Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a umelej inteligencie

Informačný systém pre diagnostiku a správu robotov Nao

(Skupinový projekt predmetu Humanoidné technológie)

Analýza a návrh systému

Michal Širochman
Jaroslav Vraštiak
Tomáš Sabol
Michal Puheim

1. ročník Ing. štúdia
Umelá inteligencia
Letný semester 2011/2012

Úvod

Úloha systému je definovaná nasledovne:

1. Vyhľadať dostupných Nao robotov na sieti a zistiť ich meno, IP, stav a prihlásených užívateľov.
2. Výstup prezentovať v podobe web stránky.
3. V určitom intervale toto zbieranie a reprezentovanie dát opakovať.

Štruktúra takéhoto programu je popísaná v sekcii "Návrh architektúry systému", a preto sa úvodom budeme venovať len konceptuálnym základom návrhu systému.

Vyhľadávanie pracovných staníc je založené na predpoklade, že poznáme rozsah IP adries, ktoré priraďuje DHCP server, teda na ktorých adresách sa reálne môže robot vyskytovať. Pokiaľ sa zmení sieť a súčasne aj rozsah serverom priradovaných IP adries, alebo bude tento rozsah neznámy, je možné upraviť dané hodnoty v centrálnom konfiguračnom súbore. Viac o hlavnom konfiguračnom súbore je možné si prečítať v užívateľskej príručke.

Po nájdení pracovných staníc, ktoré odpovedajú na ping, sa program pokúsi identifikovať, ktoré z nich sú robotmi Nao. To vykoná tak, že pokiaľ sa mu podarí pripojiť na pracovnú stanicu s používateľským účtom "nao" a heslom "nao", alebo inými údajmi uvedenými v centrálnom konfiguračnom súbore, tak to je Nao robot, inak to Nao robot nie je.

Po úspešnom prihlásení sa si program zistí meno robota, IP adresu už pozná, keďže sa cez ňu na robota prihlásil, a užívateľov používajúcich robota si tiež zistí pomocou systémového príkazu. Stav robota bude reprezentovaný hodnotami "GOOD", "BAD", "FAIL" a percentuálnym vyjadrením pomeru GOOD voči súčtu všetkých týchto hodnôt:

- GOOD dostane robot len vtedy, ak test vráti hodnotu a jej hodnota patrí do intervalu daného zariadenia.
- BAD dostane robot, ak test vráti hodnotu, ale tá už padne mimo povoleného intervalu pre dané zariadenie - dané hodnoty intervalu budu znázornené červeným pozadím tabuľky pre daný interval.
- FAIL sa vyskytne vtedy, keď test vráti chybovú hlášku - tá bude zobrazená celá červeným písmom v tabuľke namiesto vrátenej hodnoty a intervalu.

Čo je to však "hodnota merania" a "interval"? Chyby sa budú zisťovať pomocou dodávateľom určených hraničných hodnôt komponentov, ako sú voltáž batérie, poloha kĺbu vyjadrená v uhloch, alebo hodnota dotykového senzoru. U všetkých týchto súčastí vieme rozsah hodnôt, ktoré môžu podľa výrobcu nadobúdať, a tak "hodnotou merania" rozumieme aktuálnu hodnotu týchto komponentov. "Intervalom" rozumieme rozsah povolených hodnôt, ktoré môže dané zariadenie nadobúdať.

Reprezentácia výstupu na webovej stránke je znázornená a popísaná nižšie, konkrétne v kapitole "Návrh používateľského rozhrania". Principiálne sa však výstupné údaje filtrujú do HTML súborov a do tabuliek podľa okruhu dát, ktoré reprezentujú. Hlavná stránka je súhrnný stav všetkých zachytených robotov od prvého spustenia, pričom sa navzájom odlišujú svojou MAC adresou. Pokiaľ sa pripojí na sieť robot s už použitou MAC adresou, ale iným menom, tak prepíše starý záznam. Pokiaľ sa napojí robot s rovnakým menom ale inou MAC adresou, je považovaný za odlišného robota a je mu priradená separátna identita. Každý z robotov má svoju vlastnú podstránku, na ktorú je odkaz reprezentovaný cez meno robota na hlavnej stránke. Na podstránke je úplný výpis údajov z robota.

Opakovanie procesu dosiahneme nastavením Crontabu, čo je systémový prostriedok pre

vykonávanie opakujúcej sa činnosti bez zásahu operátora. V rámci Crontabu je možné nastaviť opakovanie napríklad každé 2 minúty alebo každých 30 minút. Pokiaľ nám Crontab nevyhovuje, tak tento program pre diagnostiku je dodávaný aj s jednoduchým skriptom, ktorý stačí jednoducho spustiť spolu so zadanim časového intervalu v sekundách, a proces sa bude opakovať periodicky až do doby kým ho niekto nekillne. Tento obstarávací skript je možné pustiť aj s malým počtom sekúnd ako argumentom, keďže manažér diagnostiky má zabudovanú ochranu pred opakovaným spustením.

Diagnostika je distribuovaná ako tarball a pre inštaláciu ju stačí len jednoducho rozbaliť na miesto, kde má tento program sídliť. Pre beh programu však už je potrebné splniť niektoré požiadavky - tie sú podrobne rozvedené v užívateľskej príručke.

Prehľad podobných systémov vo svete

Self-diagnostika alebo inak health monitoring je dôležitá pre každého ľuďmi používaného robota. Jednoduchá self-diagnostika sa nachádza aj v tých najjednoduchších a bežne používaných robotoch. Napríklad aj moderný výsavač-bot obsahuje základnú self-diagnsotiku, ktorá je schopná preveriť všetky jeho časti či fungujú a následné človeku oznámiť úspešný výsledok respektíve povedať, ktoré konkrétne časti robota nefungujú.

Optimálna self-diagnostika by mala byť schopná zistiť čo všetko na konkrétnom robotovi funguje respektíve nefunguje a ohodnotiť tým stav daného robota. Táto diagnostika by mala byť schopná sa spúšťať automaticky v určitých pravidelných intervaloch, ale aj pred/po určitých činnostiach a samozrejme aj kedykoľvek na príkaz človeka. Zistený stav by mal byť prípadnému obsluhujúcemu človeku náležite vo vhodnej forme reprezentovaný.

Keďže náš projekt sa zaoberá akademickým robotom NAO-m je dôležité aby naša diagnostika spĺňala potrebné podmienky. Základné podmienky na NAO diagnostiku sú kvalitná kontrola všetkých robotových systémov. Keďže s robotmi pracuje množstvo študentov je dôležitý aj kvalitný, dobre čitateľný a ľahko pochopiteľný výstup diagnostiky.

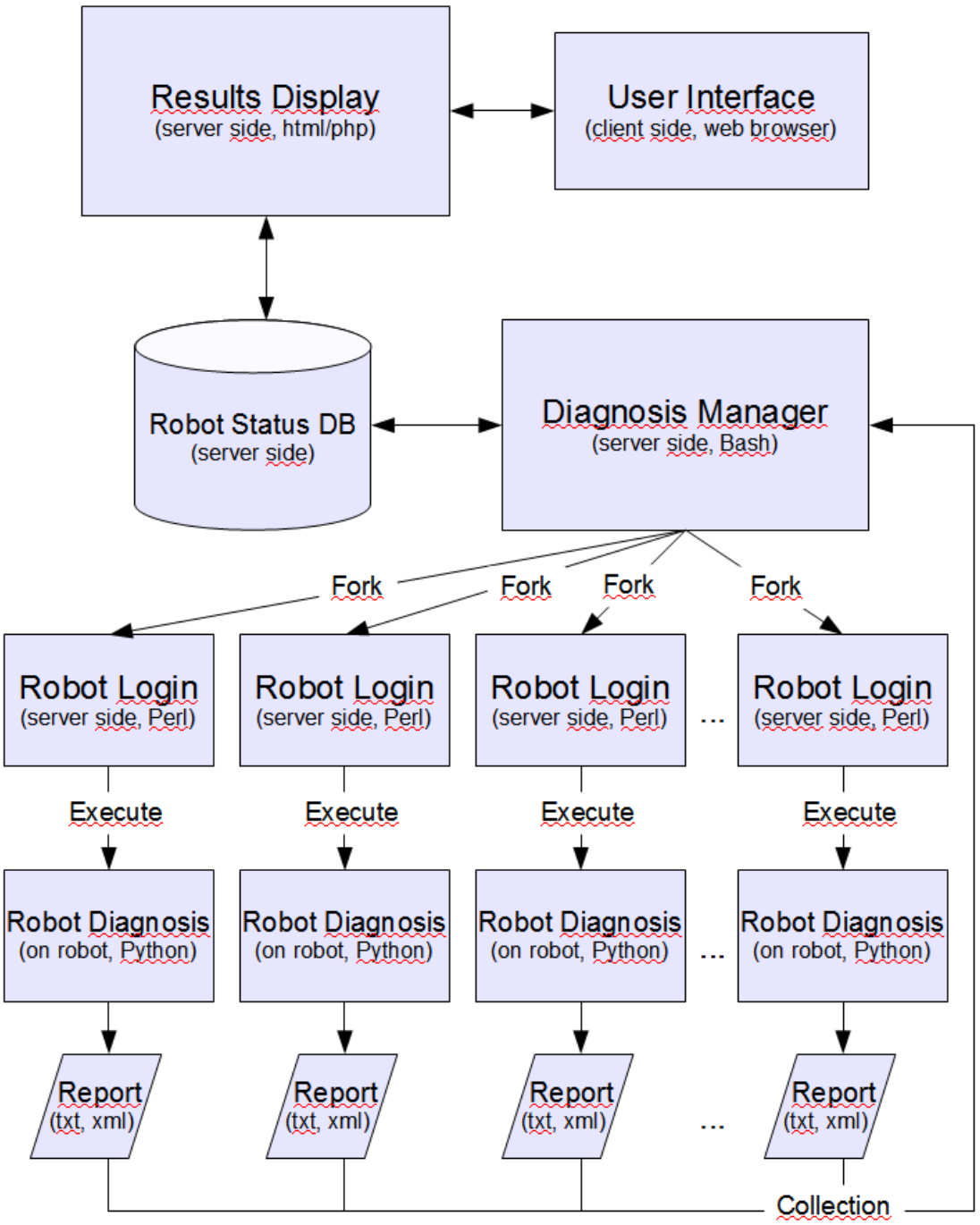
Samotný robot NAO tak ako aj ostatný zložití humanoidní roboti majú už integrovanú vlastnú kompletnú diagnostiku. Konkrétna NAO-va diagnostika skontroluje úplne všetky senzory teda hardware ako aj software. Prvý problém tejto diagnostiky je, že je veľmi pomalá. Pri našich pokusoch trvala od desiatok sekúnd až po pár minút. Druhý ešte väčší problém je jej výstup. Výstup tvorí skompresovaný balíček množstva súborov uložených na NAO-vi. K danému balíčku je najprv potrebné sa dostať napojením na robota a následne rozbaliť. Potom je možné sa ľubovoľne v obsahu balíčka pohybovať a strácať sa v 1000-riadkových neprehľadných súboroch a hľadať či je všetko v poriadku. Takýto výstup rozhodne nespĺňa potrebu dobre čitateľného a pochopiteľného výstupu diagnostiky. Na základe týchto problémov vznikla potreba vytvorenia nového na mieru ušitého diagnostického systému, keďže pôvodné diagnsotické prostriedky na robotoch sú vytvorené tak, že sú príliš zložité pre používateľov robotov.

Náš tím si stanovil za cieľ vytvoriť diagnostiku, ktorá by doplnila tieto nedostatky bežných predpripravených diagnostických systémov a to následovne:

- kontrolovať len potrebné veci a tým zrýchliť beh diagnostiky
- vykonávať diagnostiku paralelne na viacerých robotoch v rámci siete naraz
- vytvoriť dobre čitateľný a ľahko pochopiteľný výstup diagnostiky umiestnený na internete
- doplniť kontrolu nami zvolených vecí (napríklad zoznamu IP, MAC adries robotov)

Návrh architektúry systému

Navrhnutý systém je implementovaný pomocou kombinácie skriptov v jazykoch Bash, Perl a Python (resp.HTML pre užívateľské rozhranie). Systém spravuje komunikáciu s robotmi na lokálnej sieti, posiela na nich diagnostické skripty, ktoré následne spúšťa. Získané výsledky zbiera a ukladá vo forme súborov do jednoduchej databázy. Aplikácia zabezpečujúca užívateľské rozhranie tieto výsledky na základe požiadaviek užívateľa filtruje a zobrazuje v žiadanom formáte na webovej stránke.



Obrázok 1: Bloková schéma informačného systému pre správu a diagnostiku robotov Nao.

Jadrom systému je ***manažér diagnostiky***, čo je skript (v *Bash*-i) spravujúci komunikáciu s robotmi Nao, ktorý sa podľa nastavenia spúšťa buď jednorázovo, alebo v určených časových intervaloch. Tento skript sa v závislosti od rozsahu prehľadávaných IP adries na sieti pomocou volania *fork* rozvetvuje na množinu paralelne bežiacich skriptov. Každá vetva (príkazom *ping*) zisťuje, či na príslušnej IP adrese je pripojený nejaký počítač. Ak áno, potom daná vetva spúšťa pokus o *prihlásenie na robota*.

Prihlásenie na robota je skript (v *Perl*-e), ktorý sa pomocou SSH tunelu pokúsi pripojiť na príslušného robota. Pri inicializácii spojenia skript predpokladá, že všetky roboty používajú rovnaké prístupové heslo. Ak pokus o pripojenie zlyhá, potom skript usudzuje, že daný počítač na sieti nie je robotom, a ukončí sa. V prípade úspechu skript nakopíruje do pamäte robota *diagnostický* skript a spustí ho. Pre diagnostikovaného robota vytvorí v *databáze* súbor (vo formáte *txt* a/alebo *xml*) a uloží do neho výsledky diagnostiky.

Diagnostika robota je skript (v *Python*-e) spúšťaný na robotovi, ktorý zbiera informácie o stave robota (polohy kĺbov, stav batérie, teplota, pripojení užívatelia a iné).

Databáza stavu robotov obsahuje kolekciu súborov (vo formáte *txt* a/alebo *xml*), kde každý súbor predstavuje informácie o stave jedného robota na sieti. Databáza môže byť aktualizovaná na základe požiadavky užívateľa alebo pravidelne *manažérom diagnostiky*.

Zobrazenie výsledkov a ***rozhranie používateľa*** je vo forme html web stránok, ktoré obsahujú všetky informácie prehľadne usporiadané a formatované. Web stránku je tak možné spustiť v ľubovoľnom prehliadači.

Časový harmonogram implementácie projektu

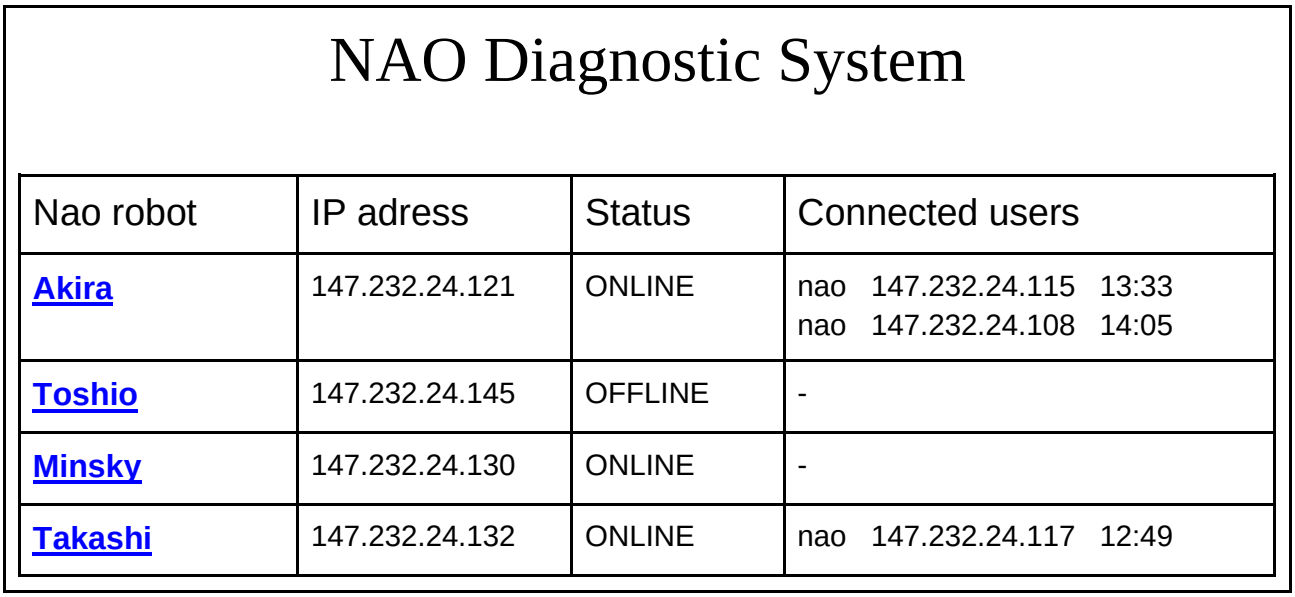
Časový harmonogram podľa týždňov :

1. Úvodná hodina Humanoidných Technológií, prevzanie zadania, vytvorenie HTML web stránky.
2. Základný návrh projektu a spôsob realizácie, prvý kód pre overenie potrebnej funkcionality.
3. Pokročilejší návrh projektu, perl a perl moduly, diskusia s cvičiacim predmetu, ujasnenie požiadaviek pre tím, vytvorenie BASH - PERL - PYTHON reťaze pre realizáciu projektu.
4. PYTHON - realizáciu skriptu na základnej úrovni.
5. PERL - po sprevádzkovaní python scriptu realizácia perl zložky realizačnej reťaze.
6. BASH - wrapper pre perl script, ktorý je 1-to-1 script (jeden script obslúži jedného Naa).
7. BASH - diagnostika siete, forkovanie, zber údajov, formátovanie výstupu.
8. BASH - diagnostika siete, forkovanie, zber údajov, formátovanie výstupu.
9. BASH - diagnostika siete, forkovanie, zber údajov, formátovanie výstupu.
10. Spustenie BASH - PERL - PYTHON reťaze, bug hunting, tweakovanie, základná realizácia web stránky aka užívateľského rozhrania.
11. Záverečná konfrontácia členov projektu s užívateľmi.
12. Prezentácia projektu, tvorba užívateľského manuálu.

Samozrejme, projekt sa bude priebežne konzultovať s cvičiacim a/alebo prednášajúcim.

Návrh používateľského rozhrania

Používateľské rozhranie je realizované ako webovská stránka. V hlavnej časti stránky sa nachádza tabuľka so základnými údajmi o robotoch Nao, ktoré boli nájdené v sieti, ako to je znázornené na obrázku č.2.



Obrázok 2: Hlavná stránka diagnostického systému pre robotov Nao.

Základné údaje o robotovi sú jeho názov, IP adresa, status a pripojení užívatelia. Status hovorí o stave, či sa na danom robotovi nevyskytla nejaká chyba. Pripojení užívatelia sú identifikovaní pomocou IP adresy, z ktorej sa na robota pripojili. Rovnako je k dispozícii aj údaj o konte a čase pripojenia.

Každý robot má vytvorenú vlastnú podstránku, kde sa nachádzajú kompletné informácie o danom robotovi. Okrem základných údajov to sú hodnoty polôh všetkých kĺbov, stav batérie, úroveň jasu LED svietidiel a podobne. Vizuálne je podstránka znázornená na obrázku č. 3.

Nao robot	IP adress	Status	Connected users
Akira	147.232.24.121	OK	nao 147.232.24.115 13:33 nao 147.232.24.108 14:05

BATTERY

CHARGE	TEMPERATURE	CURRENT
0.73	42	1.5

JOINTS

NAME	VALUE	RANGE	TEMPERATURE	CURRENT	STATUS
HeadPitch	0.0	-2.0857 to 2.0857	45	0.32	OK
HeadJaw	-0.46	-0.6720 to 0.5149	51	0	OK
RShoulderRoll	0.24	-0.3142 to 1.3265	38	0	OK
RShoulderPitch	1.35	-2.0857 to 2.0857	39	0	OK
RElbowRoll	0.12	1.5446 to 0.0349	36	0	OK
...					

LED

NAME	ID	VALUE
Head/Rear/Left	0	0.4
Head/Rear/Left	1	0.4
Head/Rear/Left	2	0.4
Head/Rear/Right	0	0
Head/Rear/Right	1	0.6
...		

Obrázok 3: Stránka vybraného robota s kompletnými informáciami.