smekal.at :: IT Consulting

# Monitoring
# The Internet of Things

## „Überwachung von Smart-Home Systemen mittels Nagios“

Linuxwochen 2013 - Goesta Smekal

smekal.at :: Ihr Open Source Systemintegrator
http://smekal.at :: office@smekal.at

# The Internet of Things

- origin in the late 1990s, invented for tracking and logistics
  - MIT AutoID Lab http://autoid.mit.edu/cs/
  - Fraunhofer Institute for Material Flow and Logistics http://internet-der-dinge.de/en.html
- later also used for general machine to machine communication
  - European Commission: IoT@Work http://www.iot-at-work.eu/
- currently generally interconnected consumer devices and „Smart Home“ systems

# Smart Home

- sensors

  temperature, humidity, light motion, leakage, ...

- actuators

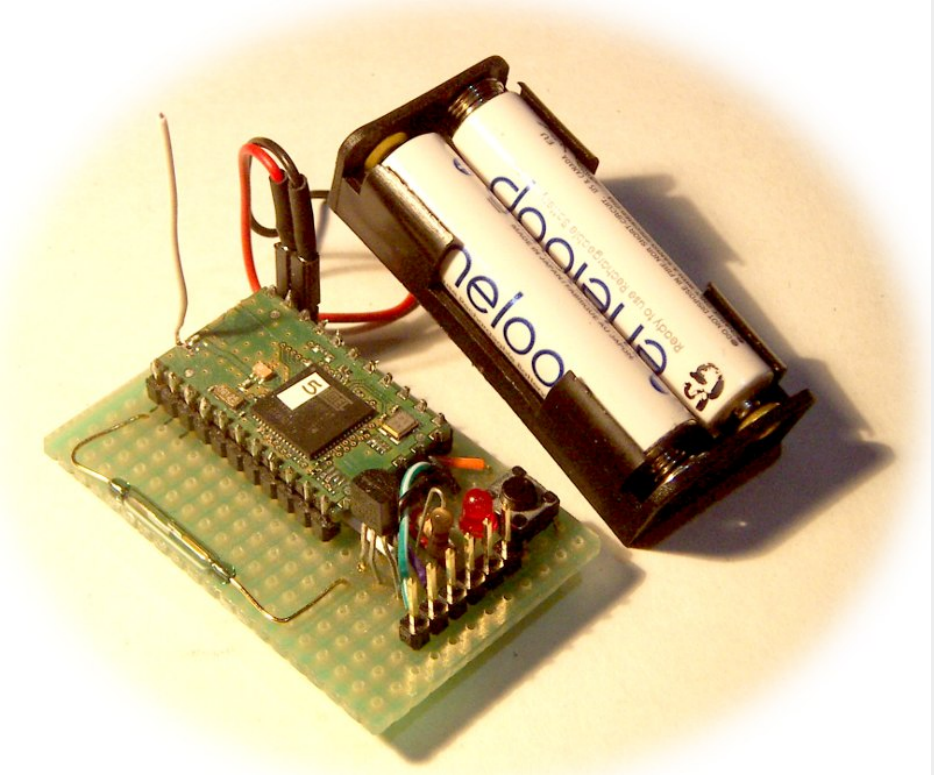
  lights, power outlet, magnetic valve, alarm, ...

- commodity

  central control unit, remote access, automation

- security

  alarms (fire, water, unauthorized access, ...)



# Protocols

„The protocol wars are over“* - are they?

- Transport: Zigbee, zWave, FS20, 6LoWPAN
- Application: KNX/Instabus, Modbus, Lonworks, OPC ...

nearly every commercial solution has its own “standards” - they do not inter operate

most of them are proprietary, including non published “security” features

* (Craig Hunt, „TCP/IP Administration“, O´Reilly 1992)

# Smart-SARAH Definition

opensourcedomotics

Smart-SARAH is a proposed standard for home automation systems, defined by the OpenSourceDomotics Group.

It covers all aspects from the user interface down to the low level hardware layout of example devices. The entire design is based on freely available standards:

- OpenSource Software for firmware, low level interfaces, middleware and user interface
- OpenHardware making reference implementations freely available
- Internet standard protocols regulate communication

# Smart-SARAH by Layer

opensourcedomotics

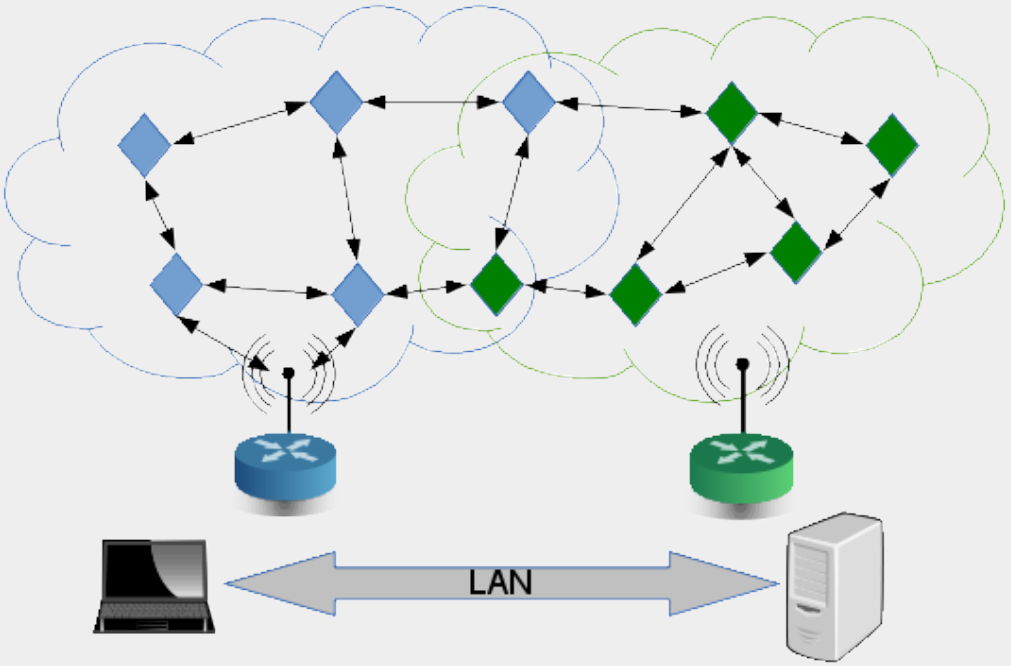
| OSI Layer | Smart-SARAH | Comment |
|---|---|---|
| 7 - application | KnxWeb | Web based GUI |
| 6 - presentation | Linknx, OpenHAB, Lielas | Home Automation Server |
| 5 - session | CoAP | Restful services |
| 4 - transport | UDP, ICMPv6, TCP | Low overhead |
| 3 - network | IPv6 | End to end communication, including encryprion/authentication |
| 2 - data link | 6LoWPAN<br>IEEE 802.15.4 | Low power radio |
| 1 - physical | IEEE 802.15.4 | 2.4 GHz wireless communication |

# Protocols :: 6LoWPAN

- IPv6 Low power Personal Area Network (RFC 4919)
- started as successor if Zigbee (aka “Zigbee 2.0”)
- based on IEEE 802.15.4
- operates both wireless (868 MHz, 2.4 GHz) and wired (industrial powerline)
- features: ad hoc networks, header compression, encryption ...
- inherits end to end connectivity and security from IPv6

# Routing

- „old style“ networks
  - classic networks have simple parent/child relationships
  - routing is semi-static
  - protocols are well established (RIP, IS-IS, OSPF, BGP ...)
- Internet of Things
  - mesh networks are complex
  - routes are dynamic, depend on nodes
  - protocols are evolving (RPL)



# Routing :: RPL

IPv6 Routing Protocol for Low-Power and Lossy Networks (RFC 6550)

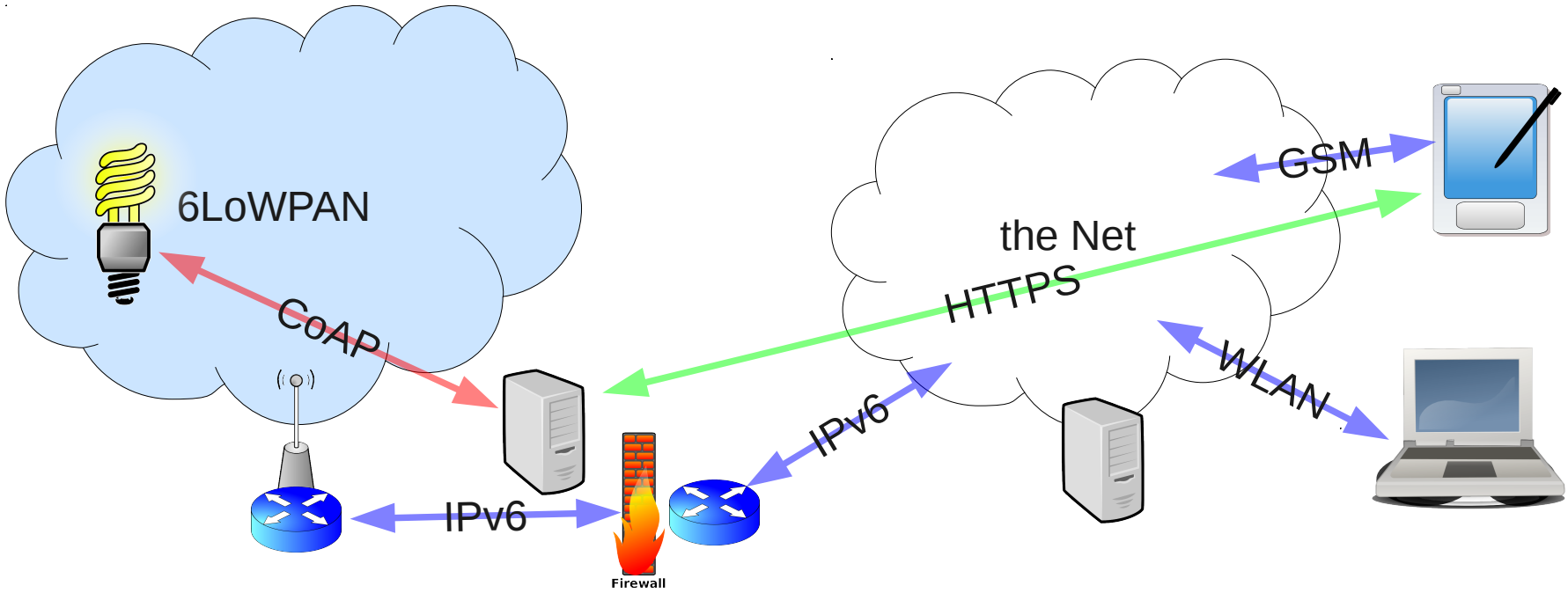
- optimized for small devices and low power consumption
- deals with high loss rates, low data rates and frequent changes in topology
- based on distance vector routing
- nodes broadcast routing information
- nodes may sleep quite long

# Protocols :: CoAP

Constrained Application Protocol
(IETF Draft http://datatracker.ietf.org/wg/core/)

- REST-full services via UDP
  (GET, PUT, PASTE, DELETE)
- URI ↔ resource matching
  (coap://node3.6lowpan.smekal.at:5683/sensors/button)
- resources can be “observed”
- small message size to fit a single network packet
  (header fields 1-4 bit)
- supports encryption/authentication

# Internet of Things




a simple example

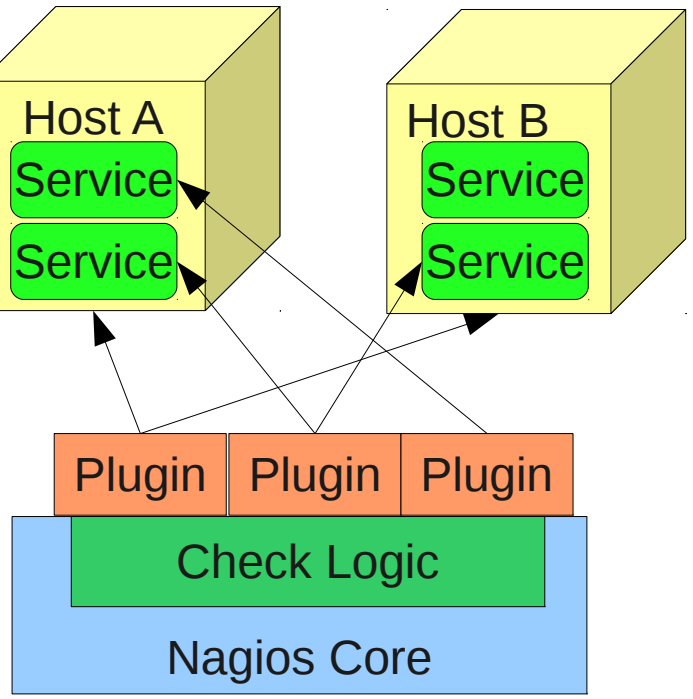
# Monitoring

- collecting indicators from devices/the environment
- alerting in case of „failure“
  - threshold comparison
  - availability
  - integrity

- „System Monitoring“ usually means „IT-System Monitoring“
  - Server(farm)s
  - Datacenters
  - Networks
  - Services and business processes
- OpenSource Monitoring usually means Nagios™ and derivatives

# Monitoring :: Nagios

- Nagios core deals with scheduling checks, collecting results and triggering actions
- Plugins do actual checks, alerts and other actions
- 4 states: ok, warning, critical, unknown
- flexible design
- light weight core
- easy to extend




Nagios architecture

# Nagios :: Plugins

- standard plugins:

  check_apt, check_bgpstate, check_breeze, check_by_ssh, check_clamd, check_cluster, check_dhcp, check_dig, check_disk, check_disk_smb, check_dns, check_dummy, check_file_age, check_flexlm, check_fping, check_ftp, check_game, check_host, check_hpjd, check_http, check_icmp, check_ide_smart, check_ifoperstatus, check_ifstatus, check_imap, check_ircd, check_jabber, check_ldap, check_ldap_monitor.pl, check_ldaps, check_linux_raid, check_load, check_log, check_mailq, check_mem.pl, check_mrtg, check_mrtgtraf, check_mysql, check_mysql_query, check_nagios, check_nntp, check_nntps, check_nt, check_ntp, check_ntp_peer, check_ntp_time, check_nwstat, check_oracle, check_overcr, check_pgsql, check_ping, check_pop, check_procs, check_radius, check_real, check_rpc, check_rta_multi, check_sensors, check_simap, check_smtp, check_snmp, check_spop, check_ssh, check_ssmtp, check_swap, check_tcp, check_time, check_udp, check_ups, check_users, check_wave

- example:

```
check_icmp -H smekal.at -w 50,10 -c 100,20
OK - smekal.at: rta 0.269ms, lost 0%
```

# Monitoring :: Thresholds

- Nagios-Plugin guidelines offer complex threshold definition
- available as libraries (C, Perl, ...)
- don't try to implement it yourself!

  but: networks do not know about negative values – the environment does

  example: ambient temperature
  - datacenter: -w 25 -c 30
  - home: -w 18:25 -c 15:30
  - outside: -w 3:25 -c -3:35

| Range | Alert |
|---|---|
| 10 | <0 or >10<br>(outside {0..10}) |
| 10: | <10<br>(outside {10..∞}) |
| ~:10 | >10<br>(outside {-∞..10}) |
| 10:20 | <10 or >20<br>(outside {10..20}) |
| @10:20 | ≥10 and ≤20<br>(inside {10..20}) |

# Monitoring :: Why?

Why bother monitoring a smart-home system?

- are all my light switches still working?
- are they still my light switches?
- is there a leakage in the basement?
- is there still a leakage sensor in the basement?
- is the central management system up and operational?

# Monitoring :: Availability

- Host checked by ICMP Echo request
- reachability by parent ↔ child information (network outage)
- evaluate round trip time, packet loss
- lossy networks are inherently ... lossy
- routes do change eventually – patchwork family

- Service status by special plugins
- off the shelf plugins available for most standard internet services HTTP, ICMP, SNMP, SSH, DNS, SMTP ...
- example metrics:
  - valid response
  - response time
  - content

# Monitoring :: RPL

- check for routes and neighbors in RPL routing table:

```
Neighbors

fe80::ff:fe00:3
fe80::221:2eff:ff00:1ee2
fe80::ff:fe00:5
fe80::ff:fe00:6
2001:858:5:b06::ff:fe00:11

Routes

2001:858:5:b06::ff:fe00:5/128 (via fe80::ff:fe00:5) 16711374s
2001:858:5:b06::ff:fe00:6/128 (via fe80::ff:fe00:6) 16711423s
2001:858:5:b06::ff:fe00:12/128 (via fe80::ff:fe00:12) 15320084s
2001:858:5:b06::ff:fe00:3/128 (via fe80::ff:fe00:3) 15786104s
2001:858:5:b06::ff:fe00:11/128 (via fe80::ff:fe00:6) 16709400s
2001:858:5:b06:221:2eff:ff00:1ee2/128 (via fe80::221:2eff:ff00:1ee2) 16711322s
```

# Monitoring :: CoAP

- no Nagios Plugin (yet)
- no working Perl module (yet?)
- libcoap provides client application, written in C
- Perl based wrapper

```
eval {
  alarm($plugin->opts->timeout);
  # set timeout trigger
  open (COAP, "coap-client -m get
coap://".$host.":5683/".$plugin-
>opts->uri." 2>/dev/null |")
      or $plugin->nagios_die ("Cannot
contact CoAP host:\n$!");
  while (<COAP>) {
    next unless (/([-0-9\.]*)/);
    $coap_value = $1 / $plugin->opts-
>divider;
    if ( $plugin->opts->kelvin
{ $nagios_value = $coap_value +
273.15; }
    else { $nagios_value =
$coap_value; }
  }
  alarm(0);
};
```

# Monitoring :: Integrity

- check node identity (IPSec AH)
- check data integrity (IPSec ESP)
- check for valid response (ex.: DS18S20 thermal sensor responds “85°C” in case of failure, 127°C on low power)
- include test routines into application logic

# Monitoring :: Conclusions

- possible smart home related metrics include:
- Network:
  - availability of gateways and nodes
  - round trip times
  - contents of routing tables
  - protocol specific tests (i.e.: CoAP URI “/.well-known/core”)
- Sensors:
  - specific sensor readings (temperature, humidity, contact closure ...)
  - general node parameters (battery level, firmware release, CPU temperature, ...)
- Application:
  - response of UI (response time, valid response, test cases, ...)

# Conclusions continued ...

- mind your thresholds
  - round trips can vary across magnitudes of 10
  - use Kelvin for temperature monitoring (avoids negative values)
- relax – mesh networks repair themselves ... eventually
- be prepared to customize your plugins