СИСТЕМА ОТ ГАРДОВЕ В КИБЕР-ФИЗИЧЕСКОТО ПРОСТРАНСТВО

Борислав Тосков, Ася Тоскова, Даниел Русев
Пловдивски университет „Паисий Хилендарски“

GUARD SYSTEM IN CYBER-PHYSICAL SPACE

Borislav Toskov, Asya Toskova, Daniel Rusev

Abstract: *This publication presents a model of possible architecture of an intelligent guard system developed on the concept of IoT. The Guard System is part of the cyber-physical space of the Faculty of Mathematics and Informatics at Plovdiv University. It is built of intelligent agents and hardware components, providing personalization and ensuring the normal flow of the educational process.*

Key words: *IOT, Cyber-physical social space, architecture, intelligent guards system.*

Въведение

Във Факултета по математика и информатика на Пловдивския университет се разработва Виртуално Образователно Пространство (ВОП) [10] за подпомагане на електронното обучение. Като естествено продължение се явява необходимостта от интеграция на неговата виртуална среда с физическия свят, в който се провежда реалният учебен процес. Така ВОП прераства в кибер-физическо пространство, използващо тенденциите в развитието на най-съвременните технологии, една от които е интернет на нещата (ИН) [7].

Бурното развитие на мобилните технологии е последвано от ново масово навлизане на технологиите чрез ИН. Прогнозите за 2020 година е свързаните устройства да достигат 50 милиарда. Това разрастване на ИН води до появата на много нови стандарти за комуникация. Устройствата, осигуряващи свързаност в кибер-физическия свят, са доста специфични и се разработват от различни производители, така че е трудно да се приеме единен стандарт за комуникация между тях. Те често са с ограничени ресурси и не могат да използват пълни стандартни протоколни пакети. Освен това информацията не може да бъде предавана твърде често, заради пестенето на заряда на батериите. През по-голямата част от времето сензорите се намират в спящ (sleep) режим и сработват само при настъпване на събитие. Често те са свързани чрез хетерогенни безжични мрежи, чийто комуникационни протоколи са твърде специфични. Също така се използват различни езици за кодиране на информацията. Свързването на различни по вид сензори включва необходимостта от преодоляване на набор от проблеми, възникващи в различните слоеве на комуникационния модел. Използването на общи данни или извършването на различни действия изисква взаимодействие между устройства, разпръснати и достъпни чрез множество безжични технологии. Поради всички тези причини липсва интегриран подход на обмяна на информацията.

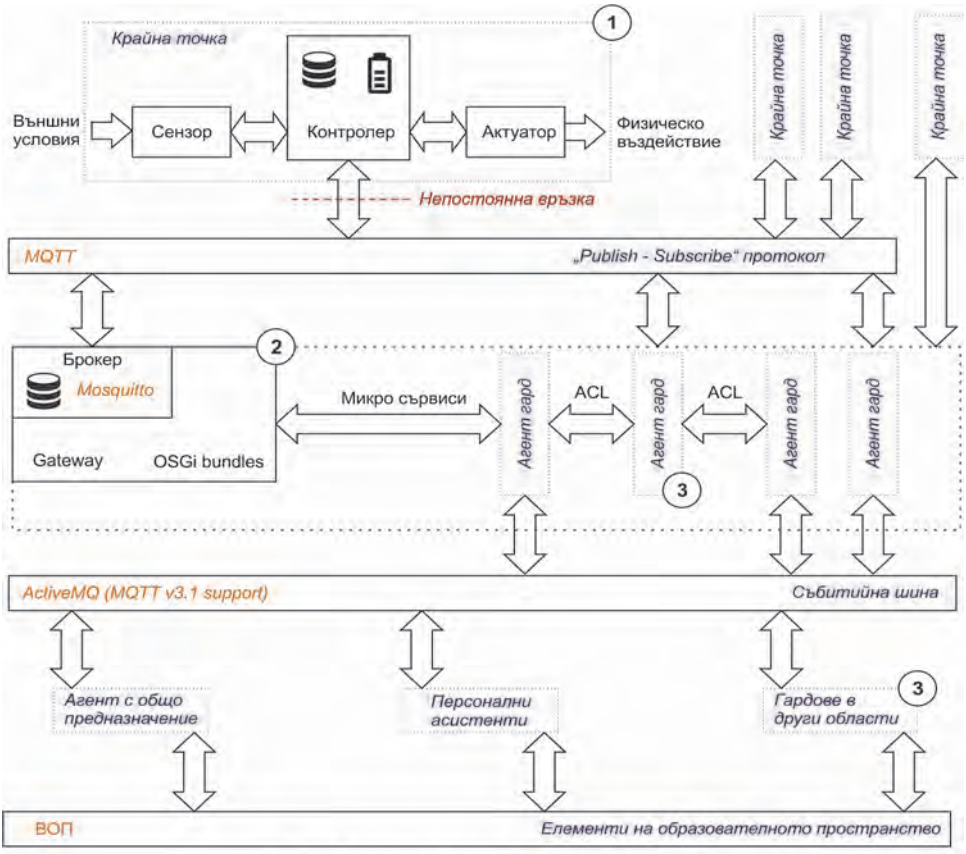
С настоящата разработка се предлага архитектура на интелигентна система, базирана на технологията на ИН, която позволява семантична свързаност между различни крайни устройства и ВОП чрез мултиагентна среда.

Архитектура на интелигентна гардова система

Един от възможните варианти за решаване на проблема с интеграцията е изграждане на система от гардове, които представляват софтуерни агенти, свързани със сензорни компоненти. Гардовата система отговаря за връзката на виртуалното пространство с физическия свят и подсигурява нормалната му работа, дори при извънредни ситуации. Системата от гардове се изгражда на базата на различни технологии – вградени системи, мрежова комуникация, мултиагентни системи и облачни технологии.

Софтуерните агенти са свързващите компоненти в гардовата система. От една страна всеки агент комуникира с другите агенти от виртуалното образователно пространство чрез съобщения, използвайки езика Agent Communication Language (ACL) на FIPA [4]. От друга страна агентите обменят информация със свързаните към тях сензори през специализирани протоколи.

На фиг. 1 е представена общата архитектура на гардовата система.



Обща архитектура на гардова система

Фиг. 1

Всеки гард интегрира няколко компонента:

1. Крайна точка (1) за връзка с физическия свят, която може да включва:
- сензор за измерване на физическа величина
- актуатор за промяна на физическа величина
- контролер за превръщане на физическата величина в цифров сигнал, готов за предаване за последваща обработка
- комуникационен модул за обмяна на сензорната информация, използващ жична или безжична връзка. Предаването се осъществява в зависимост от типа физическа свързаност по някой от стандартите IEEE 802.x
- вградена база данни за автономна работа

2. Маршрутизатор (2):
- комуникационен модул за събиране на информацията от сензорната мрежа
- брокер за обработка на съобщенията от и към сензорите
- мост за осъществяване на свързаност към други събитийни шини за препредаване на информацията или предоставяне на сървиси за работа със сензорите
- база данни за събиране на събития

3. Софтуерни агенти (3)
- OSGi бъндъли за агенти
- база знания

Възможни са два варианта на реализация на архитектурата в зависимост от местоположението на контейнера със софтуерни агенти.

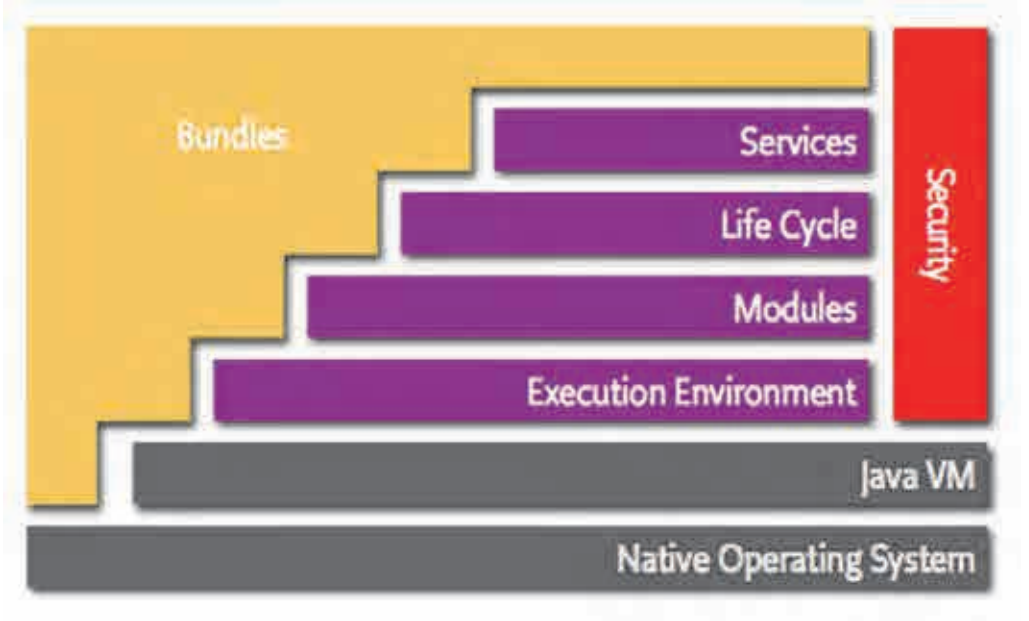
Контейнерът може да бъде разположен в маршрутизатора. Това е подходящо, когато се реализират агенти с реактивно поведение, които директно управляват актуатори. Това позволява повишаване бързодействието на реакция на системата.

При необходимост от проактивно поведение, контейнерът може да се разположи на сървъра. Така агентите ще имат бърз достъп до бази знания, подпомагащи изпълнението на поставените цели.

Софтуерните агенти, които живеят в контейнера, изграждат мултиагентна система. Софтуерните агенти по дефиниция са комуникативни, автономни, разпределени и мобилни. Всеки проактивен агент е интелигентен и рационален и за постигането на своите цели може да се кооперира с други агенти. Възможен вариант за реализация на мултиагентната система е платформата JADE [9]. От версия 3.7., освен стандартната дистрибуция, JADE се разпространява и като OSGi (Open Services Gateway Initiative) бъндъл [8]. JADE-OSGi бъндълът прави възможно да се стартира JADE-контейнер вътре в средата на OSGi. Това позволява създаването на множество агенти в този контейнер. Кодът на отделните агенти може да бъде пакетиран в отделни бъндъли. Това дава възможност за лесна актуализация и замяна на самите агенти. Така се дава достъп до всички характерни възможности на OSGi технологията, като регистрация и използване на различни микросървиси.

За работа с различните сензори в агентна среда е необходимо достигане на високо ниво на абстракция, което се предоставя от динамичната модулна система OSGi. Тя е изградена от слоеве, което я прави удобна за работа и тестване. Могат да се ползват готови бъндъли в различни комбинации. Модулният слой позволява бъндълите да се инсталират и обновяват отдалечено, чрез система за менажиране. Тяхната замяна не влияе на работата на системата. Отделните компоненти комуникират по-

между си локално или мрежово чрез сървиси. На фигура 2 е представен слоестият модел на OSGi.



Фиг. 2

Бъндълите се разполагат в контейнер. Понастоящем има четири реализации с отворен код на контейнери: Apache Felix, Eclipse Concierge, Eclipse Equinox и Knopflerfish.

За нашия експеримент може да бъде използван Eclipse Concierge [1]. Concierge е компактна реализация на OSGi Core Specifications R5 и е оптимизиран за мобилни и вградени устройства. Concierge доставя OSGi за устройства с ограничени ресурси, като например Raspberry Pi и Beaglebone black. Също предлага и поддръжка за Android върху Dalvik VM.

За осъществяване на интеграция на крайните устройства е необходим хардуерен посредник и мидълуеър софтуер. Възможно е хардуерният посредник да е специализиран маршрутизатор, изграден върху мини линукс система, в която мидълуеър софтуерът представлява логика за управление на сензорите. Тази логика предоставя локална автономност на свързаните крайни устройства, като позволява нормалната им работа при липса на интернет свързаност. Хардуерният посредник разделя PAN от WAN. Мидълуеър софтуерът изгражда логически междинен слой в структурата на интернет на нещата, който може да се разглежда като мъгла [5] с неопределена височина. Така се образуват зони на частична свързаност, работещи самостоятелно. Такива автономни зони могат да бъдат от различна величина, например: терморегулатор, поддържащ зададена температура; лаборатория, в която се контролира микроклимата; сграда, в която се провеждат учебни занятия и др. Недопустимо е умните уреди да престанат да работят при липса на връзка с Интернет. Разделянето на различни нива на свързаност е важно и при нарастване на броя на крайните устройства, за да може да се разпредели равномерно натовареността им.

Физическата връзка между устройствата в системата може да се реализира безжично по някой от стандартите: Bluetooth, WiFi, ZigBee, Z-Wawe, EnOcean, LoRa и т.н.

Комуникацията между крайните устройства и маршрутизатора се осъществява върху „publish-subscribe messaging” комуникационни протоколи. Този тип протоколи позволяват да се предава информация от сензора само в момент на настъпване на събитие. По този начин не е необходимо да се поддържа постоянна връзка с останалите устройства или приложения, които използват данните от сензора. Така крайното устройство може да работи в енергоспестяващ режим.

В днешно време за изграждане на ИН системи се използват няколко протокола - 6LoWPAN, Constrained Application Protocol (CoAP) и Message Queuing Telemetry Transport (MQTT).

6LoWPAN е протокол, базиран на стандарта IEEE 802.15 и се използва за изграждане на мрежи, които работят с ниска скорост на предаване на данни.

Constrained Application Protocol (CoAP) реализира REST сървис, като използва UDP и DTLS протоколи за M2M комуникация.

Message Queuing Telemetry Transport (MQTT) [6] е протокол, базиран на publish-subscribe messaging. Предимство на протокола е компактният отворен код, правещ го приложим във вградени системи. Той работи в приложния слой над TCP/IP и поддържа TLS криптирана връзка, което гарантира надеждно пренасяне на информацията.

Много популярни приложения използват MQTT:

- Facebook използва някои от аспектите на протокола за онлайн чат.
- EVRYTHNG IoT платформата използва MQTT като M2M протокол за комуникация между милиони свързани продукти.
- Amazon Web Services анонсира Amazon IoT, базиран на MQTT през 2015
- Microsoft Azure IoT Hub ползва MQTT като основен протокол при съобщения за телеметрия.
- XIM разработват MQTT Buddy – мобилен клиент за Android и iOS за управление на IoT устройства.
- Pimatic home automation framework за Raspberry Pi, базиран на Node.js предлага плъгин с пълна поддръжка за MQTT протокола.
- Компонентната среда на Bitreactive, Reactive Blocks, също поддръжат MQTT протокола.
- Node-RED –визуален инструмент за свързване на IOT устройства, съдържащ се в дистрибуцията RASPBIAN на Raspberry Pi включва блокове за връзка с MQTT брокери.

За да се осъществи publish-subscribe комуникация се използва брокер на съобщенията. Той се грижи за доставянето на съобщения до всички клиенти (абонати).

Реализирани са множество MQTT брокери: Moquette, Mosquitto, Xively, RabbitMQ, Apache ActiveMQ, HiveMQ, Mosca, JoramMQ и други.

За целта на нашия експеримент е избран Eclipse Mosquitto [2], работещ върху Rasspbery Pi. Eclipse Mosquitto заема малко памет и е подходящ за устройства с различни възможности – от едноплаткови компютри до мощни сървъри. Изпълнимите му файлове са от порядъка на 120 kB и при 1000 свързани клиента използва около 3MB RAM.

Клиентите могат да бъдат реализирани на различни програмни езици, като C++, C, Java, Javascript, Phyton и други. MQTT предоставя възможност всички клиенти да работят заедно, независимо от езика, на който са написани.

За да се създаде клиент, могат да се използват различни библиотеки, като например: M2Mqtt, MQTTnet, Eclipse Paho и др.

Eclipse Paho Java [3] е клиентска библиотека за MQTT, написана на Java, за разработване на приложения, работещи върху JVM или други, съвместими с Java платформи, като Android. Библиотеката е подходяща за нашия проект, тъй като той изцяло е базиран на Java.

Предложената архитектура позволява да се осъществят експериментални модели на софтуерни агенти, които да формират система от гардове. Предоставяйки връзка с физическия свят, гардовете ще следят за нормалното протичане на процеса на обучение.

Благодарност

Изследването е частично финансирано по проект No. MU17-FMI-001 EXPERT^{L} (Experimental Personal Robot That Learn), 2017-2018.

Литература

1. Eclipse Foundation. Concierge project, https://www.eclipse.org/concierge/.
2. Eclipse Foundation. Eclipse Mosquitto Project, https://projects.eclipse.org/projects/technology.mosquitto
3. Ecipse Foundation. Eclipse Paho Project, http://www.eclipse.org/paho/
4. FIPA. Agent Communication specifications, http://www.fipa.org/repository/aclspecs.html
5. F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, *ACM*, August 2012, ISBN: 978-1-4503-1519-7, 13-16.
6. MQTT community. MQTT v3.1.1., http://mqtt.org/
7. С. Стоянов, И. Попчев, Интернет на нещата, Техносфера, БАН, 3(37)/2017, ISSN:1313-3861, 33-44.
8. The OSGi Alliance. OSGi standard, https://www.osgi.org/.
9. TILAB. (2013, June) TILAB, http://jade.tilab.com.
10. Д. Орозова, С. Стоянов, И. Попчев, Виртуално образователно пространство, *Научна конференция с международно участие „Знанието – източник на иновация“*, *БСУ*, Бургас, ISBN 978-954-9370-99-7, 2013, 153-159.
11. С. Стоянов, Д. Орозова, И. Попчев, Е. Дойчев Виртуално пространство за продължаващо обучение, БСУ Научна конференция с международно участие, 2015, стр. 419-425.
12. Стоянов, С., Орозова, Д., Попчев, И. Виртуално образователно пространство – настояще и бъдеще, Юбилейна научна конференция с международно участие „Новата идея в образованието“, 2016, Бургас, 410-418.