PLATFORM FOR MOBILE AGENTS IN WIRELESS SENSOR NETWORKS

Jan Horáček
Master Degree Programme, FIT VUT
E-mail: xhorac08@stud.fit.vutbr.cz

Supervised by: František Zbořil
E-mail: zborilf@fit.vutbr.cz

ABSTRACT

Article deals with design of agent platform for sensor networks. Basically agent platform allows to run interpreted agent in sensor networks. This paper describes how to solve synchronization of agent platform with code interpret. Work also contains list of functions provided by platform to interpret and it will be shown basic comunication protocol.

1 ÚVOD

Senzorové sítě se začínají v poslední době přesouvat do poředí zájmu. Senzorovou síť [1] tvoří několik vzájemně propojených zařízení, která umožňují běh programu, stejně tak jako běžné PC. Navíc má možnost snímat data ze senzorů a komunikovat s ostatními uzly sítě. Důležitou odlišností od klasického PC je nízká energetická náročnost, související s nižším výpočetním výkonem. Pro naši potřebu bylo využito platformy *MICAz*, která definuje základní parametry mikrokontroléru.

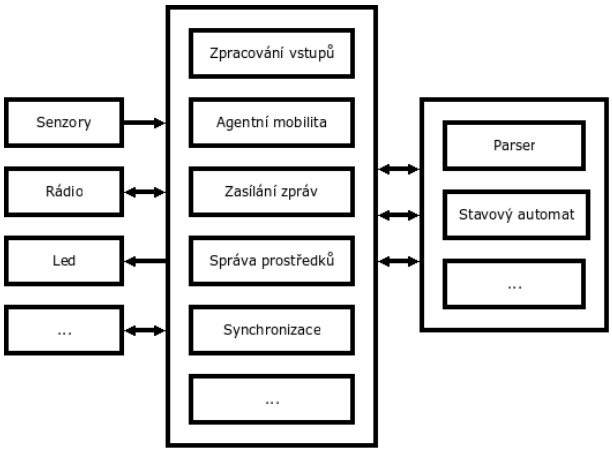
Také umělá inteligence a aplikace založené na agentním prostředí nacházejí svá uplatnění v mnoha oblastech. V tomto dokumentu bude nastíněn návrh agentní platformy, která je základní vrstvou pro samotný interpret agenta. Bude uveden seznam funkcí, které musí platforma nabízet a dále nastíněn základní návrh implementace.

2 ZÁKLADNÍ NÁVRH APLIKACE

Hlavním problémem v senzorových sítích je nedostatek systémových zdrojů. Pro platformu *MICAz* máme k dispozici 4kB (případně 8kB) paměti RAM, což je pro naše potřeby hlavním limitujícím faktorem. Jako operační systém se využívá operační systém TinyOS [2], který poskytuje sadu základních rozhraní k hardware mikrokontroléru.

Důležitým faktem zůstává, že platforma *MICAz* umožňuje nahrání či změnu programu pouze po propojení počítače se senzorovým uzlem pomocí USB sběrnice. Díky nemožnosti měnit program pomocí bezdrátové sítě a nutnosti zajistit agentní mobilitu, musíme agentní kód interpretovat. Pro popis agenta existuje již navržený jazyk ALLL, který je natolik jednoduchý, aby se dal interpretpretovat stavovým automatem. Na obrázku 1 můžeme vidět základní návrh aplikace.

Platforma zaujímá v celé aplikaci klíčovou roli. Operační systém TinyOS nabízí celou řadu asynchronních operací. Příkladem je zaslání zprávy z uzlu na uzel. Nejprve požádáme o zaslání zprávy, pokračujeme v činnosti a po nějaké době se teprve dozvíme, zda byla zpráva zaslána. Platforma se proto snaží z těchto asynchronních událostí udělat synchronní. Dále nabízí funkce na vyšší úrovni abstrakce. Při již zmiňovaném zasílání zpráv můžeme zaslat pouze zprávy o maximální velikosti 28 bytů. Pro naši potřebu ale vyžadujeme zasílání zpráv s proměnnou velikostí, kdy 28 bytů není dostačující.



Obrázek 1: Schéma aplikace

Platforma tedy abstrahuje základní rozhraní TinyOS a je jakýmsi prostředníkem mezi operačním systémem a samotným interpretem. Ve schématu jsou v levé části rozhraní poskytovaná TinyOS. Uprostřed je znázorněna část platformy a napravo interpretu.

3 FUNKCE POSKYTOVANÉ PLATFORMOU

Základní nutnou funkcí, kterou je potřeba realizovat, je správná synchronizace. Platforma vždy informuje interpret, že může provést jeden krok. V tomto kroku provede interpret malý úsek kódu, obvykle zavolání nějaké funkce platformy, přičemž přejde z původního stavu do stavu nového. Tím činnost interpretu končí. Platforma pak provede požadovanou funkci (pokud nějaká byla) a po dokončení informuje interpret, že může provést další krok. Součástí informace o dalším kroku mohou být výsledky operací provedených platformou v minulém kroku.

Je nutné stanovit omezení jednoho volání platformy na jeden krok interpretu. Pokud bychom toto omezení nedodrželi, mohly by vznikat cykly a program by nepracoval správně. Takto navržený model má stejné výpočetní schopnosti jako model bez tohoto omezení a proto se s tímto omezením nemusíme moc zabývat.

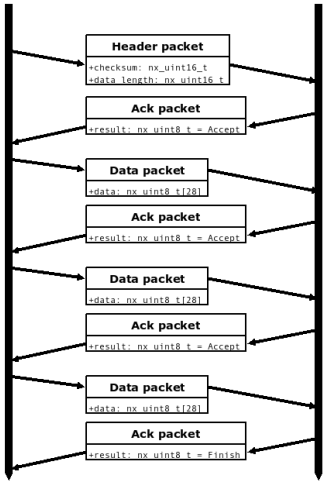
Další důležité funkce:

- **Čtení dat ze senzorů** - zpracování dat ze senzorů. Uchovávání historie naměřených dat a intervalové měření.
- **Zasílání zpráv** - zasílání zpráv proměnné velikosti. Rozdělení zprávy na jednotlivé pakety a následné sestavení zprávy. Potvrzování příjmu paketu a reakce na výpadky paketů.

- **Agentní mobilita** - přesun agenta z uzlu na uzel. Příjem a nahrání kódu agenta do paměti mikrokontroléru a inicializace nutná k začátku interpretace kódu.
- **Správa systémových prostředků** - ukládání zpráv do báze znalostí. Práce s dynamickou pamětí.
- **Sada algoritmů** - práce se seznamy (operace `cad` a `cdr` z *LISP*) a další.
- **Služby** - například vyhledání okolních uzlů.

4 KOMUNIKACE

Na obrázku 2 je znázorněn diagram popisující komunikaci dvou uzlů. Zprávu rozdělíme na několik paketů. Nejprve pošleme hlavičku obsahující velikost zprávy a následně pakety obsahující data. Druhá strana potvrzuje příjem. Stejným způsobem je přenášen kód agenta. Pro odlišení běžné zprávy od kódu využijeme hodnoty HandlerID. Jedná se o hodnotu z hlavičky přenášené na začátku každého paketu - viz. obrázek 3.



Obrázek 2: Dělení na pakety

Start	Dest addr	Source addr	Msg len	GroupID	HandlerID	var1 [16bit]	var2 [16bit]
00	00 02	00 01	04	22	06	8F 43	66 A0

Obrázek 3: Vzorový paket
V levé části hlavička, napravo datová část. Každý paket mimo jiné obsahuje zdrojovou a cílovou adresu, délku datové části (max 28 bytů) a HandlerID, kterým odlišujeme běžnou zprávu od kódu agenta.

5 ZÁVĚR

Byl navržen základní model celého systému. Práce díky požadovanému rozsahu nepopisuje implementační detaily, ale poslouží k základnímu pochopení funkce navržené platformy. Navržený systém je již z větší části funkční a schopný běhu jednoduchých agentů. Budoucí část vývoje bude zaměřena na návrh agentů běžících pod tímto systémem.

REFERENCE

[1] Haenselmann, T.: Sensor Networks, GFDL Wireless Sensor Network textbook, 2006

[2] Levis, P.: TinyOS/nesC Programming Reference Manual, 2006

[3] Zbořil, F.: Framework for Model-Based Design of Multi-agent Systems, In: Proc. EUROSIM 2007, Vienna, AT, 2007, s. 11, ISBN 978-3-901608-32-2