e (try 1/3)	
Today 15:10:58	KERN	INFO	ymkim-AO756	kernel:		[3089.012779] wlan0: authenticated	

7.7.5 최근 동향

<http://jasonwilder.com/blog/2012/01/03/centralized-logging/> 에는 최근에 새로 등장하고 있는 centralized logging 툴들을 소개하고 있다. google trends로 분석해 본 결과 logstash 의 인기가 급상승하고 있다.



CHAPTER 8

nagios

모니터링이 무엇이냐고 묻는다면, 뭐라고 답할 수 있을까? modbus와 syslog, nagios 차원에서의 답은 이런식이 아닐까 한다.

모니 터링 방법	모니터링이란
mod- bus	원격시스템의 변화하는 값들을 주기적으로 관찰하여 표시하고, 필요할 경우 서버에서 원격시스템의 설정값을 변경한다.
sys- log	원격시스템에서 발생하는 이벤트 메시지를 서버에서 통합관리한다.
na- gios	서버에서 원격시스템에게 정의된 내용을 묻거나, 원격시스템에 정의된 이벤트가 발생할 경우 서버로 보고한다. 단순 정보 수집에 그치지 않고 미리 정해진 횟수 만큼 연속적으로 오류가 발생하면, 정해진 관리자들에게 알림 기능을 제공한다.

정보의 방향 차원에서는 아래표를 보라.

모니터링 방법	정보의 방향
modbus	양방향 (서버 주도: 서버에서 질의를 보내고 원격시스템에서 응답을 받는 형식)
syslog	단방향 (원격시스템에서 서버로 로그 메시지 전달)
nagios	양방향 (원격시스템에서 서버로 정보를 전달할 수도 있고 서버 주도로 정보를 수신하거나 원격 시스템을 제어할 수도 있음)

nagios는 간단한 명령어에서 스케줄링, 알림, 웹 UI 까지 넓은 영역을 포함하고 있어 한마디로 쉽게 설명하기 어렵다. 이 글이 nagios의 전반적인 이해를 돕는데, 도움이 되기를 바란다.

Note: Icinga, Shinken 등 nagios 와 동일한 또는 비슷한 기능을 하는 소프트웨어 변종들이 존재한다.

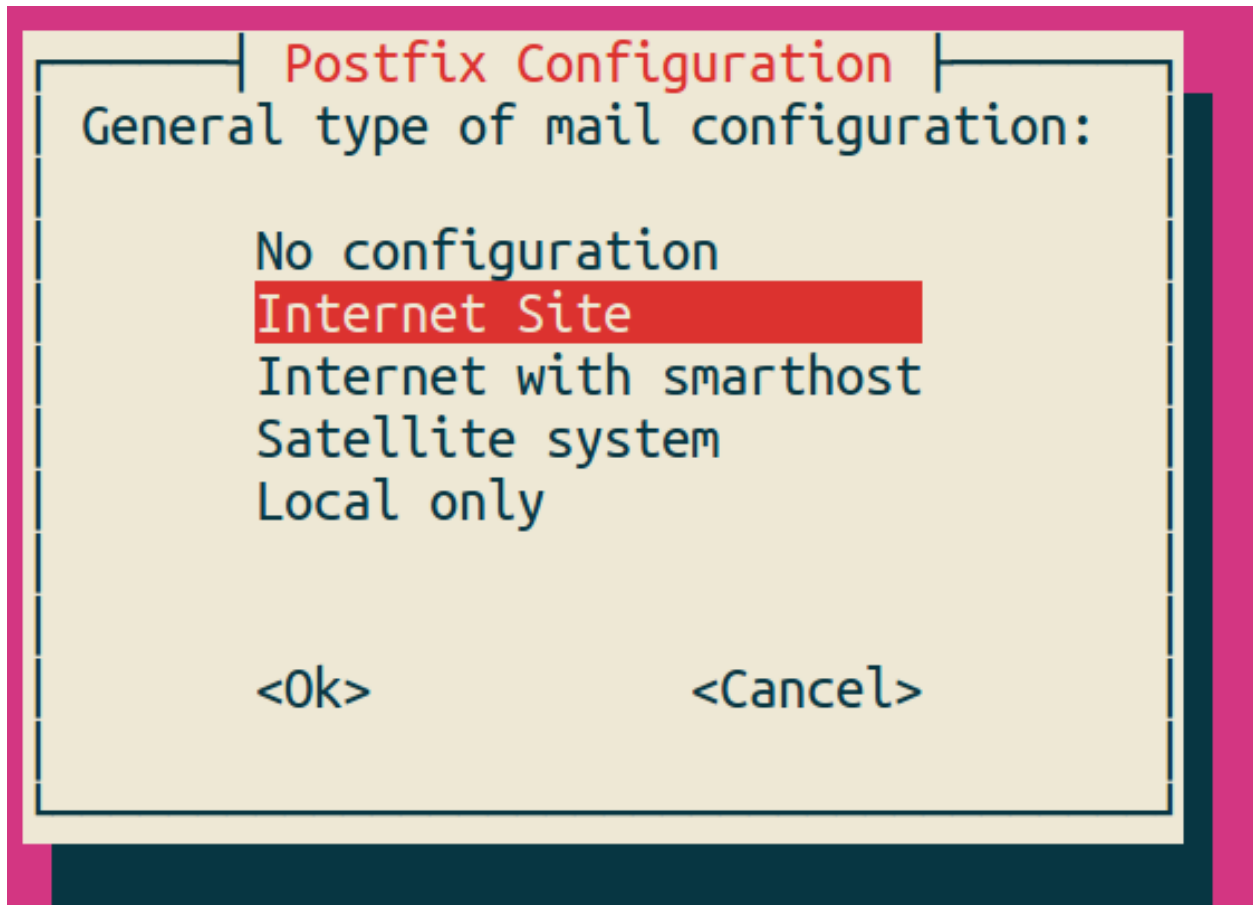
8.1 nagios 설치

Note: 이 절은 <http://askubuntu.com/questions/145518/how-do-i-install-nagios> 을 참고하여 작성되었다.

ubuntu에서 nagios 설치하는 간단하다. apache는 이미 설치되어 있다는 가정하에 아래 명령을 수행한다.

```
$ sudo apt-get install -y nagios3
```

명령 수행 후 몇몇 입력창을 보게 될 것이다. 제일 먼저 나오는 창은 메일 전송을 담당하는 postfix 에 대한 설정이다. postfix는 nagios를 설치하고 웹화면을 구경한 후에 설치 및 설정할 수 있으므로 이 시점에서 중요한 부분은 아니라고 할 수 있다. “Postfix Configuration” 이 제목인 첫 창에서 아래와 같이 Internet Site를 선택한 후 OK를 클릭한다.



다음에 나오는 도메인 네임은 보내는 메일 주소에 사용될 @의 뒷 부분을 입력한다.

다음으로는 nagios 웹 화면에 접속할 비밀번호를 입력하는 창이 나온다. 아이디는 nagiosadmin 이며, 이 아이디와 여기서 입력한 비밀번호로 웹화면에 로그인 할 수 있다.

설치를 마친 후 브라우저에서 <http://localhost/nagios3> 를 입력하면 로그인 창이 나오고 아이디와 비밀번호를 입력하면 nagios 웹 화면을 볼 수 있다.

nagios는 기본적으로 localhost의 load, 현재 사용자, 디스크 공간 등을 검사하도록 설정화일을 자동으로 생성한다. nagios 웹 화면 좌측 메뉴에서 services 를 누르면 아래와 같이 기본으로 설정된 항목들을 볼 수 있다(ubuntu 13.04 환경).

The screenshot shows the Nagios web interface. On the left is a sidebar with navigation links like General, Home, Documentation, Current Status, Tactical Overview, Map, Hosts, Services, Host Groups, Service Groups, Problems, Reports, Availability, Trends, Alerts, and History. The main content area is titled 'Current Network Status' and includes a 'Host Status Totals' section with a table showing 1 Up, 0 Down, 0 Unreachable, and 0 Pending. Below this is a 'Service Status Totals' section with a table showing 4 Ok, 0 Warning, 0 Unknown, 2 Critical, and 0 Pending. The central part of the page is titled 'Service Status Details For All Hosts' and displays a table of services for the host 'localhost'. The table has columns for Host, Service, Status, Last Check, Duration, Attempt, and Status Information. The services listed are Current Load (OK), Current Users (OK), Disk Space (CRITICAL), HTTP (OK), SSH (CRITICAL), and Total Processes (OK). The status information for Disk Space and SSH indicates critical errors related to disk space and connection refusal.

Note: Disk Space와 SSH에서 발생한 에러에 대해서는 <https://help.ubuntu.com/community/Nagios3> 에서 Post Install Tasks를 참고하라.

이제 nagios 설치를 완료하였으므로 설정을 해야 하지만, 그전에 nagios의 기본 개념에 대한 이해를 하고 넘어가자.

8.2 nagios core에 대한 간단한 소개

Note: 이 절은 David Dosephen의 Building a Monitoring Infrastructure with Nagios 를 참고하여 작성되었다.

nagios의 핵심은 작은 크기의 모니터링 프로그램인 plugin 의 스케줄링과 알림(notification)을 위한 프레임워크라고 정의할 수 있다. 본 절에서는 nagios plugin의 원리와 구조에 대해 알아본 후, nagios의 스케줄링에 대해 알아보려고 한다.

8.2.1 nagios plugin

plugin은 exit 코드를 반환하여 plugin의 실행결과를 nagios에게 알릴 수 있으며, exit 코드는 아래와 같은 의미를 갖는다.

Code	Meaning
0	Ok
1	Warning
2	Critical
3	Unknown

Note: ls와 같은 유닉스 명령어도 nagios plugin과 동일한 방식으로 exit code를 반환하며, echo \$? 명령어로 결과를 확인할 수 있다.

nagios plugin은 exit code 이외에도 문자열을 반환하여 세부 정보를 관리자에게 알릴 수 있다. 다음은 예제 plugin 이며, echo문에서 문자열을 출력하고 exit문으로 exit code를 반환한다.

```
#!/bin/sh
OUTPUT='ping -c5 8.8.8.8 | tail -n2'
if [ $? -gt 0 ]
then
    echo "CRITICAL!! $OUTPUT"
    exit 2
else
    echo "OK! $OUTPUT"
    exit 0
fi
```

nagios plugin의 역할은 다음 두가지로 나눌 수 있다.

- Host로부터 정보를 가져온다. (예, CPU 로드, index.html)
- Host의 특정 상태나 비교 결과를 exit code로 반환한다.

이상의 내용에서 알 수 있는 바와 같이 nagios plugin은 독립적인 명령어의 역할도 수행할 수 있으므로 테스트 목적으로 간단하게 사용해 볼 수 있다.

Note: nagios에서는 많은 수의 plugin을 제공하고 있다. <https://www.nagios-plugins.org/> 를 참고하라.

원격지의 호스트에 대해서도 nagios를 실행할 수 있다. 이 절에서는 ssh를 이용한 원격 모니터링의 원리 설명에 집중할 것이다. 원격 호스트의 상태를 모니터링하기 위해서 ssh의 원격지 명령어 수행방법을 이용한다. 아래 명령은 원격 호스트 example.org의 test 계정 홈 디렉토리에서 ls를 수행한 결과를 반환한다.

```
$ ssh test@example.org "ls -CF"
build/                               log/
tmp/
```

이 명령어에서 “ls -CF” 부분을 nagios plugin으로 교체하면 ssh 문 자체로 nagios plugin과 같은 역할을 하게 된다.

원격호스트(example.org)에 /usr/local/bin/load_checker.sh를 생성하고 아래 코드를 내용으로 입력하라. 시스템 부하의 값이 1를 넘어가면 Critical 오류를 발생시키는 코드이다.

```
#!/bin/bash
LOAD=`uptime | awk '{print $12}'`

if (( $(bc <<< "$LOAD > 1") ))
then
    echo "Critical! load on 'hostname' is $LOAD"
    exit 2
else
    echo "OK! Load on 'hostname' is $LOAD"
    exit 0
fi
```

다음 명령을 실행하여 실행권한을 주고 실행시켜 보자.

```
$ sudo chmod a+x /usr/local/bin/load_checker.sh
$ load_checker.sh
OK! Load on 'hostname' is 0.15
$ echo $?
0
```

이제 아래 명령으로 원격호스트의 명령을 실행시킬 수 있다.

```
$ ssh test@example.org /usr/local/bin/load_checker.sh
OK! Load on 'hostname' is 0.13
$ echo $?
0
```

위의 ssh 문을 nagios의 plugin으로 만들기 위해 아래와 같은 스크립트를 작성하여 서버에 저장한다.

```
#!/bin/sh
#get the output from the remote load_checker script
OUTPUT=`ssh test@example.org "/usr/local/bin/load_checker.sh"`

#get the exit code
CODE=$?
echo $OUTPUT
exit $CODE
```

nagios 서버에 위치한 위 코드는 완벽한 nagios plugin으로 원격호스트의 시스템 부하에 대한 출력문과 exit 코드를 반환한다.

이 방법은 nagios에서의 원격 모니터링 원리를 잘 설명하지만, 하나의 단점이 존재한다. 서버에서 원격 시스템으로 로그인 없이 ssh 접속이 가능해야 한다. 이에 대해서는 [서버에서 원격시스템 접근](#) 을 참고하라.

nagios에서는 ssh를 이용하는 방법 이외에 nagios에서 개발한 NRPE (Nagios Remote Plugin Executor)를 이용해 원격시스템의 모니터링을 수행할 수 있다. 자세한 방법은 각자 알아보시고, 여기서는 이 정도로 마무리하고자 한다.

8.2.2 host 와 service

아래 명령어를 cron에 등록해 두면 특정 서버로의 연결이 불가능할 경우 이메일을 받을 수 있다. 이 명령은 icmp 메시지 5개를 server1으로 전송하여 한번이라도 응답을 받지 못했을 경우 서버가 다운되었다는 메시지를 메일주소로 보내는 스크립트이다.

```
$ ping -qc 5 server1 || (echo "server1 is down" | mail dude@domain.org)
```

간단하면서도 강력한 모니터링 방법이라고 할 수 있다. 하지만, 이 방법에는 약간의 문제가 있다. 관리하는 서버가 40대 있다고 하자. 모니터링 정비와 원격 서버들 사이의 라우터에 고장이 발생했을 경우, 관리자는 40개의 중복된 메일을 한꺼번에 받게 된다. 또, 메일을 받을 사람이 늘어나면, 메일링 리스트를 관리해야 한다. 모니터링 항목에 따라 그룹핑을 해야 할 경우에는, 두 개이상의 메일링 리스트에 포함되어 중복된 메일을 수신하는 사람이 생길 것이다. nagios는 바로 이런 문제를 해결하는 방법을 제시하며, 이 과정에서 host와 service의 개념을 정의하였다.

host는 인터넷에 연결된 장비를 가리키며, service는 host에서 제공하는 소프트웨어 대문을 의미한다. 그러므로 하나의 장비에 대해 host는 up/down 의 단일 정보만 존재하지만, service에 대해서는 여러 개의 서비스 check들이 있을 수 있다.

host 와 service의 구분이 필요한 이유는 다음과 같다. host로의 접근이 불가능한 상태에서 그 host의 service에 대한 점검을 진행하지 않으며, host가 up 상태일때만, service에 대한 스케줄링을 수행한다.

이와 같은 계층구조를 host간에 또는 service간에도 설정할 수 있다. host들 간에는 앞으로 언급할 설정화일에서 parents 지시자(directive)를 사용하여 설정하며, 물리적인 계층 구조로 설명할 수 없는 경우에는 의존성 정의 (dependency definitions)를 이용하여 논리적 계층구조를 정의한다.

host 와 service 개념으로 나타내기 어려운 모니터링 대상도 있다. 예를 들어 대학내의 메일 서비스를 예로 들면 메인 메일 서버와 메일을 전송하는 통신 장비들, 사용자용 웹 메일 관리자 등 여러 host와 service들이 하나의 통합 서비스를 구성하는 경우도 있다. 이런 경우를 위해 nagios에서는 host group과 service group이라는 개념을 제시한다. 즉 하나의 service group을 구성하는 host들과 service들을 하나로 묶어서 관리할 수 있는 기법이다.

nagios 웹 화면의 왼쪽 메뉴를 보면 “hosts”와 “services”, “host groups”, “service groups”가 있다. nagios 웹을 활용할 때 제일 많이 볼 내용들이므로 이 시점에서 한 번씩 보면 좋을 것 같다. 물론 아직 아무런 설정도 하지 않아, localhost에 대한 내용만 들어 있지만, 이 곳에 나의 목적상 무엇이 추가될 수 있는지 각자 고민해 보기 바란다.

8.2.3 스케줄링

사용자 입장에서 보면, nagios는 주기적으로 plugin을 수행하면서 exit code에 변화가 발생할 때 사용자에게 알려주는 역할을 한다. 여기서 주기적으로 plugin을 수행할 때 어떤 주기로 진행되는지, 그리고 exit code가 우연히 한번만 변화했을 때가 아니라 몇 번 동안 변화된 값을 유지하는지에 따라 상태 변화가 있는 것으로 간주하는지를 결정하는 방식을 스케줄링(scheduling)에서 처리한다.

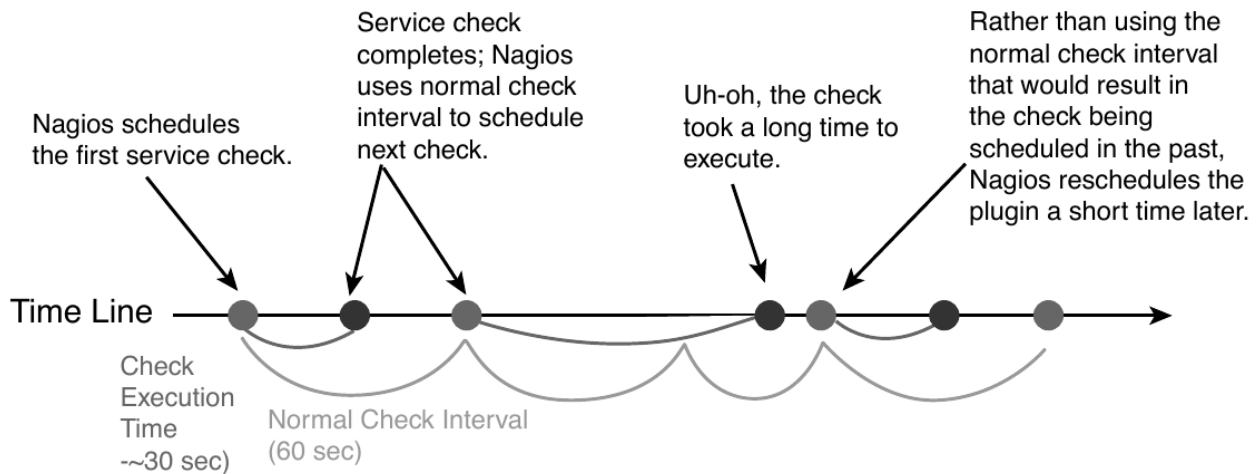
service에 대한 점검은 host가 살아있는 상태에서만 의미를 갖는다. 그러므로 일반적인 상황에서 host에 대한 점검은 service에 대한 plugin에서 오류 (0이 아닌 exit 코드값)를 반환할 때에만 수행된다.

cron에서는 명시적인 날짜와 시간을 이용하여 특정 작업을 수행한다. 하지만, nagios에서는 plugin에서 결과값을 반환할 때까지 기다리는 시간을 정해 스케줄링을 수행한다. 이 지점에서 중요한 두 가지 점이 있을 수 있다.

- plugin을 정해진 시간에 수행할 수 있는가?
- 정해진 시간 내에 plugin의 수행을 완료할 수 있는가?

첫 번째 지적에 대해, 정상적인 또는 시스템에 여유가 있는 상황에서는 정해진 시간에 plugin을 실행할 수 있겠지만, 관리하는 모니터링의 대상이 많아질수록 제시간에 실행하지 못하는 상황이 발생할 수 있다. 제때에 실행하지 못했을 경우 다음 스케줄링은 지연되지 않고 원래 실행되어야 할 시간을 기준으로 그 다음 스케줄링 시간에 plugin을 실행하기 위해 시도한다.

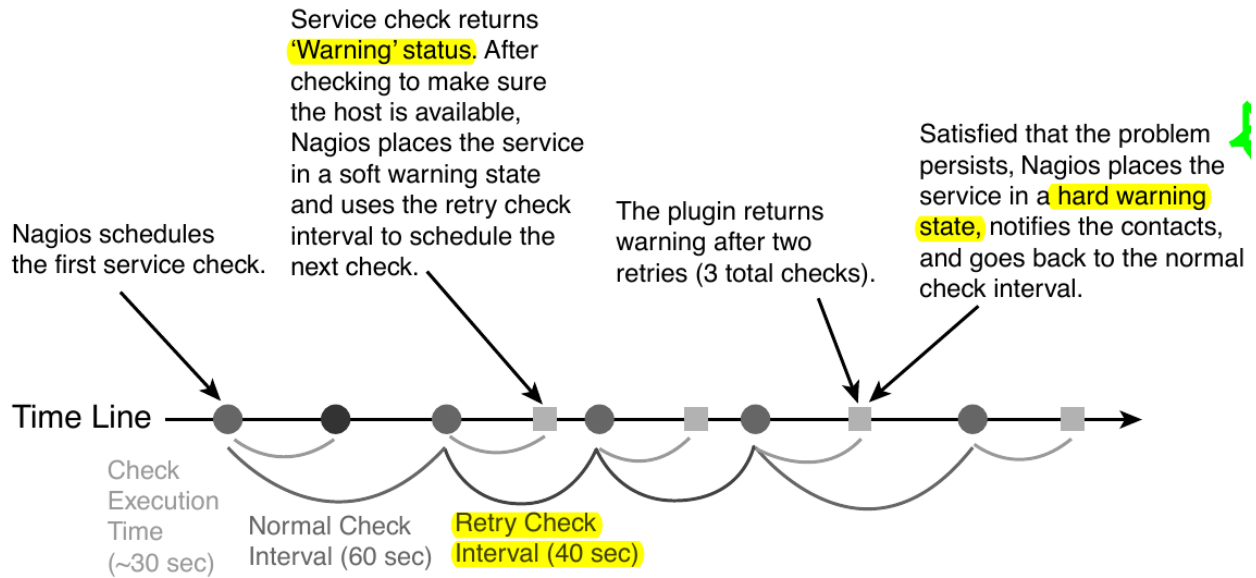
두 번째 경우에서는, 네트워크의 문제 등으로 결과값을 받는데까지 걸리는 시간이 길어질 수 있다. 다음 그림은 두 번째 경우에 대해 잘 설명해 주고 있다.



“Normal check interval”은 설정화일에서 정해지는 값으로 exit 코드 0(OK)을 반환할 경우의 스케줄링 기본 간격을 의미한다. exit 코드의 반환이 늦어지는 경우에는 “Normal check interval”을 무시하고 결과값을 받은 잠시 후에 plugin의 수행을 시도한다.

“Normal check interval”의 반대되는 개념으로 “retry check interval”이 존재한다. 0 이외의 exit 코드를 반환한 경우 해당 오류가 지속되는지를 검사하기 위한 간격이다. 정해진 횟수 (max check attempts)만큼 plugin 실행 오류가 지속될 경우 관리자에게 메일을 전송한다. “Max check attempts”에는 최초의 오류를 감지한 건도 포함한다. 그러므로 “max check attempts”를 1로 설정하면, 재시도 없이 바로 notification을 발생시킨다.

아래 그림은 “Max check attempts”를 3으로 설정한 환경에서 plugin 실행 오류 발생시 check interval의 변화를 나타낸다.



재시도(retry)의 단계에서는 아직 완전한 상태변화를 인정하지 않는다. 이 상태를 soft states 라고 말하며, 이는 soft error states 와 soft recovery states 로 나눌수 있다. 예상할 수 있는 바와 같이 OK상태에서 그 이외에 상태로 변화하는 상황을 soft error states 라고 하며, 반대의 경우는 soft recovery states 라고 한다. 상태가 확정되면 hard states 라고 부른다.

8.2.4 알림 (Notification)

이메일 에이전트인 postfix에 대한 설정을 마쳤다면, hard states 로 변화가 발생하는 시점에 nagios는 자동으로 알림을 보낸다.

알림에 대한 현재 설정은 웹화면의 “Tactical Overview”에서 확인할 수 있다. 알림과 관련한 문제발생시 유용한 정보를 제공할 것이다.

알림의 상태는 plugin의 exit 코드와 비슷한 듯하며 다르다. host와 service 각각의 알림 상태는 아래 표와 같다.

Host States	Service States
Unreachable (u)	Unknown (u)
Down (d)	Critical (c)
Recovered (r)	Warning (w)
Flapping (f)	Recovered (r)
	Flapping (f)

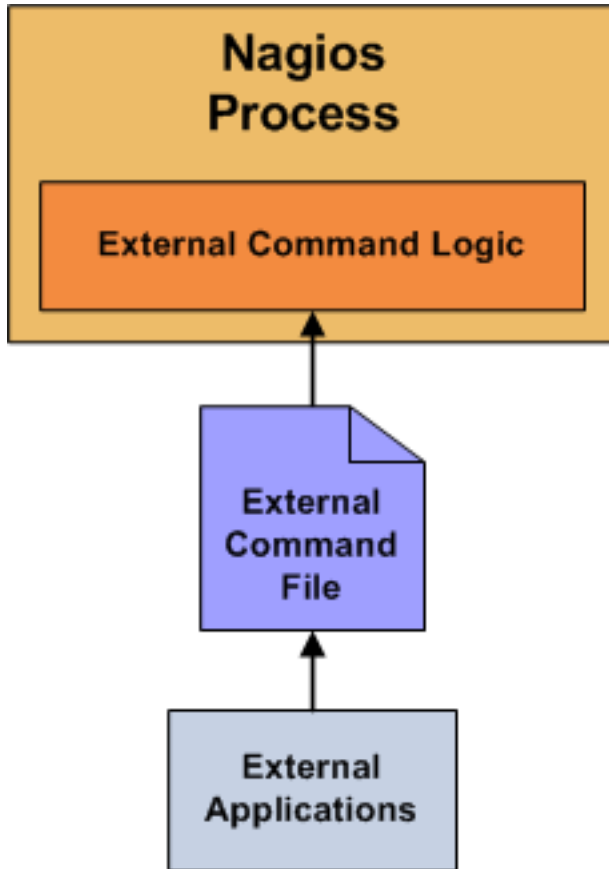
사용자는 nagios 설정시 위 표중에서 어떤 상태가 발생했을 때 알림을 받을지 결정할 수 있다. Flapping은 상태 변화가 빈번하게 발생할 때 반환되는 상태값이고, recovered는 비정상상태에서 동작상태로 복귀된 상태를 나타낸다.

8.2.5 External commands

Note: 이 절의 내용은 http://nagios.sourceforge.net/docs/3_0/extcommands.html 을 참고하여 작성하였다.

nagios는 기본적으로 nagios에서 명령어를 실행하는 주도권을 가지고 응답을 받아 모니터링을 수행한다. 하지만, 이러한 주기적인 방식이 아닌 외부에서 발생한 간헐적인 이벤트들을 처리하기 위한 방법을 제공한다. 외부 명령어 (external command)에는 cgi 명령어들도 포함된다.

아래 그림을 보면, 외부 명령어에서 External Command File 에 결과를 기록하면 nagios의 External Command Logic 에서 주기적으로 이 결과를 읽어 모니터링을 수행한다.



External Command File에는 아래와 같은 내용이 기록된다. 외부에서 기록이 가능해야 하므로 이 파일의 권한 설정을 기록 가능으로 해 줘야 한다.

```
[time] command_id;command_arguments
```

위에서 time은 unix의 timestamp로 ubuntu에서는 `date "+%s"` 명령어로 현재의 timestamp를 얻을 수 있다.

passive checks는 외부 명령어의 한 형태로 아래와 같은 형식을 갖는다.

```
[<timestamp>] PROCESS_SERVICE_CHECK_RESULT;<host_name>;<svc_description>;<return_code>  
↪;<plugin_output>
```

여기서 `PROCESS_SERVICE_CHECK_RESULT` 는 `command_id` 이며, 이후의 나머지 값들은 `command_arguments` 에 해당된다.

8.2.6 Event handler

Event handler는 host나 service의 상태가 변화할 때 실행되는 명령어이다. Event handler를 이용하면, 상태가 변하는 이벤트마다 어떠한 동작을 수행할 지를 지정할 수 있다.

8.2.7 Performance data

nagios에서는 명령어의 실행 결과로 exit code와 상태정보를 나타내는 문자열을 반환한다고 설명하였다. 이에 추가적인 옵션으로 performance data를 반환할 수 있다. check_ping plugin인을 기준으로 아래와 같이 출력의 경우, | 이후의 값들이 performance data이다.

```
PING ok - Packet loss = 0%, RTA = 0.80 ms | percent_packet_loss=0, rta=0.80
```

performance data를 처리하여 데이터의 시각화에 사용할 수 있다.

Note: 더 자세한 정보를 원한다면 http://nagios.sourceforge.net/docs/3_0/perfdata.html 을 참고하라.

8.3 nagios 설정

nagios의 설정 파일과 plugin들이 위치한 곳은 아래와 같다.

Location	File types
/etc/nagios3/	contains configuration files for the operation of the nagios daemon, CGI files, hosts, etc.
/etc/nagios-plugins/	houses configuration files for the service checks.
/etc/nagios/	on the remote host contains the nagios-nrpe-server configuration files.
/usr/lib/nagios/plugins/	where the check binaries are stored. To see the options of a check use the -h option.

Note: 위 표의 내용은 <https://help.ubuntu.com/13.04/serverguide/nagios.html> 을 참고하여 작성하였다.

기술적으로 nagios의 설정파일은 /etc/nagios3/nagios.cfg와 object 설정파일 만 존재하면 동작이 가능하다. 대부분의 경우 object의 타입에 따라 별도의 설정파일로 분리하여 저장하지만, 이를 모두 합쳐 하나의 파일로 저장해도 된다. object들은 아래 표에 정리되어 있다.

Object Name	Description	Recommended Filename
timeperiod	This is the defined block of time that other objects use to determine their operational hours and blackout periods.	timeperiods.cfg
command	Command definitions map macros to external programs. Other objects use commands for many things, such as sending notifications and running service checks.	misccommands.cfg and checkcommands.cfg
contact	This defines a notification target, which is usually a human being.	contacts.cfg
contact-group	Contacts are organized into groups called contactgroups. Objects that send notifications always reference contactgroups and never individual contacts. A contact can be a member of any number of groups.	contactgroups.cfg
host	Hosts are physical entities (or the virtual representation of physical entities if you use virtualizations, such as Xen or VMware), such as servers, routers, or tape drives.	hosts.cfg
service	Hosts provide one or more services. For a web server, httpd or IIS would be a service. The majority of Nagios configuration is made up of service definitions.	services.cfg
host-group	Hosts may belong to any number of user-defined hostgroups. Names and methodology are up to you, for example: servers-with-blue-LEDs or routers-my-boss-refuses-to-upgrade.	hostgroups.cfg
service-group	Like hosts, services may belong to any number of user-defined groups. Servicegroups are a feature unique to Nagios 2.0 and above.	servicegroups.cfg
host-dependency	Dependencies filter out checks and notifications for objects, based on the status of other objects. Be sure you read and understand the section, “Servicegroups,” before using these.	dependencies.cfg
service-dependency	This works the same way the hostdependency does.	dependencies.cfg
hostextended-info	Extendedinfo objects map titles and graphics to host and service objects for the Web interface. These definitions are entirely optional and cosmetic in nature.	hostextinfo.cfg
serviceextended-info	This works the same way the hostextendedinfo does.	serviceextinfo.cfg

Note: ubuntu의 경우 위의 표에 명시된 추천 파일 이름의 규칙을 따르지 않는다. host와 service object의 경우, /etc/nagios3/conf.d/localhost_nagios2.cfg 파일 내에 기술되어 있으며, 반드시 해당 파일을 열어 object를 어떤 방식으로 기술하는지 살펴보라. nagios.cfg는 /etc/nagios3 아래에 위치한다.

nagios.cfg는 고정된 파일 이름으로 다른 이름으로 변경할 수 없다. 하지만, 나머지 object 설정 파일들은 이름을 변경하고 nagios.cfg에 변경된 내용을 반영해주면 된다.

Note: 웹 인터페이스를 사용할 경우 /etc/nagios3/cgi.cfg 파일을 필요로 한다. 이 파일은 nagios.cfg와 유사한 방식으로 설정값을 지정하며, object 설정 파일의 기술방식과는 다르다.

object 설정 파일은 아래와 같은 방식으로 기술된다. 한 파일내에 다수의 object들을 기술할 수 있다.

```

define object{
    variable      value(s)
    variable      value(s)
    ...
    variable      value(s)
}

example)
#A comment about myHost
define host{
    host_name      myHost
    alias          My Favorite Host
    address        192.168.1.254
    parents        myotherhost
    check_command  check-host-alive
    max_check_attempts 5
    contact_groups admins
    notification_interval 30
    notification_period 24x7
    notification_options d,u,r
}

```

8.3.1 nagios.cfg

nagios.cfg 화일은 nagios 데몬 구동을 위해 반드시 필요한 화일로 ubuntu에서 기본적으로 제공하는 화일에서 필요에 따라 수정을 하면 된다. 주로 수정하는 내용은 크게 두 가지로 object 설정화일들의 위치를 지정하는 부분과 check_external_commands 지시자(directive)의 내용을 변경할 때이다. object 설정화일의 위치를 변경하는 지시자는 개별 화일의 위치를 지정하는 cfg_file 지시자와, 폴더 아래 .cfg로 끝나는 이름을 갖는 모든 파일을 object 설정화일로 인식하는 cfg_dir 지시자를 사용할 수 있다. ubuntu에서는 /etc/nagios3/conf.d 을 cfg_dir에 지정해 사용한다.

```

cfg_dir=/etc/nagios3/conf.d

# You can specify individual object config files as shown below:
#cfg_file=/etc/nagios3/objects/commands.cfg
#cfg_file=/etc/nagios3/objects/contacts.cfg
#cfg_file=/etc/nagios3/objects/timeperiods.cfg
#cfg_file=/etc/nagios3/objects/templates.cfg

```

nagios에서 우선권을 가지고 검사하는 방식이 아닌, 외부에서 발생한 이벤트를 nagios에게 알리기 위해서는 외부 명령어(external command)에 대한 검사가 가능하도록 해야 한다. 이를 위해 다음과 같이 설정을 변경해야 한다.

```
check_external_commands=1
```

다음으로 중요한 지시자들을 간략히 살펴보고자 한다. 각 설정화일에도 해당 지시자의 의미를 설명한 주석이 있으니 참고하기 바란다.

[Global enabler]

Name	Description
execute_service_checks	Setting this to 0 turns off service checks program-wide. Defaults to 1 (on).
accept_passive_service_checks	Setting this to 0 turns off passive service checks. Defaults to 1 (on).
execute_host_checks	This enables/disables host checks. Defaults to 1 (on).
accept_passive_host_checks	This enables/disables checks of hosts. Defaults to 1 (on).
enable_notifications	This setting controls whether Nagios will send notifications. Defaults to 1 (on).
enable_event_handlers	Event handlers may be globally enabled or disabled.
process_performance_data	This determines whether Nagios will check for and handle performance data from plugins. Defaults to 0 (off).

[Global time-out values]

Objective Name	Description
service_check_timeout	The length of time Nagios waits for a service check plugin to return its status. Defaults to 60 seconds.
host_check_timeout	The length of time Nagios waits for a host check plugin to return its status. Defaults to 60 seconds.
event_handler_timeout	The length of time Nagios waits for an event handler to finish execution. Defaults to 30 seconds.
notification_timeout	The length of time Nagios allows a notification command to run. Defaults to 30 seconds
perfdata_timeout	The length of time Nagios allows a perfdata handler to run. Defaults to 5 seconds.

설정을 마치면 아래 명령으로 nagios를 재 실행하여 변경된 설정을 적용하자.

```
$ sudo service nagios3 restart
```

8.3.2 Templates

8.4 email notification

/etc/nagios3/conf.d/contacts_nagios2.cfg 에서

CHAPTER 9

Visualization

9.1 rrd

9.2 d3.js

9.3 jarmon