TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering

Denis Konstantinov 111615 IASM

Embedded service oriented microcontroller architecture.

Extensible client-server communication architecture for small devices

Master thesis

Supervisor: Peeter Elervee
Associate Professor at the Department of Computer Engineering / Ph.D., Dipl.Eng.

Tallinn 2013

Author's Declaration

This work is composed by myself independently. All other authors' works, essential states from literary sources and facts from other origins, which were used during the composition of this work, are referenced.

Signature of candidate: Date:

Acronyms

ASIC Application-specific integrated circuit. 5

Annotation

Current work introduces conceptual approaches for implementing an extensible service oriented client-server application on a small microcontroller. This is a general-purpose transport and hardware independent embedded server that uses remote procedure calls as primary communication protocol. This server looks like remote service that could provide defined functions to the client. ...

Annotatsioon

Annotatsioon eesti keeles

Contents

1 Introduction

1.1 Outline

1.2 Contributions

This is a test of acronyms ASIC[1]

This is a test of code listings

```
32 public Response getResponse(long timeout) throws InterruptedException {
33         synchronized (lock) {
34                 if (response != null) {
35                         return response;
36                 }
37
38                 lock.wait(timeout);
39         }
40
41         reader.removeInputHandler(this);
42         return response;
43 }
44
45
46
```

Listing 1: Example of a listing.

At first here will be lots of words about this cruel world and how it was changed in resent years.
Speech about reusable components.
Speech about small mobile devices that are everywhere.
Communication.
Interaction.

[1] Application-specific integrated circuit

2 Preliminaries

2.1 Service oriented architecture

2.1.1 Authentication and Authorization

Need of security Users are essential part of every system. System should be designed with a requirement, that there will be at least one user. System without any users does not make sense. Usually information systems have lots of users with different roles. There should be an system administrator - the most authorized individual in the system, managers and normal users. System should distinct them all somehow.

Another requirement is system and information security. System may contain sensitive data, that should not be available to general users. In case of remote services there are some services that are not open. These services or some of their parts are require some identity to pass through.

Let's take a usual website as example. Common website has at least three different user roles: user or guest, content publisher and system administrator. Last two roles may be joined together, but in general content publishers do not do system maintenance, they just work with content of webpages. There may be more different roles, but these are the main ones. Imagine you open a web page and you see the content. You follow the links and surf the web site. If you want to change something, for example you do not like the design or some words on web page were misspelled, you need to find special place where you can input your **credentials** and get into the system. This will happen only if you have proper **permission** to do that. When you get inside you are still not able to do anything due to lack of privileges. For example you cannot turn of the webserver or disable the your website. There may be lots of different roles and responsibilities in the system and each role has limited access to system resources.

Embedded device as a service may be similar to system example above. Device may have some limited use cases, that are not available to not authorized service clients. This may be internal information retrieving, some device manipulation functions (turn on/off something, delete/remove something from the system, change of system preferences). Some device functionality may be available only for limited people, for example system owner. The real life example of such system is the wireless router. Router clients are other computers, they can send and receive network packets. Router uses wireless security protocols, which permit unauthorized access. Even if you are connected to a secure access point, you are not able to change system settings. You should have admin permission (password) to manage the system. This kind of system, like many embedded systems, is made for one purpose. Router purpose is to provide access for the network. You also can remember lots of similar systems, that use authorized access. Nowadays it is not new to get remotely into some device and to change internals, but embedded system integrations are still not so common. Imagine near future, you are sitting at work and thinking to go home. After a long day you became really hungry. You take your smartphone and connect to your remote wireless fridge service at home. You type your password and get list of all food in your fridge. Now you know what you need to buy and the real candidates to be throwed to the rubbish bin. You adjust power in some fridge area and your beer will be very cold when you get home. Is it just a dream? Is it really hard to realize using present time technologies?

The main problem here is the security. Wired embedded network protocols are mostly not designed with security in mind. These networks were isolated from the Internet in the past. They could only be attacked using direct physical access to the network. Nowadays Internet and wireless networks are popular. Lots of communication between different systems goes through the wireless channel. Radio link is also available to your neighbour behind the wall. Generally, you do not want to broadcast what is in your fridge or to give ability to connect to your air conditioning service. Therefore you need to use some authentication scheme for your service.

Authentication protocols Authentication is any process by which you verify that someone is who they claim they are. (https://httpd.apache.org/docs/2.2/howto/auth.html)

Humanity has already invented a lot of different authentication techniques.

The ways in which someone may be authenticated fall into three categories: (http://en.wikipedia.org/wiki/Autl

- the ownership factors: Something the user has (e.g., wrist band, ID card, security token, software token, phone, or cell phone)
- the knowledge factors: Something the user knows (e.g., a password, pass phrase, or personal identification number (PIN), challenge response (the user must answer a question), pattern)

- the inherence factors: Something the user is or does (e.g., fingerprint, retinal pattern, DNA sequence (there are assorted definitions of what is sufficient), signature, face, voice, unique bio-electric signals, or other biometric identifier).

Authentication may be one way (only client is checked for validity) and two way (both client and server check each other). Some systems may requre to use different security factors together: you say password, provide ID card and show you fingerprint There are also available many standard authentication protocols. If you start searching you will probably find similar list:

- Transport Layer Security (TLS)
- Extensible Authentication Protocol (EAP)
- Password authentication protocol (PAP)
- Challenge-Handshake Authentication Protocol (CHAP)
- Password-authenticated key agreement
- Remote Authentication Dial In User Service (RADIUS)
- Kerberos
- Lightweight Extensible Authentication Protocol (LEAP)

Choosing suitable protocol is not trivial problem. There is no any case general protocol. Most of them are designed to interconnect big computers inside a network. Mostly they operate on transport and application level and use TCP/IP protocol stack.

All these protocols could be devided into these groups:

- Protocols that transmit the secret over the network. (For example Password authentication protocol). These protocols are not secure.
- Protocols that not send secrets and provide authentification through sending messages. (CHAP and Password-authenticated key agreement).
- Protocols that require a trusted third party.

Protocols of first type have been deprecated because of security reasons. They send sensitive data over the network and everyone else between two nodes can catch this data.

Second group of protocols was invented because the first group was unable to provide proper level of security. Link between client and server (two parties) does not contain pure information about the secret. Parties use cryptography and send encrypted messages to each other. Finally they authenticate each other when there is enough information gathered to validate the authority.

Last group uses trusted third party authority to check each other. There is assumption that all three parties should have connection between each other. Embedded device during client authentification needs to connect some server and ask for a secret. Third party should always have a high authority, two other parties should trust him. This scheme should be used in case of high security requirements.

Choosing of right authentication protocol in general should depend on application. Sometimes, there will be enough just to send plain text passwords over the network. Engineer should analyze all hazards during system design process.

Lifecycle of an embedded system is more longer than livecycle of average personal computer. Application specific controller may run for decades and it will be still functional. Choosed security algorithm may be not secure enough after some years. Some vulnerabilities can be discovered during that period. Computational power of modern processors raises every year and secure encryption may be cracked during some seconds in the future. There is no 100% secure system, everything can be breaked.

Your system security should have such encryption, that provides proper security level to your application data and can not be cracked quickly. How quick it is depends on your data and security requirements of your data.

Another aspect is the complexity of cryptography algorithms. Embedded devices are usually small low power devices with limited computation abilities. Not every algorithm suits well. It should be quick and resource friendly , and in same time it should be secure.

Nowadays, the last versions of the Wifi and Wimax standards include the use of Extensible Authentication Protocol (EAP) declined in different versions (LEAP - EAP using a Radius Server -,

EAP-TTLS, etc...). In practice, EAP is interesting for workstations or desktop computers but does not fit the needed security of particular systems such as handheld devices, short-range communication sys- tems or even domestic Wireless LAN devices. The reason being that many versions of EAP use certificates, public key encryption or exhaustive exchanges of information, that are not viable for lightweight wireless devices.[A new generic 3-step protocol for authentication and key agreement in embedded systems]

Protocol should small in code size. You should not to place a separate controller, which deals with communication, into your system. Everything is needed to be inside one small and cheap device. Business requires as low price as possible, because only that it could give you any money.

Embedded networking has constraints that developer should keep in mind while developing a system.

Embedded Network Constraints [A Flexible Approach to Embedded Network Multicast Authentication] Embedded networks usually consist of a number of Electronic Control Units (ECUs). Each ECU performs a set of functions in the system. These ECUs are connnected to a network, and communicate using a protocol such as CAN, FlexRay, or Time-Triggered Protocol (TTP). These protocols are among the most capable of those currently in use in wired embedded system networks. Many other protocols are even less capable, but have generally similar requirements and constraints:

- Multicast Communications - All messages sent on a distributed embedded network are inherently multicast, because all nodes within the embedded system need to coordinate their actions. Once a sender has transmitted a packet, all other nodes connected to the network receive the message. (In CAN, hardware performs message filtering at the receiver based on content.) Each packet includes the sender's identity, but does not include explicit destination information. The configuration of the network is usually fixed at design time, and changes a littel or does not change at all.
- Resource Limited Nodes - Processing and storage capabilities of nodes are often limited due to cost considerations at design time. Controllers, that are used usually have no more than 32 kilobytes of RAM and 512 kilobytes of Flash memory. Their operating frequency is no more than 100 MHz. Authentication mechanisms which require large amounts of processing power or storage in RAM may not be feasible.
- Small Packet Sizes - Packet sizes are very small in embedded network protocols when compared to those in enterprise networks. The bandwidth is very limited. Network synchronization and packet integrity checks should be added to this. For example data rates are limited to 1 Mbit/sec for CAN and 10 Mbit/sec for TTP and FlexRay. Devices cannot store large packets in memory during processing, as it was mentioned in previous requirement. Authentication should have minimal bandwidth overhead.
- Tolerance to Packet Loss - Embedded systems often work in a very noisy environment. Data may corrupt during transmission. Authentication schemes must be tolerant to packet loss.
- Real-Time Deadlines - In real-time safety-critical systems, delays are not tolerated. Processes should be completed within specified deadlines. Authentication of nodes must occur within a known period of time. There should not be unspecified delays.

Challenge-Handshake Authentication In this work i decided to use Challenge-Handshake Authentication Protocol [http://tools.ietf.org/html/rfc1994].

CHAP is an authentication scheme used by Point to Point Protocol (PPP) servers to validate the identity of remote clients. CHAP periodically verifies the identity of the client by using a three-way handshake. This happens at the time of establishing the initial link (Link control protocol), and may happen again at any time afterwards. The verification is based on a shared secret (such as the client user's password).

1. After the completion of the link establishment phase, the authenticator sends a "challenge" message to the peer.
2. The peer responds with a value calculated using a one-way hash function on the challenge and the secret combined.
3. The authenticator checks the response against its own calculation of the expected hash value. If the values match, the authenticator acknowledges the authentication; otherwise it should terminate the connection.

4. At random intervals the authenticator sends a new challenge to the peer and repeats steps 1 through 3.

The secret is not sent over the link. Although the authentication is only one-way, you can negotiate CHAP in both directions, with the help of the same secret set for mutual authentication.

This protocol is described in the document [http://tools.ietf.org/html/rfc1994]. Document specifies main protocol consepts and packet formats.

There are some procol extensions like MS-CHAP and CHAP is used as a part of other protocols like EAP(EAP MD5-Challenge) and RADIUS (uses CHAP packets). They all use CHAP concepts somehow.

One of the main puproses of this work is to develop a prototype of an embedded service. This system uses JSON [SEEE JSON SECTION] object format to encapsulate pieces of information. I will port CHAP packet format to JSON object. It need to be the same CHAP protocol but it should be placed into JSON. See [CHAP IMPLEMENTATION SECTION] for more details.

Conclusion Information security is continuing process. There are lots of scientist all over the world, that are trying to invent new approaches how to protect data.

In the Internet-connected future, designers will have to port existing security approaches to embedded control systems. This requires the use of lightweight security protocols.

Embedded control and acquisition devices may be integrated to the main infrastructure of the organisation. These connections need to be secure enough. Resent decades ago Internet was also a research project, and top computers were like nowadays microcontrollers are. But now it is used in whole world as one of the main communication methods. Even banks are using it for transactions. There are lots of security schemes with different level of protection. I believe that even small 8-bit microcontroller can be securely connected to World Wide Web it the near future.

2.2 Data serialization

2.2.1 JSON

2.2.2 XML

2.2.3 Others

3 System architecture

This section will introduce you a main architecture of the system.

3.1 Introduction

Here will be about coffee machine example in general

3.2 Server architecture

3.3 Client architecture

4 Implementation

Here will be implementation report.

4.1 Implementation of the embedded server

Here will be STM32 server implementation.

One solution is to use closed encrypted proprietary protocol and be calm, but as it was mentioned earlier, it limits the possibility of integration between other embedded systems. It this case all of your devices should support that protocol and you choise of different hardware is limited. Proprietary protocols are often vendor-specific, code is closed, documentation is not free and all it works only with the proprietary devices from the manufacturer.

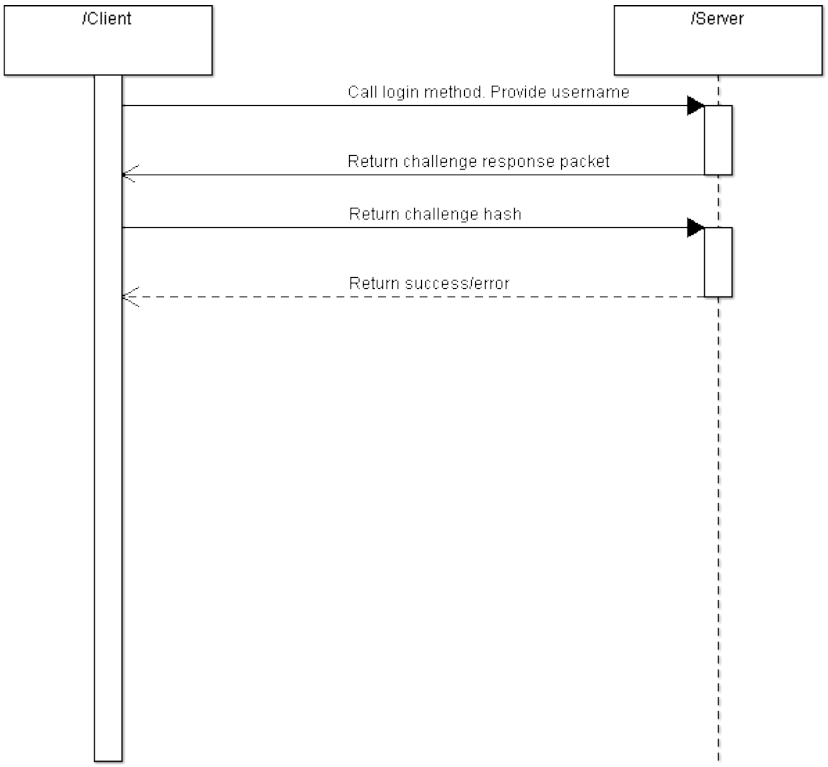

Figure 1: Client authentification process

reversably encrypted form.

4.2 General purpose service library implementation

Here will be general purpose library implementation report.

4.3 Implementation of android client

Here will be android java client implementation report.

Android development Some words about development under Android platform

5 Conclusions

5.1 Future work

List of Figures

List of Tables

References