sDOMO – A Simple Protocol for Home Automation and Robotic Systems

Marcel-Titus Marginean and Chao Lu

Abstract: *Today's market for domotic systems it is dominated by proprietary communication protocols incompatible with each other. This not only prevents engineers to build highly integrated home automation and domestic robotic systems, but keeps prices artificially high due to lack of competition on the market since the customers have to restrict their choices to components compatible with the system already deployed. In this paper, we propose a communication protocol for fully integrated domotic systems running on top of house network, protocol which is simple enough to accommodate as peers micro-controller based sensors but scalable enough to handle a house wide distributed computer vision system for domestic robot assisted living. Based on a house centered architecture, the protocol is putting emphasis on privacy and protection of the residents.*

Introduction.

In our paper [1] we introduced a distributed computer vision system architecture for domestic robots operation assisted by both external (house mounted) and local(robot) video processing systems. One of the obstacles encountered during development has been the lack of a suitable communication protocol on top of which to implement the proposed architecture. After exploring a few options we decided to implement our own in order to be optimized for this kind of deep home automation.

Our vision regards the whole intelligent home as a integrated domotic system working together to keep the inhabitants comfortable, the robot being just a mobile component into the smart house of the future. A direct result of this vision was the fact that the protocol should allow various sensors or actuators with small computing power, like micro-controllers driven thermostats or door locks to integrate effortlessly into the system therefore eliminating a couple of other proposed protocols that employs higher overhead XML or JSON messages sent over full blown SOAP over HTTP. The Simple Domotic Protocol (sDOMO) employs binary packed messages sent over raw UDP while using the power of XML exclusively at the higher powered House Hub, House Intelligence Unit, Administrative Console and by the developer tools used to generate code to parse/build the protocol messages.

This paper presents an introduction of the sDOMO protocol and it is not an exhaustive protocol design document. To fully define the protocol, offer downloads, tutorials and demos a companion site of this paper is being lunched at:

http://www.mezonix.com/sDOMO.html

A similar architecture with ours having closely related security goals has been presented in [2], however their reliance on public key infrastructure for devices will not permit smaller devices to connect directly with the rest of the network, therefore the system must employ Room Bridge acting as protocol adapters. While bridging architecture it is also supported by sDOMO for devices that are unable to speak the protocol, our work reduced the protocol requirements to the basics, allowing much smaller devices to be part of the network directly.

To solve the interoperability problem between otherwise incompatible platforms in [3] the idea of a SOAP based middle-ware has been proposed. A similar SOAP based middle-ware has been presented in [7]. The high overhead and verbosity associated with SOAP however would require that any small sensor or device to depend on a more powerful adapter.

In [8] an microcontroller based module has been presented but it relied on external serial-to-lan converter and no overall domotic network architecture has been attempted. An IP based home automation system has been presented in [9] but the reliance on ssh excluded microcontroller based devices from working as peers.

A similar idea with our Home Intelligence Unit (HIU) has been presented in [4] by Kao and Yuan where they developed a more elaborate and complex set of meta-rules than our HIU. We may be looking more in depth at their ideas for our future development of HIU.

The idea of a framework for developing domotic systems by following a model driven approach has been introduced in [5] while [6] presented a state of the art report on home automation technologies.

Architecture overview

The sDOMO protocol proposed a hybrid communication structure with a House Hub acting both as a central registry for all the devices into the house and providing reliable and secure communication between devices; and an optional direct device to device communication.

The sDOMO preferred method of communication is for devices to communicate with each other by messages routed via the House Hub. In this model a Device exchange Packets with the Hub, most of the packets carrying Messages intended for other Devices. The Hub receives, validate, authenticity and unencrypt packets if necessary, assemble their payload into messages route them toward the appropriate receiver and deliver them to the target in signed and eventually encrypted packets if the security requirements specify it.

To support direct communication sDOMO allows for each device to advertise to other components on the network their own external/proprietary interfaces in this case sDOMO hub acting exclusively as a rendezvous server, allowing other devices to discover which peers are onto network and how to talk with them. This feature it is implemented in order to allow vendors a seamless transition from their current protocols to sDOMO, or to support special high throughput applications like High Definition Video Streaming for which the House Hub would just introduce unnecessary delays.

It is possible for a house to contain more than one House Hub for different applications. For example a Hub will handle the video security where the high bandwidth video is being processed while all the rest of the system is being handled by a separate Hub to separate low power microcontroller based sensors from the high traffic. An Inter-Hub extension of the sDOMO protocol is being developed to allow multiple hubs to transparently act as a single hub from the devices point of view. From now on when we refer to the House Hub we are using the words to mean either a single hub network architecture or a multi-hub architecture having inter-hub protocol enabled.

A typical house automation architecture as presented in the figure 1 consists on a House Hub, a set of devices connected to the hub, and House Intelligence Unit and Internet Gatweay.

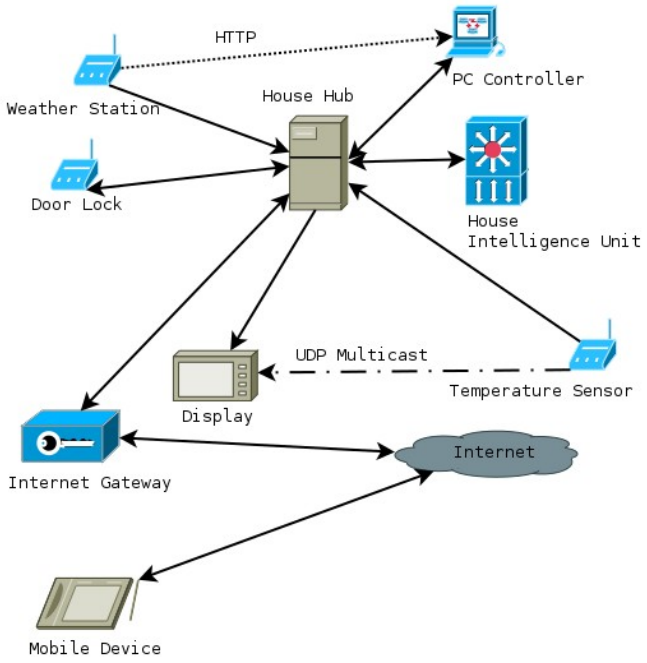


Figure 1. Hybrid communication architecture

In a simple example from Fig. 1 after all of the nodes checked in with the Hub the temperature Display downloads from the Hub registry the parameters required to listen directly to the multicast sent by the Temperature Sensor. The PC acting as console is also receiving direct data via HTTP from the Weather Station while is using the Hub in order to send secured commands to the Door Lock. Some devices like the Door Lock for example will only accept signed commands from the Hub while other devices sending sensitive data will use the Hub exclusively in order to send encrypted messages.

Internet connectivity is allowed exclusively via an Internet Gateway which isolates the sDOMO network from the cloud. The Internet Gateway provides encrypted communication with the outside world and require authentication of the users requesting access. The access to information is controlled via Access Control Lists based on the user. In order to allow remote sensors and sDOMO Devices, the Internet Gateway can also provide sDOMO tunneling toward a specified IP address. In order to open the sDOMO tunnel, the remote device must first log in via HTTPS or SSH to open the tunnel and check-in periodically to keep it open. Supplementary, regular web application can run on the Gateway allowing controlled access via a regular browser to specific functionality.

There is no restriction for a sDOMO device to have other interfaces incompatible with sDOMO. For example a Power Meter can have a totally proprietary interface to be accessible by the power company while also exporting a sDOMO device in order to inform the household monitoring software about their energy consumption. The sDOMO specifications are responsible for defining a house-centrist architecture and to not interfere with proprietary interfaces of leased devices.

Definitions of terms

A ***Device*** is the generic name for a node into a domotic network. It can be an embedded device like a micro-controller based thermometer having Ethernet connectivity, a serial device speaking sDOMO over SLIP or PPP and connected to a Serial-to-Lan converter or a program running on a laptop, or tablet used to display data or to manage the sDOMO network. Bluetooth and ZigBee are also strong candidates to be added to sDOMO supported protocols beside UDP/IP.

The network associated to a particular house is called a **Domain** and is defined by a **Domain Name**. A Network can assign to a Device a **Given Name** for human convenience if it need to be refereed later. If no Given Name is assigned the default device name will be it's DUID.

A Device is identified by the ***Device Unique ID (DUID)*** which is a 20 bytes alphanumeric string where the first character determine how the unique number has been achieved. A DUID starting with E is followed by 12 Hex-digits representing the Ethernet ID of the device and optional by another 7 characters to distinguish, for example, between different programs running on the same computer. The strings starting with an X are reserved for experimental devices or hobbyists and they can choose any remaining 19 characters with the only restriction to be unique on the LAN. It is important to note that a DUID is assigned for a particular device and all the settings, configurations and access rights are indexed by the DUID. If for example a Console Monitoring program have it DUID changed, all the devices it previously had access will be now inaccessible until new rights are granted again.

Once a Device joined a Domotic Network by being accepted to a **Hub**, the Hub issue a ***Device Session ID (DID)*** and a ***Session Key*** used to authenticate messages and eventually encrypt the communication between the Device and Hub. The Hub may or may not keep DID in a persistent storage to reuse it across multiple sessions, therefore the Device must accept whatever DID is assigned by the server at every connect.

The type of devices are defined by the ***Spec ID***. This is a string of up to 128 characters that define an URL from where the Spec File can be retrieved. For example if a Device advertises a Spec ID of:

mezonix.com/sDOMO/devs/dev1

that means that a XML file defining the device can be downloaded either from:
mezonix.com/sDOMO/devs/dev1/device.spec or from
www. mezonix.com/sDOMO/devs/dev1/device.spec
either by HTTP or HTTPS.

A **Spec File** is an XML file describing the device,

linking to user and developer documentation, defining connection requirements and the interfaces implemented by the software objects implemented in the device. For example a indoor video camera can specify that it transmits sensitive data therefore the Hub may decide to require encryption. What types of encryption are available for a given device and their strength and order of preference is also specified in the Spec File. For privacy enforcement Spec File also contains a list of the interfaces that this device must connect as client in order to perform it tasks.

From the software architecture point of view a Device is composed of one of more (up to 255) **Objects** identified by a contiguous number between 0 to NoOfObjects. Each Object implements a Interface whose Definition File if referred by the Spec File. The **Interface Definition File (IDF)** is another XML file defining the list of all Messages understood and/or sent by a given Object, the way they can be associated to provide method calls and the events that triggers them. The Interface Definition file is used by the developers tool to generate code for a Client that connects to an Object or for the Stub that implement the respective functionality. It is also used by the Hub for privacy enforcement and by the House Intelligence Unit for alerts and adaptive automation.

Each Device have a **Device Main Key** stored inside the Device firmware which is a key used by the Hub to encrypt the Session Key when it is delivered to the Device. A Hub that will accept a particular device must know the Device Main Key. The standard sDOMO way for this operation is for a **Console Application** to upload the key and DUID either typed in by the user either as a result of a scanning a QRCode printed on the Device or a RFID tag attached to it. Figure 2 shows a sDOMO QRCode encoding the DUID: E79E51B96A9FC-0AF211 and Device Main Key : 45687f12239540b4c5473b95c1904193e0f4fd28ce26717c349654080ca7aa65

Fig. 1: QRCode for a sDOMO device

Two sDOMO Devices can communicate either via Messages routed by the Hub either talking directly to each other bypassing the Hub in what is called **Streaming**. **Messages** are blocks of data to up to 4GB in size (subject to Hub / Device memory limitations) transmitted between Devices. Messages are carried as payload in Packets. A **Packet** is a chunk of data capable to be carried into a single UDP Datagram over the local LAN without fragmentation. Packets are sent between a Device and the Hub only, they are the backbone of connectivity and they can carry fragments of Messages as their payload. Messages are routed by the Hub from the source to destination Device. There are two different types of Messages: Notifications and Direct Messages.

A **Notification** is a sDOMO message sent by a device and received by any number of devices that subscribed for it prior to the moment it was sent. The Hub will copy the Notification and send it to all subscribers.

A **Direct Message** is a one to one sDOMO message sent by a Object part of a Device specifically toward another Object of a connected Device.

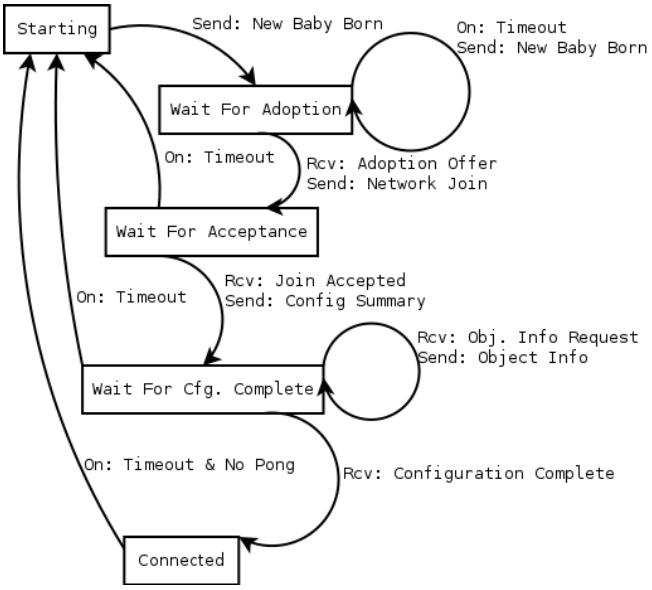
Beside carrying Messages and Acknowledgments for them there are various packets for device discovery, configuration, connection maintenance etc. The full definition of the packets structure and a C++ library for working with them is provided on the companion website for this paper.

A **Virtual Device** it is a set of services provided by the Hub to the connected devices via Direct Messages and Notifications. In order for a piece of code from the Hub to talk via Messages as opposed to packets, the Hub must allocate to it a similar entry as allocated to a real Device. In order to facilitate the process of discovery the Virtual Devices have well known DID's allocated into a reserved range from 1 to 999 range in which only Virtual Devices and **Devices with Special Duties** are allocated. The Virtual Device No. 1 have the object #0 designated as the **DeviceManagerInterface** which is allowing other devices to list the connected devices and their objects and get extended information from it. The **AdministratorServices** object allows for a console program to upload devices keys and edit their parameters. The **AnonymousServices** allows a device with level of trust 0 (without a recognizable Main Key) to log in as a program with special access or administrative rights using a user/password credentials pair or presenting a PKI certificate signed with an authoritative Private Key.

The **House Intelligence Unit** (HIU) it is a special device having "intimate" relationship with the Hub. HIU is mainly a meta-rules processor and is responsible for handling out of ordinary conditions, issuing alerts and perform complex automation tasks coordinating multiple devices based on scripts.

Discovery and Configuration.

A Device starting up will use Multicast to the standardized sDOMO Local Mcast destination (224.68.12.8 port 1881) sending a "NewBabyBorn" packet containing it DUID, Spec ID and previous Domain and Given Name if any. A Hub that matched the Domain (if present) and wants that Device to join it network will send a "AddoptionOffer" packet specifying a adoption bid between 0 and 255. The Hub that sends the highest bid will be selected by the Device to join by sending a "NetworkJoin" packet. The Hub accepting a Device replies with a JoinAcceptedPack which contains the Device new DID and Session Key. The reception of JoinAcceptedPack triggers the Device to send the ConfigurationSummaryPack containing some configuration parameters that can overwrite the default values specified in the Device Spec File. For each Object of the new device the Hub query for that object information and the Device replies with a ObjectInfoPack. Once all the Objects has been configured the Hub sends an Configuration Complete packet which turn the Device in full operational mode. From this moment the device can start communicating with other devices via Notifications or Direct Messages.



After a Device received a Session Key all outgoing packets are signed using that session key and all incoming packets are verified to be valid signed by the Hub with the same Session Key.

A Device whose Device Main Key is unknown by the Hub will receive an adoption offer with a bid of 0. A regular device will ignore any bid with value of 0. However an Administrative Console program or other GUI based device that can prompt the user to type a username and password will join the hub at trust=0 and use Anonymous Services to upload it Main Key encrypted with the user's credentials. Such GUI based application are then used to scan and upload to the Hub the DUID / Key for any new device that will be connected to the network.

A second type of Main Key upload procedure is developed for the usage of sDOMO as a Robotic Systems Building Protocol. This is the case , for example, when an army unit deployed on the battle field receive a new payload for one of their Unmanned Vehicles. Plugging in the payload will trigger a set of credential exchanges between the UAV and the new equipment, they verify each others identity and then the payload will upload it main key to the vehicle's hub. This procedure acts in the following way: the new device is connecting with level of trust 0 and then uses the Anonymous Services to upload a certificate for the device public key and requests the certificate of the Hub for verification. Following a successfully credential validation on the both sides, the device uploads a newly generated Main Key to the system encrypted with Hub's public key and signed with Device's private key. This procedure however is applicable only for devices having non trivial computing power on board since it involves CPU intensive public key cryptography. The devices and the hub must be prepared to handle multiple levels of certificates. As presented in the hypothetical example above, the manufacturer of the vehicle may not know about the manufacturer of the payload and reversal. However, both manufacturers will be able to provide a certificate signed by the army for the key they used to sign the key of the device and vehicle respectively. This dual level of certificates allows full validation since both the vehicle and the payload knows the army public key.

Packets and Messages

A sDOMO Packet is carried on a single UDP Datagram. Each packet contains a header as specified bellow (in 32 bits / line):

0 8 16 32

Packet ID		DID
Packet Number		
Sig Type	Enc Type	Signature Value .....
..... Variable length based on the SigType		

The Packet ID specify the function of this packet. DID is set to 0 for packets that does not have yet assigned a Device ID like NewBabyBorn or Addoption Offer. Once a session is initiated the DID specify the assigned Device ID. Packet number is an unique value incremented at every sent packet, it is used to eliminate eventual duplicates on the network and to prevent reply attacks.

Signature Type specify the HMAC algorithm used sign the packet. No signature =0, SHA1 HMAC=1, SHA256 HMAC=2. Encryption Type specify which encryption algorithm is used for the Packet: 0- No encryption, 1 – XOR, 2-AES128, 3-AES256 are currently defined. The signature value is dependent on the signature type, for no signature the length is zero, i.e. the packet body starts immediately after the encryption type. For SHA1 based HMAC the signature is 20 bytes long and for SHA256 it is 32 bytes. Numeric values are packet in network order i.e. big endian format.

The Packet Body is also a collection of network order fields. Strings are packed as a zero terminates ASCII string, the byte 0 being required at the end of the string. Arrays are prefixed with the size encoded as variable-length code then the specified number of items follows. sDOMO does not sends floating point over the wire and recommends the message designers to use fixed point format represented as integers.

Messages are sent as a sequence of one or more Message Carrier Packets. Each packet contain the start position of the current packet in the final message. Packets arriving in expected order are not ACK-ed until the final packet arrives and then the whole message is therefore ACK-ed with a single ACK packet. However, any packet out of expected order triggers a negative ACK containing the expected sequence. The Hub will receive a whole message and ACK it reception to the sender before delivering it to the intended recipient. It is important to notice that therefore receiving an ACK for a whole message does not mean that the receiver got it, and if a confirmation of delivery is required, the designer of a Interface must implement it at the message level.

Packet Level Security Considerations

After a new connection has been established and a Session Key has been downloaded from the Hub to the Device each sDOMO packet exchanged between a Device

and the Hub is being signed with a Hash based Message Authentication Code (HMAC). The HMAC is calculated with the formula:

HMAC(K,M) = H(K | H(K | M))

where H is the Hash function used, K is the Session Key and M is the message. The operator | specify byte stream concatenation. The Message subject to hashing consists on all the fields in the Packet Header with the exception of the Signature itself and all the fields on the Packet Body. Some of the packets are deemed to be to short to provide enough entropy for the security purposes and therefore a random number is added to their definition in order to increase the entropy of the packet over which the HMAC is calculated.

In order to prevent reply attacks, when the highest PacketNumber of the incoming or outgoing packets is approaching the overflow limit, the Device shall re-send a Network Join packet that will renew the SessionKey and reset the counters for PacketNumber to 0. By making sure that the combination PacketNumber Session Key is unique and by not accepting Packet Numbers lower or equal than the previous received one, the system is protected against reply attacks.

To download a session key a Device is generating a new 64 bit random number CR every time it sends a Network Join packet toward the Hub. The Hub will in turn generate it own 64 bit random number HR and encrypts the Session Key with a Download Key calculated as:

DldKey=SHA1(CR | Device Main Key | HR)

The HR is sent by the Hub toward the Device along with the encrypted key in the Join Accepted packet. The existence of both random numbers insure that the DldKey is unique every time a download happen and this prevents a reply attack in this stage in the connection process when a sequence number is not yet initialized. The fact that the DldKey is unique also allows usage of very small devices that have no encryption capabilities other than a simple XOR. Since the encrypted SessionKey is a random number without any structure to reverse engineer having a size equal with the SHA1 hash and since the Main Key is not known to the attacker the simple XOR with the unique DldKey is an approximation of one-time-pad encryption as long as the same key is not reused too many times to allow a statistic based recovery attack.

While HMAC based signatures are mandated by sDOMO protocol specifications for all devices, the Packet Encryption is not required but just recommended. Sensitive Devices like indoor surveillance cameras or microphones are expected however to encrypt all their communication. The Spec file of a particular Device can specify that a certain type encryption need to be used. It is also important to notice that the XOR encryption it is used exclusively for downloading the session key for small devices and will not be used to encrypt packets. The rationale behind that is the fact that a packet have a predictable structure and a larger size than the key and this can be used for an reverse engineering attack on the encryption key. A direct consequence of this aspect is that really small devices incapable of at least AES128 will have to send data unencrypted, therefore can not be employed for privacy sensitive applications.

Streaming Interfaces as specified above are direct data connections between two Objects from different devices over the network, without the involvement of the Hub in communication process. sDOMO Hub however it is involved in managing the security keys of the system. A Device advertising an object with a “sensitive” interface triggers the Hub to send an Object Session Key for that object. The Object Session Key must be different than the Device Session Key for security purposes and the Hub uploads it encrypted with the Device Session Key. Once an Object have an Object Session Key, it will immediately start to use it to encrypt and decrypt all the messages sent to it via that Streaming Interface. When a Device wants to communicate via Streaming Interface with an Sensitive Object it needs to request it Object Session Key from the Hub. Of course, this is going to be subject of approval by the ACL rights management system into the Hub.

While sDOMO allows for unsecured Streaming Interface, this is a feature that needs to be thought very carefully. For example if an attacker can spoof a thermometer sending data to a Tablet, the worst that can happen is for the residents to get confused. However, if the same thermometer is also connected to the A/C control unit the results can have more serious implications. If a hacker sends to A/C unit a fake high temperature when the house is really cold it can put chilling in overdrive an eventual sleeping diabetic patient unable to properly sense the temperature may go into hypothermia. Therefore, for any control action the secured communication via the Hub or encrypted Streaming Interface is highly recommended.

A unique security feature, as far as we know, in sDOMO protocol is the prevention of device kidnapping by the treat of disclosure of the attacker. If a dishonest neighbor or contractor is solicited to help with the installation of a new device, the installer will have access to the new Device MainKey in order to upload-it into the Hub. If the bad guy installs a Trojan Horse masquerading as a fake hub on the residents laptop there is nothing a cryptographic protocol can do to prevent the fake hub to take control of the device since it knows it MainKey. However, as a deterrent for this kind of situations sDOMO protocol design requires that the Network Join packet sent by a device have to use multicast or broadcast, if the underlying network technology permits it, as opposed to unicast toward the hub intended to be joined. The Join Network pack also contains the Address of the Hub intended to be joined allowing a immediate identification of that computer. The sDOMO specifications also requires for any Hub to log all the Network Join events it receives regardless if they are intended for itself or not. A Hub is also required to inform House Intelligence Unit if it heard on the local network about a join to a hub that is not known to be part of this domain. That is, if a Trojan Horse attempts to hijack a device, this results in immediate disclosure of the Trojan, the event can not not go unnoticed and measures are going to be taken by the House Intelligence Unit.

Message Level Security Considerations

Attacks on a domotic system are not necessarily done by penetrating the network by outlaws but also by the legitimate software or devices the user purposely installed for a certain tasks. We are living into a world where many companies, some of them very large, generate most of their revenue from collection and selling personal information to whomever is willing to pay for it. Some of these companies already entered in the field of Domotics and Robotics. While putting a small-print note into the user agreement may give to the vendor legal rights to snoop on other devices, our approach to privacy does not agree with this practice and sDOMO is designed to take some steps toward protecting user privacy by protocol design.

To aid the users privacy and security sDOMO specifies in Interface Definition Files for each message the Security Level for each message in the interface. Security level 0 are messages which even if sent maliciously to a device have totally benign results while the messages at level 3 must be permitted only by the authorized system administrator. Similarly for outgoing messages, a level 0 means that any information in a message can be made public without any concern, while a level 2 or 3 respectively means that this piece of information must be kept top secret and received only by authorized devices and administrative programs respectively.

Similarly, any Device connected to the system have an associated level of trust from 0 to 3. The level of trust 0, Anonymous, is assigned to devices that connected to the Hub but which do not have a Main Key uploaded into the Hub. Beside Hub's Anonymous Services interface that allow console applications with a user password to log-in, very few devices, if any, are expected to accept level 0 messages.

When a device D1 sends a message to another device D2 , or when D1 submit a request to subscribe to a notification emitted by D2 the message/request get assigned a priority of:

MP=min(Device Trust Level, ACL(D1, D2))

where ACL(D1,D2) is returned from an Access Control List maintained in Hub's Database. If MP >= Security Level then the request is granted otherwise an error packet is sent back to the requester with an permission denied error code. The ACL entries are assigned on a device to device basis so snooping onto the notifications sent by other devices will be denied unless the permission has been explicitly granted to do so.

To facilitate the constructions of ACL's by regular home-owners without training in computer security, sDOMO requires that any device that needs to communicate with another as client must list the interfaces it needs to access. In the Device Spec File, a tag <client> is used to list all the interfaces this device needs to subscribe / send to. The <client> entry must also provide the level of access required, specify if granting this right is mandatory for the intended functionality of the device and a clear text description of the reason why this access is required. The Console Application used to scan a new device will download the spec file first time when the device is connected to the network and will initiate an user friendly dialog for granting the rights. If a particular interface is not listed on the Spec File, no ACL entry will exist for that particular pair and therefore the access will not be possible for anything but Security Level 0.

Of course a dishonest manufacturer can list in the Spec File, as required, all the possible devices it wants to snoop onto; and here came the first line of defense: Shame. It is not required high level security training to understand that a thermostat does not need access to all the indoor cameras and microphones to "regulate the temperature inside the house" as may be stated on <reason> tag from the <client> entry of the device. It is hopefully enough a small number of users to take their ire to blogosphere to make the manufacturer life harder.

Of course, only reliance on shame alone is not enough to protect the user's privacy so sDOMO introduces a third XML file for Standardized Expert Advice. The Hub will maintain a list of web-sites and a database with PKI Certificates of experts publishing device reviews. These expert reviews will be presented as an signed XML files for each device that has been reviewed and will contain a set of proposed alternative ACL's each of them with explanation of what privacy problems it solved and what functionality shortcomings generates. The Console Application used for installing a new device will automatically download the reviews and present the home owner with alternatives configuration options based on the compromise privacy/functionality they are willing to make. The Expert Advice Files can be automatically scanned by the online stores and provide ratings of the devices before purchase. This direct feedback can have the effect of making the vendors more privacy conscious with their choices.

It is important to notice that only the sDOMO messages sent via Hub are subject of these security checks which are performed by the Hub, allowing the devices in question to be small , microcontroller based, devices while in the same time all the security and privacy rules are enforced. This security level is not applicable to Streaming Interface and devices using streaming interfaces must in turn implement their own security and privacy models.

XML Files

sDOMO protocol has been designed to be very compact, fast and efficient to allow small footprint devices while conserving network bandwidth when sending large amounts of data like our distributes computer vision system for domestic robots. However the Hub, House Intelligence Unit and Installation Console needs to know more information about a Device while software developers writing an app to communicate to a Device must know the full set of specifications. Some alternative communication protocols are going the full route of using XML messages, sacrificing efficiency, speed and small footprint for convenience, expressiveness and easy to use.

To be able to achieve in sDOMO the same ease to use and power of expression while providing all the advantages of a fast and compact protocol, sDOMO introduced the XML Specification and Interface files. The Device is going to advertise only a compact form of the URL where the Device Specification File is located. The Hub will download this file and parse it when a Device of this type it is encountered, and use the values from it to provide default settings for the Device.

The Device Spec file contains documenting features, capability specifications and default configuration values, links to interfaces exported by the device (as a server) and to interfaces imported by the device (acting as client).

Documenting features allows for alternate language specifications for Device Name, short description and link to URL for both user and developer documentation; icons for representing the device in management GUIs etc.

Capabilities specifications contain the list of the configuration parameters of the Device, their valid ranges of values and default values for them. For example a sensor for the door or windows powered by a coin cell battery will want to stay in sleep state for up to 24 hours to preserve battery life. This device needs to specify in the Spec File that it can go idle for up to 90000 seconds at a time otherwise the Hub may attempt to ping it after 10 seconds of inactivity and

disconnect it after failed to respond to 3 consecutive pings.

```
<?xml version="1.0" encoding="utf-8"?>
<sDOMO-spec>

  <info>
    <specs>mezonix.com/sDOMO/demo/Sample1</specs>
    <desc lang="en">Test device for XML Parsing</desc>
    <desc lang="ro">Exemplu de dispositiv pentru testarea procesarii XML-ului</desc>
    <icon src="icon1.png" /> <!-- left -->

    <signature capable="Yes" required="Yes">
        <algorithm name="None" id="0" preference="0" security="0"/>
        <algorithm name="sha1" id="1" preference="5" security="7"/>
        <algorithm name="sha256" id="2" preference="4" security="8"/>
    </signature>

    <encryption capable="Yes" required="No">
        <algorithm name="None" id="0" preference="0" security="0"/>
        <algorithm name="xor" id="1" preference="1" security="1"/>
        <algorithm name="aes128" id="2" preference="5" security="7"/>
        <algorithm name="aes256" id="3" preference="4" security="8"/>
    </encryption>
  </info>

  <interfaces>
    <interface name="intf1" />
    <interface name="intf2" intf="mezonix.com/sDOMO/demo/Sample1/intf2" />
  </interfaces>

  <clients>
    <client spec="sDOMO.example.com/sDOMO/devs/Dev1" interface="int1" level="2" required="No">
      <reason lang="en">Just because I want to do that</reason>
      <reason lang="ro">Pentru ca asa vreau muschii mei</reason>
    </client>
  </clients>

</sDOMO-spec>
```

Interface Definition Files are referred from the Device Spec file specify and specify all the Messages and additional data structures used to build a Message. As stated before each software Object implements a given interface. In addition to this, the Interface File specifies how the Direct Messages defined above are combined to provide Remote Method Calls. The Interface Definition file it is used by an Interface Compiler that parse it and generate classes used to communicate with the device (Clients) or implement the specified service functionalities (Stubs).

```
<?xml version="1.0" encoding="utf-8"?>
<sDOMO-interface name="intf1" def="mezonix.com/sDOMO/demo/Sample1/intf1">
  <inherits>mezonix.com/sDOMO/defs/demo/intf0</inherits>

  <messages>
    <message name="Notification1" type="notification" id="100" security="1">
      <field name="field1" type="uint8_t" default="0"></field>
      <field name="field2" type="String[64]" default="&quot;default value&quot;"></field>
      <field name="field3" type="ByteArray[256]" default="0" />
    </message>

    <message name="Message1" type="direct"  id="100" direction="IN" security="2">
      <field name="field1" type="uint8_t" default="0"></field>
      <field name="field2" type="String[64]" default="&quot;default value&quot;"></field>
      <field name="field3" type="ByteArray[256]" default="0" />
    </message>

    <struct name="BasicInfo">
      <field name="field1" type="uint8_t" default="0"></field>
      <field name="field2" type="String[64]" default="&quot;default value&quot;"></field>
      <field name="field3" type="uint32_t" default="1"></field>
    </struct>

    <message name="Message2" type="notification"  id="101" direction="OUT">
      <field name="field1" type="int16_t" default="0"></field>
      <field name="field2" type="String[64]" default="&quot;default value&quot;"></field>
      <field name="field3" type="vector&lt;BasicInfo&gt;" default="0" />
    </message>
  </messages>

  <methods>
    <method name="firstMethod" input="Message1" output="Message2" />
  </methods>

</sDOMO-interface>
```

An interface file can inherit one or more interfaces and this is the preferred way to define them. If a particular Objects defines a totally new interfaces, a developer must write a totally new client program to talk with it. If an objects inherits from a standard interface and just add a few specific messages, any client program already written for that interface can connect and use the standard features of that device, while specialized programs written specifically for it can also take advantages of the new features.

A third XML file used by our domotic system it is the Standardized Expert Advice File. This file it is used by the specialists analyzing the devices existent on the market in order to eliminate potential privacy treats to users. Beside that the expert advice files can contain alternative default settings for the device (overriding parts the Spec file) and conditions when they are a better use that the one provided by the manufacturer. Another entries that may be added to the expert files will be rating of the device in accordance to various criteria to allow buyers make more informed decisions while shopping for a new device. Scripts for House Intelligence Unit and Internet Gateway can also be offered as part of the expert advice files.

All the files above are XML files signed using Public Key Cryptography. We are experimenting currently with “GnuPG clearsing” signatures but on a commercial system XML signature as specified by the W3C XML-Sig proposal are likely to be available too. Every sDOMO Hub is expected to maintain a database of PKI certificates along with a list of servers for expert advice and methods for querying them for a given file.

House Intelligence Unit and Internet Gateway

The House Intelligence Unit (HIU) it is a special device having close relationship with the Hub with which share access to the Hub Database. HIU is responsible for handling out of ordinary conditions that needs special attention. This handling it is achieved by a set of meta-processing rules allowing HIU to monitor critical sDOMO Notifications inside the system, and when certain pre-conditions are met emit alerts or handle the condition into other ways by the means of scripting.

To aid the automated building of the HIU rules for a device the manufacturer can add into the interface definition file potential alerts which upon a device start-up are loaded by the HIU. Beside rules from interface files, HIU can have it own set of rules added by the console from a system administrator or from the expert advice files for this device.

```
<intelligence>

  <meta_alert notification="CycleCompleted"
criticality="Low" domains="House" type=”text”>

        <condition> (msg.Counter >= 8) and
(msg.Counter %4 == 0)</condition>

        <text lang="en">Washing machine needs your
attention</text>

        <text lang="ro">Masina de spalat are nevoie de
atentia dumneavoastra</text>

        <append_msg_text msg_text=”Text”/>

  </meta_alert>

</intelligence>
```

Into the example from the figure above, when the washing machines cycle is completed, a CycleCompleted notification is broadcast into the system. The tablet used to control the washing machine remotely can be located into another room than the person watching her favorite show on TV so the person will not notice the beeping. The Washing machine keep broadcasting notifications at every 30 seconds which beside tablet are also received by the HIU due to the <intelligence> tag from the interface file. When the counter into the Notification reached 8 (i.e. 4 minutes in our example),

the condition tag (msg.Counter >= 8) and (msg.Counter %4 == 0) returns true every two minutes and in turn, HIU will start broadcasting an text alert to all the devices from the house having the ability to display them. The Smart TV will therefore display a bubble with the first line of text text: ”Washing machine needs your attention” and having on the next line the content of the field Text from the notification sent by the washing machine. This way the distracted house-keeper will be informed that an action is needed even if she forgot the machine remote-controller into the laundry room.

By specifying various criticality levels and domains of actions, HIU can alert a nurse about a fall of a resident as detected by a computer vision enabled surveillance camera or send to the home-owner vacationing out of town an text message in the same time with calling the fire-fighters with a prerecorded message if the smoke detector and heat sensors simultaneously detects activity in a room. HIU it is also the first line of defense against hacking, being the one alerting the users and the company installing the system about the attempt of device hijacking as detected by Hub using the sDOMO device hijacking prevention schema presented before.

The Internet Gateway is the only point of access from the outside world into the home domotic system. The house automation network contain sensitive information and can harm the residents in misused with intention or by accident. However, the home owner should be able to watch his surveillance cameras when an motion alert text message is received or a nurse should be able to take control of a robot from an independent living assisted house when an alert about a fall has been emitted. To allow for this contradictory requirements the Internet Gateway will be able to tunnel sDOMO messages toward devices located outside the house network. The communication is taking place over an encrypted network connection and the tunneling connection it is opened only for limited amounts of time subject to periodic re-autorization between the remote device and the gateway. The original authorization for opening the connection must be done via a separate communication channel like https or ssh and a Tunneling Session Key it is issued for the other end of the tunnel the Remote Device Service. The RDS in turn will establish an encrypted communication with the Gatweay encrypting the channel with the Tunneling Session Key and authenticating with the gateway with a separate key pre-assigned to that device. Once the tunnel is opened the Device software located in the mobile device will be able to connect to the Hub like any regular sDOMO devices..

Conclusions an future work

We successfully demonstrated the implementation of a native speaking sDOMO thermometer based on an Arduino Uno (2 KB SRAM and 32 KB Program Flash) an W5100 “Ethernet Shield” and TMP-102 I2C temperature sensor. This implementation proved that the sDOMO it is scalable enough to accommodate small devices as full nodes on the network while sending packets authenticated with SHA1 HMAC. The House Hub running on Mageia Linux has been shown to successfully route direct messages and notifications across multiple devices connected simultaneously and being able to properly manage their state. Performance measurements as well as implementation of Console Application, HIU and Internet Gateway are still pending.

References

[1] Marcel-Titus Marginean and Chao Lu, “A Distributed Processing Architecture for Vision Based Domestic Robot Navigation”, 2013 International Conference on Computers, Communications and Systems, Korea.

[2] Theis Solberg Hjort, Rune Torbensen, “Trusted Domain: A security platform for home automation”, Elsevier, Computers & Security 31 (2012) 940-955.

[3] Thinagaran Perumal, A. R. Ramil, Chui Yew Leong, “Interoperability Framwork for Smart Home Systems”, 0098 3063/11 2011 IEEE.

[4] Yung-Wei Kao, Shyan-Ming Yuan, “User-configurable semantic home automation”, Computer Standards & Interfaces 34 (2012) 171-188.

[5] Pedro Sanchez et al., “A framework for developing home automation systems: From requirements to code”, The Journal of Systems and Software 84 (2011) 1008-1021

[6] Poul Ejnar Rovsing et al., “A Reality Check on Home Automation Technologies”, Journal of Green Engineering 303-327 2011

[7] Vittorio Miori et al., “An Open Standard Solution for Domotic Interoperability”, 0098 3063/06 2006 IEEE

[8] Juing-Huei Su et al., “The Design and Implementation of a Low-cost Programmable Home Automation Module”, 0098 3063/06 2006 IEEE

[9] Ali Ziy Alkar et al., “IP Based Home Automation Systems”, 0098 3036/10 2018 IEEE