„Universal Data Exchange Server“

Petr PFEIFER

Katedra měření, Fakulta elektrotechnická,
České vysoké učení v Praze, Technická 2, 166 27 Praha 6

Abstrakt: Příspěvek představuje vyvinutý datový server a související toolbox systému MATLAB pro univerzální výměnu dat mezi aplikacemi s využitím především pod platformou Windows. Pomocí tohoto toolboxu lze velmi jednoduše a poměrně rychle vyměňovat data, zasílat vstupní data a vysílat výsledky výpočtů mezi různými aplikacemi včetně DOS oken (!) nebo jiných aplikací, pracujících pod Windows, což výrazně odlišuje tento toolbox od jiných služeb Windows, které mohou být standardně MATLABem využity (např.DDE). Příspěvek se dále zabývá možnostmi použití, zkušenostmi z ročního používání a využití serveru v různých aplikacích.

1. Úvod

Pravděpodobně každý se již při přechodu na nový systém setkal s problémem konverze stávajícího softwarového vybavení a modelů na nový, zpravidla zcela odlišný systém. Tento problém je o to větší a zpravidla až neřešitelný v situaci, kdy je potřebné propojit programy na dvou, i tak zcela odlišných platformách, jako jsou DOS a Windows.

S podobným problémem jsem se taktéž potýkal. Stávající software, vytvořené kombinací assembleru a Pascalu s částmi původně C kódu bylo nutné nějakým způsobem integrovat s jinou úlohou v MATLABu, která prováděla převážně drobné finální zpracování výsledků a především velmi pěknou grafickou vizualizaci dat. Zde logicky vyvstal problém, jak tuto celou integraci zajistit, resp. konvertovat staré části do nového systému. Ihned se zde nabízí dvě řešení:

První možností je kompletní a časově náročný převod celé úlohy do Windows, resp. do MATLABu. Zpravidla ale potřebujeme nejdříve ověřit, zda má celá operace smysl, zda bude celá náročná konverze vyvážena výsledným efektem nebo parametry nového systému .

Druhou možností je najít způsob, kdy se velmi jednoduchou úpravou v dosavadních, poměrně obsáhlých DOS programech vytvoří nějaký komunikační kanál s MATLABem, jakožto aplikací běžící pod OS Windows. V případě pozitivních výsledků není poté problémem začít paralelně práci na novém systému a zároveň zdokonalovat nebo testovat hardware či postupy se stávajícím softwarovým vybavením.

Tento celý a vcelku složitý problém byl nakonec poměrně jednoduše vyřešen. Výsledkem snažení bylo vytvoření komunikačního serveru, jehož možnosti jednoduše řeší vzniklou situaci. Vývoj této aplikace včetně souvisejících úprav programů si nakonec vyžádal velmi krátký čas. Ve velmi krátké době se tak ověřila funkce celého nového systému a smysl celých uvažovaných úprav. Některé části byly sice nakonec přepsány do MATLABu, několik programů ale přesto zůstalo v původní formě, především díky své rychlosti nebo příliš složitému převodu do nového systému.

2. Stručný popis vyvinutého univerzálního serveru pro výměnu dat a jeho možností.

Problém nastíněný v úvodu může být i pojat jiným způsobem. Stále mnoho aplikací je totiž jednodušší a zpravidla i rychlejší napsat v některém starším a pro programátora známějším vývojovém prostředí, např. Turbo Pascal, Borland C, atd. Pomocí vyvinutého serveru se naprosto jednoduchým způsobem můžeme napojit do jakékoliv Windows aplikace, Matlab Simulink nevyjímaje, a provádět libovolnou výměnu dat.

Je zcela zřejmé, že řešení nastíněného problému není zcela triviální. Nicméně po hlubší úvaze je možné dojít k závěru, že všechny uvedené aplikace zase tak zcela rozdílné nejsou, že jisté softwarové pojítko mezi nimi přeci jen existuje. Je evidentní, že jedním takovýmto pojítkem je systém softwarového přerušení, resp. výjimek.

Posledně uvedený systém je použit i v představeném datovém serveru. V případě klasické Windows aplikace se po různém volání po knihovnách aplikací nakonec dojde ke komunikačnímu bodu v podobě vygenerování výjimky, která je serverem zpracována. Obdobně končí všechny DOS aplikace po volání softwarového přerušení na společném bodu obsluhy v podobě uvedeného datového serveru. Určitá část paměti je tak všem aplikacím „společná“. Datový server se tak tváří jako transparentní datový kanál.

Celý server je realizován jako jediný soubor velikosti cca 33KB, jehož instalace do systémů Windows95/98/Me se provádí velmi jednoduše automaticky při nabíhání systému.

3. Popis některých důležitých funkcí PDES Toolboxu

Neboť je MATLABu vlastní práce s funkcemi, byl vytvořen jejich soubor v podobě PDES Tolboxu, který pokrývá (nejen) všechny základní operace, které můžeme od takového datového serveru očekávat. Právě tato kapitola představuje některé důležité funkce tohoto PDES Toolboxu. Jak bude z následujícího přehledu zřejmé, je při obsluze serveru kladen důraz na její maximální jednoduchost. Jednotlivé funkce očekávají a vracejí minimum potřebných parametrů nebo výsledků tak, aby bylo jejich použití co nejjednodušší. V případě zpráv se obvykle pracuje přímo s typem double, což dále zjednodušuje použití funkcí. Uvedený seznam není samozřejmě úplný, ale lze si na jeho základě vytvořit dostatečný obrázek o funkci a jednoduchosti použití jeho jednotlivých částí. Z hlediska programového jsou všechny funkce reprezentovány DLL knihovnami pro MATLAB, které byly vytvořeny v jazyce C. Každá taková funkce má samozřejmě svůj *.m soubor, který obsahuje HELP část s příslušnou nápovědou a příklady na použití té které funkce.

3.1 Function [handle] = PDESREG(name)

Funkce zaregistruje nový komunikační kanál se zadaným identifikačním jménem *name* a vrátí přidělený *handle*. V poslední verzi serveru nabývá tato proměnná hodnoty 1-64, nicméně obecně se serverem vrací celé slovo.

3.2 Function [] = PDESUREG(handle)

Funkce zruší komunikační kanál přes server se zadaným *handle*. Protože již zaslané krátké zprávy se nemažou, může se při příštím vytvoření komunikačního kanálu se stejnou hodnotou *handle* již ihned nacházet ve „schránce“ několik zpráv. Nicméně tuto vlastnost lze zakázat příslušným parametrem konfigurační funkce. V tomto případě se po uzavření nebo otevření kanálu provede automatický výmaz všech v minulosti došlých a nepřečtených zpráv. Pozitivní stránkou je automatické udržování „čistoty“ ve vyrovnávací paměti zpráv. Nicméně u mnoha

aplikací je naopak výhodné, že při svém vytvoření a napojení na komunikační kanál mají již připraveny ve své „schránce“ vstupní , resp. počáteční parametry a nemusí se tak nijak složitě synchronizovat s protistranou.

3.3 Function [handle] = PDESSRCH(name)

Funkce se pokusí najít komunikační kanál *name* v interním seznamu a v případě úspěchu vrátí jeho *handle.* Takto je možné se jednoduše napojit na již dříve vytvořený komunikační kanál.

3.4 Function [IsPresent,Version] = PDESTEST()

Funkce otestuje přítomnost nainstalovaného serveru. Pokud je úspěšně detekován, vrátí *IsPresent* a verzi jádra serveru *Version*. Standardně nenalezení datového serveru končí automaticky zastavením programu a vypsáním chybového hlášení, což zjednodušuje použití funkce. Nicméně druhá, názvem podobná funkce PDESTESTC pokračuje v běhu programu i při zjištění nepřítomnosti datového serveru a pak musí být další větvení programu plně ošetřeno v programátorem.

3.5 Function [State] = PDESSEND(myhandle, handle , message)

Funkce zašle zprávu *message* zadanému *handle.* Pokud uvedeme parametr *myhandle* získaný při registraci kanálu, můžeme dodat příjemci zprávy další informaci o tom, kdo mu zprávu zasílá.

3.6 Function [message] = PDESRECV(handle)

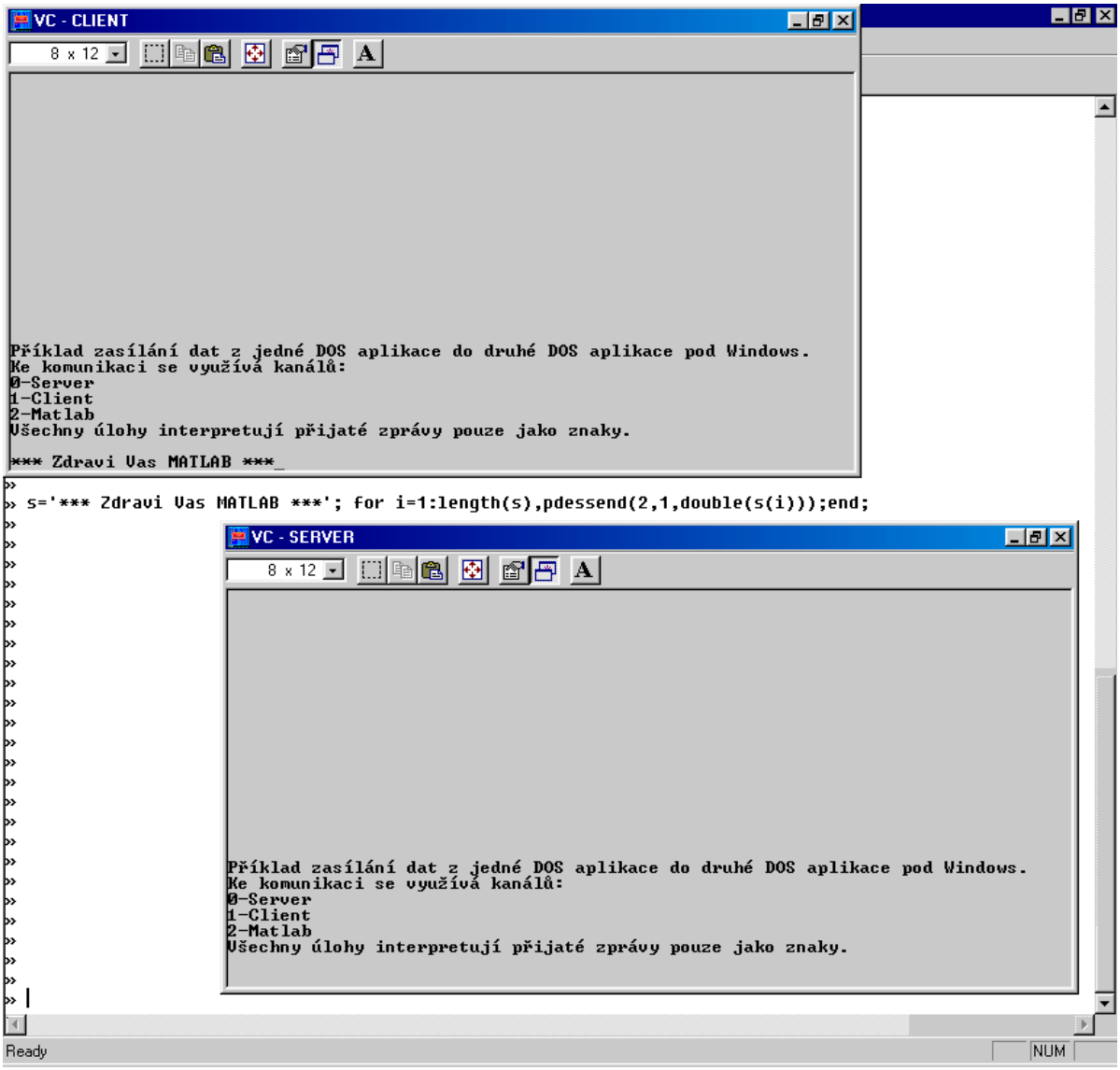
Funkce otestuje, zda zadaný *handle* má nějakou zprávu. Pokud žádnou zprávu nemá, čeká se na přijetí nějaké zprávy, přičemž se v čekací smyčce provádí volání Idle procesu (resp. jeho obdoby) tak, aby nebylo přerušeno plynulé přepínání úloh a tím se zbytečně nevytěžoval procesor touto úlohou. Po přijetí nějaké zprávy se tato vrátí jako *message.*

3.7 Function [message , sender’s handle] = PDESRECVI(handle)

Funkce otestuje, zda zadaný *handle* má nějakou zprávu. Pokud žádnou zprávu nemá, čeká se na přijetí nějaké zprávy, přičemž se v čekací smyčce provádí volání Idle procesu (resp. jeho obdoby) tak, aby nebylo přerušeno plynulé přepínání úloh a tím se zbytečně nevytěžoval procesor touto úlohou. Po přijetí nějaké zprávy se tato vrátí jako *message.* Pomocí dalšího výstupu můžeme zjistit *handle* a tím identifikovat zasilatele přijaté zprávy (pro účely odeslání odpovědi apod.).

3.8 Function [MsgCount] = PDESGETI(handle)

Funkce vrací informaci o počtu zpráv ve frontě pro zadaný *handle,* získaný při registraci. Příslušná úloha tak může provádět periodické testování příchozích zpráv a přitom provádět další potřebné operace.

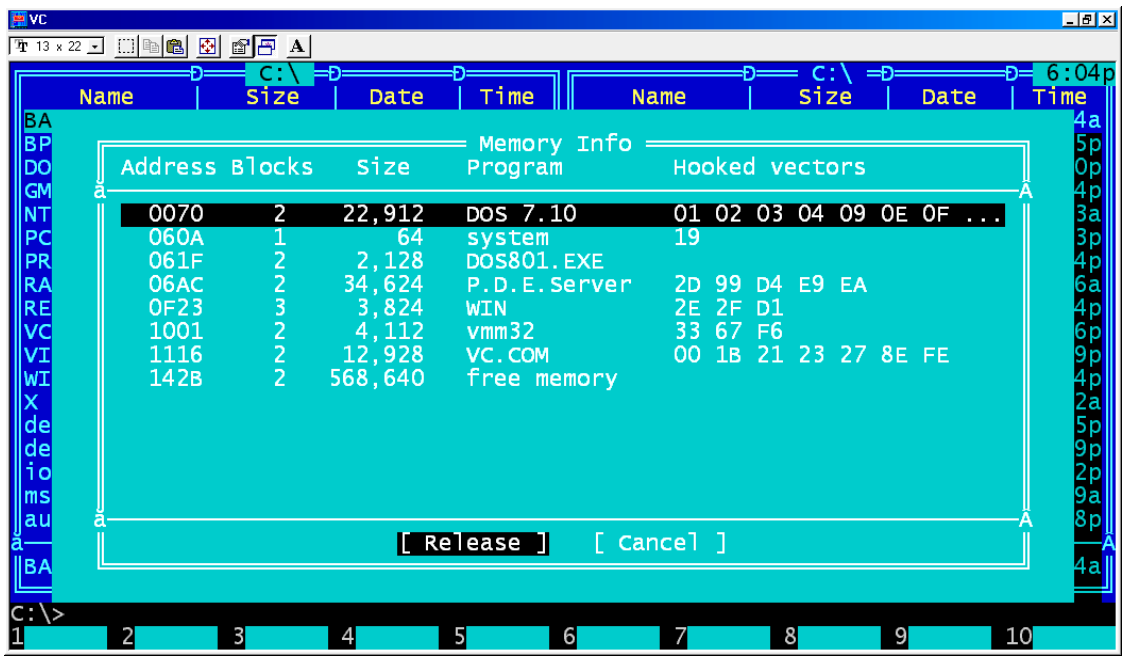


Obr. 1 Ukázka propojení MATLABU a dvou DOS aplikací
pomocí popisovaného datového serveru.

4. Příklad propojení MATLABu s DOS aplikací

Na obr. 1 je demonstrační příklad použití datového serveru a příslušného komunikačního toolboxu v MATLABu. Jak je z obrázku zřejmé, pod operačním systémem je spuštěn MATLAB a dvě DOS aplikace: SERVER a CLIENT.

Každá zúčastněná úloha má zaregistrován jeden komunikační kanál. CLIENT stále „poslouchá“ na kanále 1, kam vysílá aplikace SERVER hodnotu znaku stisknuté klávesy (psaný text se jednoduše kopíruje do druhé aplikace, co stisk klávesy, to vyslaná zpráva). Řádek uvedený uprostřed MATLABovského okna zajistí velice jednoduše vyslání zprávy v podobě postupného vysílání všech jednotlivých znaků řetězce *s* do kanálu pro CLIENTa. CLIENT má samozřejmě možnost při příjmu zprávy zjistit, kdo mu zprávu poslal a patřičným způsobem na ni reagovat.



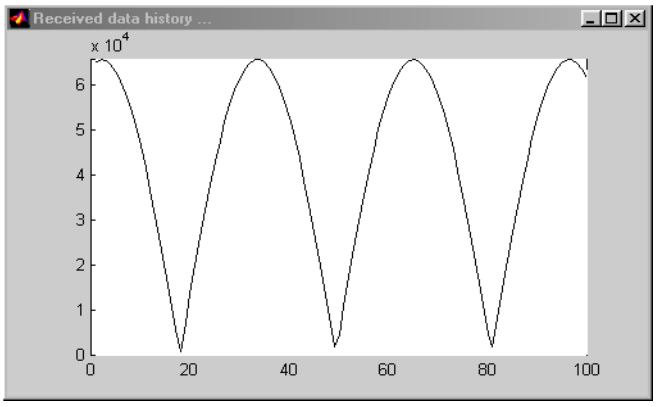
Obr. 2 Informace o serveru v DOS aplikaci (Volkov Commander)

4.1 Paměťová náročnost serveru v DOS aplikaci.

Jak je vidět z obr. 2, datový server se v DOS aplikaci objeví jako rezidentní úloha a v paměti zabírá cca 34 KB. Tato oblast paměti zahrnuje i paměť i informacemi až o 64 komunikačních kanálech a vyrovnávací paměť pro krátké zprávy. Velikost celého serveru může být zmenšena až na asi 4 KB, ale takto nakonfigurovaný server zbytečně snižuje propustnost celého komunikačního kanálu, neboť v podobě DOS aplikací se musí častěji čekat na synchronizace při přepínání procesů apod. Uvedená konfigurace serveru a tím i jeho velikost se ukázala jako zcela dostatečná i pod dnešními výkonnými systémy v podobě výkonných PC s procesory AMD Athlon XP/MP nebo Intel P4.

5. Příklad propojení Simulinku s DOS aplikací

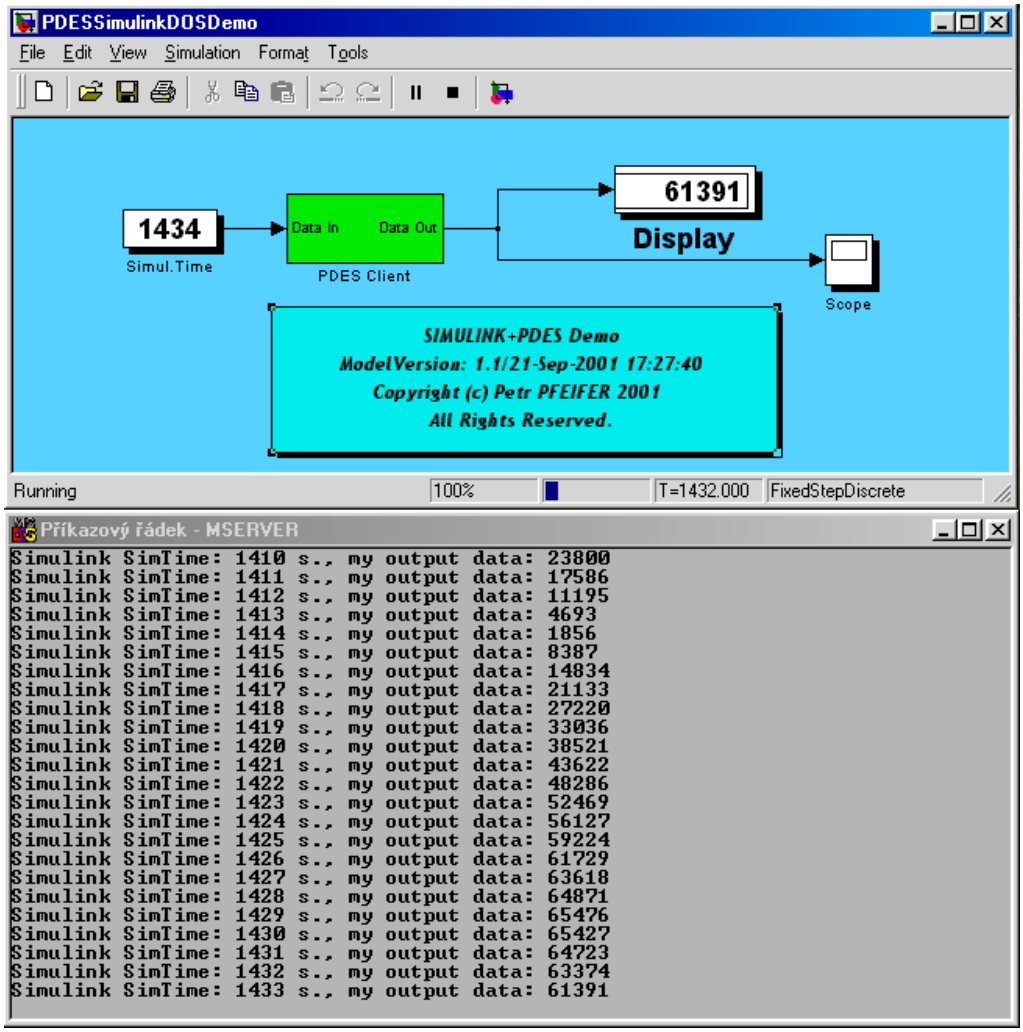
Na obr. 4 je příklad DOS aplikace, která provádí obecně nějakou výpočetní úlohu nebo sběr dat. Pomocí popisovaného serveru je propojena s modelem v Simulinku tak, že je schopna přijímat vstupní informace a informace o aktuelním simulačním čase v Simulinku. Může se takto s tímto modelem synchronizovat. Zpět do Simulinku jsou vysílány výstupní informace z operací v DOS aplikaci.



Obr. 3 Graficky znázorněná historie přijatých dat

Obr. 3 je kopií grafického okna příjemce dat ze serveru. Toto okno ukazuje nastavený výřez z historie přijatých dat přes datový server z DOS aplikace. Je zde vidět, jakou (triviální) úlohu asi DOS aplikace v tomto jednoduchém příkladě počítá.

Celá komunikace v uvedeném systému probíhá zcela jednoduše.



Obr. 4 Ukázka propojení Simulinku s výpočetní úlohou v DOS okně pomocí popisovaného datového serveru.

Simulink si sám (při začátku, resp. inicializaci simulací) spustí potřebný simulační subsystém a dále již pouze synchronizuje svoji funkci s touto úlohou. Naopak po (předčasném) dokončení simulací v subsystému v DOS úloze je zaslána informace Simulinku. Ten taktéž ukončí simulace ve své úloze.

Jak již bylo uvedeno, celý server umožňuje automatické vytvoření až 64 naprosto nezávislých komunikačních kanálů. Každá aplikace se může libovolně registrovat do vytvořených kanálů a ty používat. Zpravidla se ale každý kanál použije pro každou úlohu jako obdoba „schránky“, ze které si vybere zprávy a v jiném kanálu zašle odpověď.

Neboť se při vytváření komunikačních kanálů zadává i jméno vytvářeného kanálu, resp. jeho kód, není nutné vytvářet aplikace „natvrdo“, ale podle identifikačního řetězce si najít příslušný kanál, který vytvořil např. simulátor Matlabu pro zasílání vstupních informací a na tento kanál se napojit.

6. Další možnosti použití vyvinutého datového serveru.

V praxi může dosti často nastat situace, že DOS aplikace poskytne podstatně větší výpočetní výkon, než je schopna nabídnout i nejnovější verze MATLABu včetně jeho akcelerátorů v podobě strojově optimalizovaných prekompilací. Použití popisovaného serveru zjednodušuje začlenění této úlohy do celého simulačního a vyhodnocovacího serveru.

Z výše nastíněného popisu vyvinutého datového serveru je zřejmé, že je tento server vybaven všemi potřebnými funkcemi, pomocí kterých je možné vyměňovat data mezi všemi

aplikacemi, řídit jejich vykonávání a provádět jednoduchou synchronizaci (nejen simulačních) procesů.

Konstrukce serveru vylučuje problémy souběhů, tzn. nemusíme řešit problémy vzájemného vyloučení. Každá úloha se může libovolně dotazovat na data, vybírat nebo posílat zprávy, aniž by došlo ke konfliktním situacím ve vysílání a výběru zpráv při přepínání úloh apod.

Dosavadní verze serveru si je schopna uložit více než 4000 krátkých zpráv ve všech kanálech od všech aplikací včetně identifikace příjemce a odesílatele a dalších informací. Při vybírání zpráv příslušnou protiaplikací můžeme zjišťovat počet zpráv, přičemž při jejich „vyzvednutí“ nám je server samozřejmě nabídne k výběru ve stejném pořadí, ve kterém je přijal.

7. Závěr

Popisovaný program umožňuje velmi jednoduše vytvořit komunikační kanál mezi jakýmikoliv aplikacemi, tedy včetně DOS oken, resp. aplikací, pod operačním systémem Windows. Jedná se o poměrně jednoduchý, ale o to ve více aplikacích využitelný systém. Popisovaný systém si v žádném případě neklade za cíl nahradit některé další možnosti komunikací standardně používané v operačním systému Windows. Nicméně svojí velmi malou velikostí a naprostou nenáročností může velmi jednoduše odstranit problémy při implementacích některých starších programů do novějších systémů nebo zjednodušit počáteční testy možností implementací nových systémů nebo procedur. Mnoho aplikací je jednodušší a zpravidla i rychlejší zkušebně napsat v některém starším a pro uživatele možná známějším vývojovém prostředí, např. Turbo/Borland Pascal, Borland C, atd. Pomocí vyvinutého serveru se naprosto triviálním způsobem můžeme napojit do jakékoliv ostatní Windows aplikace, Matlab Simulink nevyjímaje, a provádět takřka libovolnou výměnu dat, tedy i implementace nových částí do stávajícího simulačního schématu, spojení se staršími „ovladači“ zařízení, apod., včetně celé integrace rozdílných systémů.

Popisované programové vybavení je již přes rok úspěšně používáno v několika aplikacích ve Windows 98 SE, včetně simulací v MATLABu. Dosud se nevyskytly žádné problematické nebo konfliktní kombinace aplikací nebo ovladačů. Některé simulace probíhaly s verzemi MATLAB 5.3 i zkušebně s MATLAB 6.1 v různých režimech zcela bez problémů.

8. Literatura

[1] Advanced Micro Devices, Inc., "AMD Athlon™ Processor x86 Code Optimization Guide", Publication #22007

[2] Intel Corporation, "Intel® Architecture Optimization Reference Manual", 1999

[3] Pfeifer, Petr: “Fast Simulation System for Simulation and Improvement of Digital Communication Systems in Transportation“, SCI2002, Orlando, Florida, USA,2002

[4] Internet (http://www.mathworks.com/, apod.)

Ing. Petr PFEIFER

Czech Technical University in Prague, Faculty of Electrical Engineering,
Department of Measurement, Technická 2, 166 27 Praha 6, Czech Republic
Telefon: (420) 2 2435 2807, *e-mail*: xpfeifer@fel.cvut.cz