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)

Systémová príručka

Michal Širochman
Jaroslav Vraštiak
Tomáš Sabol
Michal Puheim

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

Použité programovacie jazyky

Použité boli tri jazyky pre tento program, rátajúc len aktívne súčasti programu, keďže výstup je produkovaný v HTML jazyku. Použitie každého z nich má svoje opodstatnenie, avšak najprv si ich jeden po druhom vymenujeme:

BASH: `run.sh, nao_selfdiagnose.sh, formatter.sh`
PERL: `connector.pl`
PYTHON: `tractor.py`

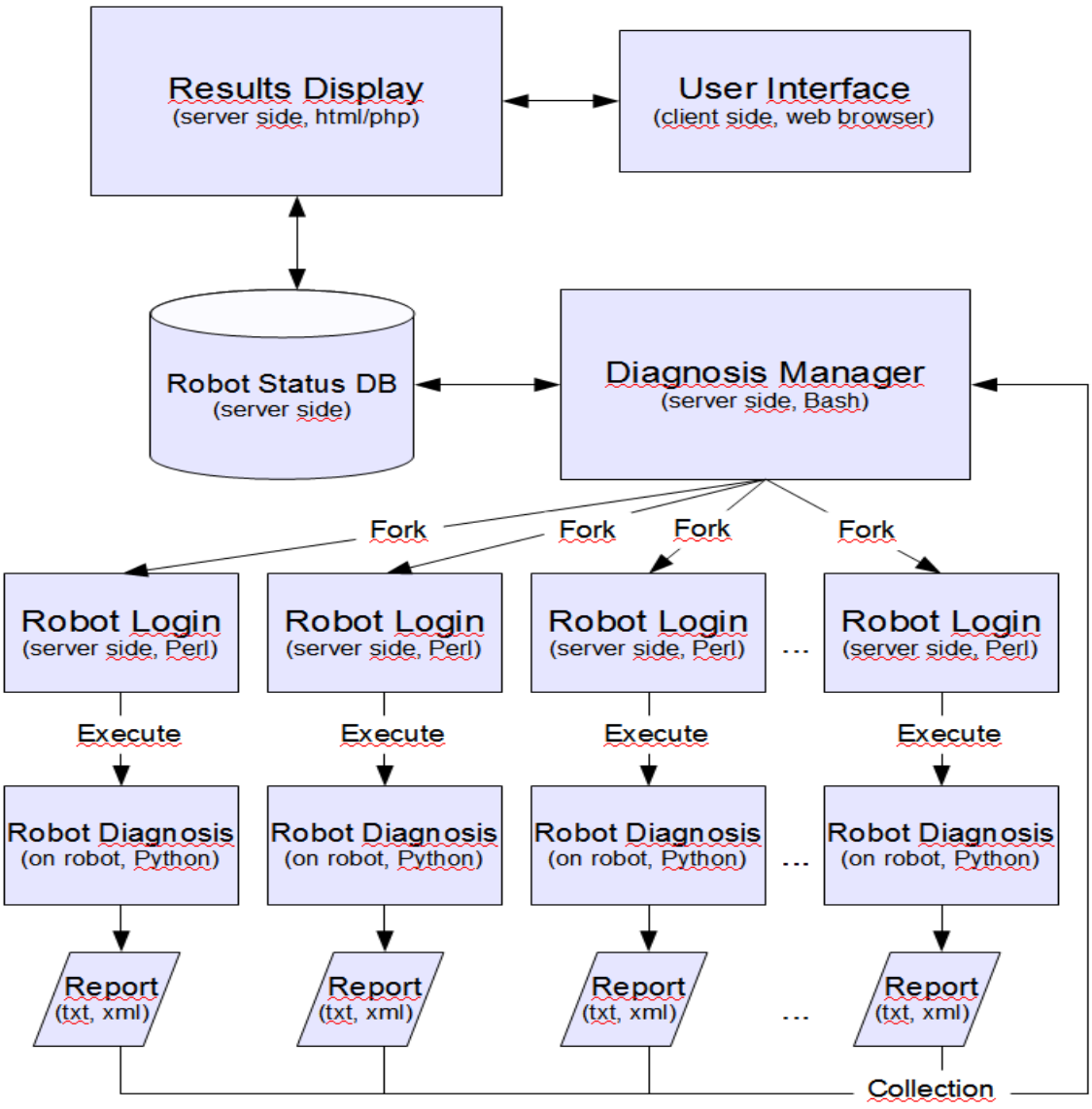
Bash je použitý ako natívny prostriedok pre automatizáciu úloh na UNIX-ových platformách, kým perl a python sú použité ako plnohodnotné programovacie jazyky pre špecializované úlohy.

Bash je akronym pre Bourne Again Shell. Je to voľne distribuovateľná náhrada pôvodného shellu, a je poskytovaný ako predvolený shell pre prakticky všetky UNIX-ové platformy ako Linux, Mac OS X a Darwin. Bash je skriptovací jazyk ako taký, no posktytuje dostatočné prostriedky pre realizáciu manažéra diagnostiky a skriptu pre formátovanie HTML stránok.

Perl je použitý najmä kvôli CPAN modulu SSH::Expect, ktorý je praktický nevyhnutný pre realizáciu diagnostického programu. Ako jediný nám poskytuje možnosť podsunúť príkazu ssh automatizovane heslo, a taktiež umožnuje prenášanie modulov pre jednotlivé verzie perlu medzi servermi, a teda pre použitie nemusí každý stroj spĺňať špecálne požiadavky - jednoducho dostane potrebné moduly spolu s programom. Nevýhodou je, že použitý modul pri chybách pripojenia zasypáva štandartný výstup chybovými hláškami, avšak to je nepodstatný problém pre automatizované úlohy bez interaktívneho pripojenia na server.

Python je použitý kvôli naoqi na robotovi nao, ktorý natívne funguje v pythone. Napriek tomu, že je zo všetkých skriptov najkratší, zabezpečuje práve to, čo nás najviac zaujíma - vykonáva merania mechanických zariadení na robotovi NAO, ako sú kĺby, senzory, a podobne.

Schéma systému



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

Manažér diagnostiky (Diagnosis Manager) je alfou a omegou, chrbtovou kosťou celého systému. V prvom kroku je to práve on, kto preveruje zoznam IP adries, a následne na tie aktívne rozošle požiadavky na prihlásenie prostredníctvom perl skriptu (Robot Login). V druhom kroku perl skript doluje informácie z robota, pokiaľ sa na neho úspešne napojil, prostredníctvom niekoľkých systémových príkazov, ale hlavne prostredníctvom python skriptu (Robot Diagnosis), ktorý na nao robota skopíroval. Ak všetko prebehne podľa očakávaní, na konci uloží na strane servera data súbor (report), ktorý obsahuje všetky dáta, ktoré získal. Vtreťom kroku zavolá manažér diagnostiky skript pre formátovanie HTML stránok z vydolovaných dát. Robot status DB tvoria všetky reporty dokopy, a z toho vychádza aj náš skript. Pre každý jeden report vygeneruje jednu osobnú stránku pre daného robota, ktorého report reprezentuje, a upraví jeho status na hlavnej stránke so zoznamom a statusom všetkých robotov. Po jeho skončení je všetko hotové a výsledky je možno vidieť vo vašom prehliadači.

Popis programu

V tejto sekcii sa budem venovať jednotlivým skriptom a ich funkciám a premenným. Premenné budú zobrazené v bloku ako výťah zo zdrojového kódu, a to iba globálne premenné. Funkcie budú vypísané a popísané jedna po druhej. Ako hlavnú referenciu pre nejasnosti sa silno odporúča použiť samotný zdrojový kód, ktorý je dostatočne okomentovaný na správnom mieste a štruktúrovaný sám o sebe. Skripty `run.sh` a `tractor.py` nebudú opísané, lebo ide len o jednoduché, aj keď šikovné, niekoľko-riadkové skripty.

nao_selfdiagnose.sh

Tento skript je hlavným skriptom, ktorý vykonáva paralelné úlohy.

1. pinguje rozsah IPčiek, aby zistil, ktoré sú aktívne
2. skúša sa pripojiť na aktívne IPčky pomocou `connector.pl`

Pre podporované prepínače spusti skript s prepínačom `-h`. Avšak všetky nastavenia odporúčam vykonávať v hlavnom konfiguračnom súbore `selfdiagnose.cfg`, kde sa dá nastaviť všetko a kľudne aj pridať dalšie premenné, ktoré chcete načítať.

Premenné (prednastavená hodnota):

`PID` - `PID` tohoto scriptu
`USER` - `USER` pod ktorým sa budeme prihlasovať na nao robota (nao)
`PASSWORD` - heslo pre `USER`-a
`IP_BASE` - prvé tri oktety IP, v ktorej doméne sa pohybujem
`IP_DHCP_MIN` - spodná hranica IPčiek pre kontrolu
`IP_DHCP_MAX` - horná hranica IPčiek pre kontrolu
`IP_END` - posledný oktet IPčky pre `IP_BASE`
`TIMEOUT` - `TIMEOUT` pre ping a PERL connector (5)
`PERL` - nazov PERL skriptu pre prihlasovanie sa na nao robota
`PERL_OUT` - výstup PERL scriptu
`PYTHON` - python skript pre vykonanie diagnostiky kĺbov & senzorov a pod.
`PYTHON_CONF` - konfiguračný súbor so zoznamom senzorov
`SENZOR_LIST` - konfiguračný súbor so zoznamom senzorov, ich typom a ich hodnotami
`CONFIGURATION_FILE` - hlavný konfiguračný súbor s nastaveniami
`HTML_FORMATTER` - názov skriptu, ktorý vytvorí z `DATA` súborov `HTML` stránky
`HTML_HOME` - kde majú byť uložené vygenerované stránky
`FORK_COUNT` - počet paralelných operácií (20)
`IP_LIST_ALL` - list všetkých IPčiek vygenerovaný pomocou `IP_BASE.IP_END`
`IP_LIST_ACTIVE` - list odpovedajúcich IPčiek spomedzi `IP_LIST_ALL`
`INFO` - info mód (off)
`DEBUG` - debug mód (off)
`START_TIME` - čas spustenia skriptu
`END_TIME` - čas ukončenia operácií
`TEMP1` - temporárny súbor pre potreby skriptu

Funkcie:

`_DEBUG` - debug hlášky (použi prepínač -d)
`_INFO` - informačné hlášky (použi prepínač -i)
`make_ip_list` - poskladá plný list IPčiek a hodí ich do poľa `IP_LIST_ALL` (global variable)
`make_active_ip_list` - vytvorí list IPčiek na ktoré sa bude connector pripájať podľa ich odozvy

na ping a vloží ho do poľa IP_LIST_ACTIVE (global variable)
fork - falošný fork - spustí v pozadí FORK_COUNT príkazov v rýchlom slede bez čakania na odpoveď, po ich skončení opakuje cyklus
readopts - načíta nastavenia z konfiguračného súboru
writeopts - zapíše nastavenia do konfiguračného súboru
check_env - skontroluje prítomnosť connector.pl, tractor.py a senzory.cfg
set_env - nastaví dôležité premenné prostredia
clean - vyčistí vygenerované perl connectory a ich dátové súbory z robotov
kill - nie je používaná, pôvodný zámer bol, aby po sebe skript ukončil aj child procesy, avšak táto potreba zanikla, a navyše táto funkcia generovala problémy
help - vypíše pomoc
set?time - nastavý časy do hlavnej stránky

V prvej fáze načíta konfiguračný súbor a inicializuje premenné, najme IP_LIST_ALL, do ktorej vygeneruje zoznam všetkých IPčiek pre kontrolu. V druhej fáze pinguje IPčky (len 1-krát) paralelne podľa nastavenia premennej FORK_COUNT, a výstup zapisuje do súboru ping.<IP>.<PID>. Následovne skript prejde všetky takéto súbory a hľadá tie, kde bol paket úspešne doručený a teda adresa odpovedala. Z týchto súborov si vezme z názvu IP adresu, a tú zaradí do IP_LIST_ACTIVE. V tretej fáze pošle skript connector.pl na všetky IPčky v premennej IP_LIST_ACTIVE (samozrejme paralelne), ktorý generuje výstupný súbor data.<IP>.<PID> s hlavičkou "OK" ak všetko prebehlo dobre, inak súbor buď nevygeneruje, alebo zahlási do neho podľa možností chybu. Vo štvrtej fáze, po dokončení tretej, zavolá skript formatter.sh, ktorý z data súborov obsahujúcich na prvom riadku "OK" vytiahne informácie, a na ich základe vygeneruje HTML stránky. Po jeho skončení nasleduje upratanie všetkých dočasných súborov, menovito TEMP1 a ping súborov.

Tento skript má základnú ochranu pre niektoré nezadefinované premenné, a taktiež sa odmietne spustiť ak už je raz spustený. že už je spustený mu indikuje súbor running.now, ktorého samotná existencia znamená, že momentálne skript pracuje.

connector.pl

Slúži pre automatizované pripojenie na robota, najmä vďaka modulu ssh::expect, ktorý umožnuje poskytnúť príkazu ssh heslo, keďže ssh nepodporuje automatizované zadávanie hesla. Po úspešnom pripojení sa na robota vykoná sled príkazov, ktorých výstup uloží do data.<IP>.<PID> súboru, a po nich skopíruje na stroj tractor.py a list.cfg, ktorý vykoná diagnostiku mechanických zariadení na robotovi. Výstup tejto diagnostiky je následne taktiež uložený do data súboru. V priečinku pmod distribuovanom v balíku má perl moduly pre verziu 5.8.8 a 5.10.1, pokiaľ používate iný perl, treba si z priečinka module_sources vybuildiť všetky moduly do pmod adresára.

Premenné (prednastavená hodnota):
$pid - PID tohto procesu
$host - host / ip adresa stroja na ktorý sa napája z nao_selfdiagnose.sh
$timeout - timeout z nao_selfdiagnose.sh
$user - USER z nao_selfdiagnose.sh
$password - PASSWORD z nao_selfdiagnose.sh
$tempFile - temporárny subor
$scriptName - názov python skriptu na poslanie
$inputScript - načítanie python skriptu do premennej

$listName - názov list.cfg na poslanie
$inputList - načítanie list.cfg do premennej

$host $timeout, $user a $password sú pozičné argumenty skriptu v danom poradí a podsúva mu ho nao_selfdiagnose.sh. Ostatné premenné slúžia na uskladnenie tých-ktorých výstupov až po finálny zápis na konci skriptu. Keďže sa jedna o priamočiary kód prakticky bez možnosti znovupoužitia nejakej časti kódu v inej časti skriptu, funkcie neboli vytvorené.

formatter.sh

Jedná sa o skript, ktorý z data súborov vyprodukovanými skriptom connector.pl vytvorí HTML stránky pre jednotlivých robotov a taktiež udržiava hlavnú stránku, ktorá je dynamicky prepisovaná pri každom behu programu. Roboti sú rozoznávaní podľa svojej MAC adresy, takže dvaja roboti s rovnakým menom sú prípustný. Ak však jeden robot bude používať MAC adresu ako nejaký iný pred ním (alebo je proste len premenovaný), tak jeho záznam prepíše ten starý záznam. Názov stránky robota je <meno>.<MAC adresa>.html.Všetci roboti sú považovaní za OFFLINE, pokiaľ sa neprihlásili ako ONLINE pri poslednej diagnostike.

premenné (prednastavená hodnota):
DATA_FILE - vstupný data súbor (ako výstup PERL skriptu)
HTML_FILE - výstupný HTML súbor, zložený ako "NAME.HWADDRESS.html"
IP - IPčka robota
$LOGIN - login na robota
HWADDRESS - MAC adresa
NAME - názov robota (hostname)
PYTHON_DELIMITER - obsah riadka označujúci, že ďalší obsah patrí výstupu PYTHON skriptu
PYTHON_LINE_NUMBER - čislo riadku, na ktorej je PYTHON_DELIMITER
WHO - výsledok [who] príkazu
UPTIME - výsledok [uptime] príkazu
SENZOR_LIST - názov konfiguračného súboru,, kde sú uložené zariadenia a ich parametre
TEMP1 - názov TMP súboru, ktorý používa tento skript pre zisťovanie parametrov zariadení
TEMP2 - začiatok main súboru až po prvý delimiter tag
TEMP3 - koniec main súboru od druhého delimiter tagu
TEMP4 - pre generovanú časť hlavnej stránky
DELIMITER_TAG - delimiter ktorý ohraničuje generovanú čast tabuľky zhora aj zdola
GOOD - premenná pre GOOD
BAD - premenná pre BAD
FAIL - premenná pre FAIL
MAIN_PAGE - názov hlavnej stránky
MAIN_PAGE_STATUS - ak súbor existuje, obnový ho, ak nie, vytvorý nový
HTML_HOME - cesta do web adresára ako prvý pozičný argument
Funkcie:
make_data_list - vytvorí zoznam data súborov
generate_personal_page - tvorba samostatnej stránky pre robota
make_rows - tvorba segmentov personálnej tabuľky robota
main_page - vytvorenie / update hlavnej stránky
clean - mazanie svojich TEMP súborov po sebe

V prvej fáze skript skontroluje či existuje hlavná stránka, ak bol zadaný `HTML_HOME` tak tam, inak lokálne. Pokiaľ ju nenájde, vytvorí novú, ak ju ale nájde, skopíruje si ju k sebe a bude pracovať s ňou. Hlavnú stránku rozdelí na tri časti, a to 2 statické - začiatok a koniec stránky, a 1 dynamickú - stred stránky, alebo inak povedané obsah tabuľky. V druhej fáze si vytvorí zoznam všetkých `data` súborov, ktoré začínajú na prvom riadku reťazcom "OK", a začne ich jeden po jednom spracovávať. Extrahuje všetky údaje okrem výsledku diagnostiky mechanických zariadení robota, a zavolá funkciu pre generovanie personálnej stránky. Tá ma v sebe prakticky len statické kúsky HTML stránky, a medzi ne volá funkciu `make_rows` pre vygenerovanie dynamického obsahu stránky. Funkcia `make_rows` je vlastne kostrou generovania personálnej stránky. Tam sa vyberajú jednotlivé komponenty zariadení z robota do kategórií a organizujú do skupín. Taktiež sa tam počíta celková chybovosť robota, a to GOOD/BAD/FAIL štatistika. Táto funkcia generuje riadok v tabuľke podľa toho, aký typ riadku dostala, a taktiež či namerané hodnoty sú správne alebo nie. Ak nie sú správne, vyfarbí problémové políčka na červeno. Pokiaľ sa meranie nepodarilo uskutočniť, vyfarbí chybovú hlášku na červeno. Po vygenerovaní osobnej stránky pre robota sa obnoví hlavná stránka už zo získanými údajmi. Takto sa to opakuje pre každého robota, pričom data súbory program zahadzuje hneď po použití. V poslednej fáze, keď spracuje všetky data súbory, poskladá dokopy hlavnú stránku, zmaže po sebe temporárne súbory, a týmto sa ukončí jeho činnosť.