

A SKELETON PAPER TO DEMONSTRATE THE UKSIM-CONF MACRO SET (REPLACE THIS LINE WITH YOUR TITLE)

YOUR NAME

Your institutions

address goes

here including

Postcode

E-Mail: your@email.address

Abstract: Insert your abstract contents here. For example you could put some claims about your wonderful results and how it is likely to revolutionise the particular field of study.

Keywords: some, comma, separated, keywords, go, here.

FIRST SECTION HEADING

The first paragraph in a section is preceded by `.LP`
all subsequent paragraphs are preceded by `.PP`.

You will need `groff-1.19.3` on a GNU/Linux system to utilise this macro set. You will also need to be familiar with the command line and familiar with GNU `make`, and you should have some knowledge of `troff` or `groff`.

Now for some example paper text which cites references. The CS1 and CS2 series of supercomputers employed a MIMD architecture from between 2 and 1024 compute nodes. The compute nodes varied in specification and ranged from T400 [Inmos, 1984], T800 [Inmos, 1988a; Inmos, 1988b], i860 [Intel, 1992] and SPARC [Sparc International, 1992] based systems. Notice how all references are referred to by:

```
. [
keyword
.]
```

and that `groff` will order and typeset the references correctly at the end of our paper. Some more dummy text follows to pad out this section. This paradigm is worth revisiting as commodity architecture now available has very similar characteristic's to that of the CS1 series in the early 1990s. For example the Cell processor used in the PlayStation 3 has a 64 bit processing element and eight 32 bit processing elements and overall the Cell can be viewed as having a MIMD architectural model with SIMD on the individual

processing elements. Irrespective of the Cell architecture many computing departments have a number of underutilised 64 bit and 32 bit compute nodes. Harnessing this compute capability has always been problematic and the availability of another message passing library offers flexibility and a migration path for legacy parallelised programs.

THE SECOND SECTION

This section is text fodder to give an idea of how input might look when composing a paper.

Parallel Virtual Machine [Geist, 1994] was originally developed at the University of Tennessee, Oak Ridge National Laboratory and Emory University in 1989. PVM is a software package which enables a variety of heterogeneous computers running a various operating systems to exchange data. In effect it provided the infrastructure necessary to implement grid computing.

The CSN slightly predates PVM and provides a simpler communication API. PVM provides facilities for fault tolerance, the CSN did not. The CSN provides support for both blocking and non blocking transmits and receives whereas PVM allows non blocking and blocking receives but only blocking transmits.

The simplicity of the CSN API together with the ability to handle non blocking transmits and receives make it attractive. It is well suited for use on a very fast, reliable and isolated local area network rather than in a grid computing scenario. Additionally the CSN is an attractive message passing interface to

employ when implementing a parallel algorithm on a modern multi core machine.

ANOTHER SECTION WHICH DEMONSTRATES AN UNORDERED LIST AND CODE

Below is an unordered list which is indented. It is a little cumbersome, nevertheless it works fine.

- it is a transport level interface,
- it provides blocking transmits and receives,
- non blocking transmits and receives
- and multiple simultaneous non blocking receives and transmits.

At this point we might wish to include some program text, this is done as follows:

Note the use of `.KF` and `.KE` (keep start and keep end) which insist that the program text is listed on the same page. Whenever we wish to typeset a program keyword in the main text of the paper we use the following macro: `.PRG` which lowers the font size by 1 unit and changes to a fixed font (courier). This macro takes 2 arguments, the first is the program text and the second (optional) argument is any punctuation. For example here is an extract of a previous paper.

Initially an application must create a `Transport` via `Open`. Any process wishing to transmit data must firstly lookup the receivers transport netid using `LookupName` and conversely a program wishing to receive data must register a transport using `RegisterName`. A process may now transmit data either by call to `Tx` which will return once data has been received by another process. Alternatively it may call `TxNb` which starts transmitting data and immediately returns allowing the caller to overlap processing activity with the transmission.

ANOTHER SECTION WITH A TABLE AND PICTURE INSIDE IT

Now we include our picture, which was created by the program `xfig` and saved as `transport.fig`. Our `makefile` generates the file `transport.pic` and here we include the picture by `.so transport.pic`

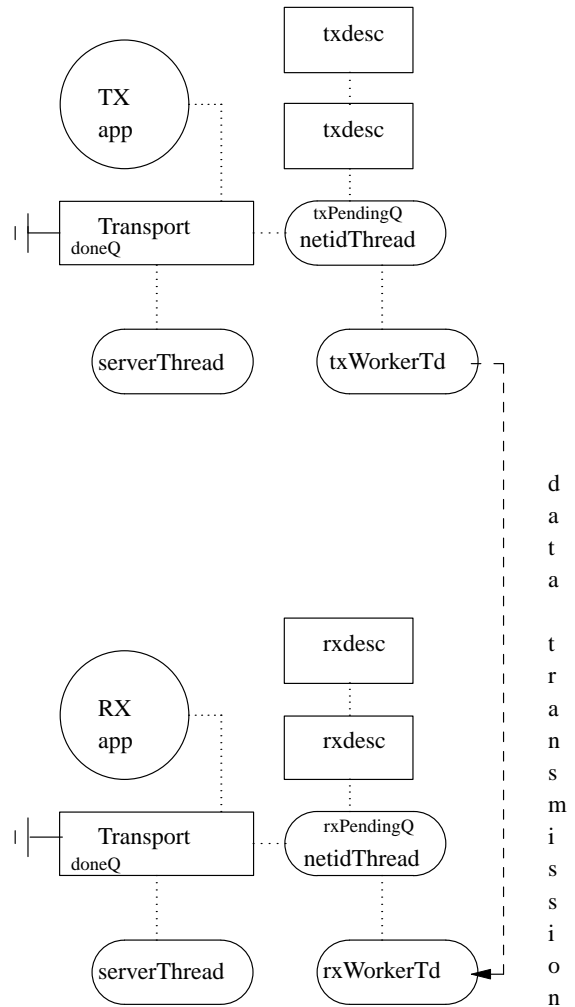


Figure 6: Relationship between the key data structure components during CSN communication

Block size (bytes)	Throughput (MByte/sec)		
	Single	Double	Quadruple
64	1	1	1
128	2	2	2
256	4	5	4
512	8	12	12
1024	14	11	23
2048	19	40	33
4096	47	37	70
8192	89	69	67
16384	137	137	178
32768	205	203	248
65536	283	274	306
131072	365	297	303
262144	329	285	257
524288	272	280	279
1048576	268	268	297

Table 1: Throughput of data in relation to buffersize

```

PROCEDURE Open (VAR t: Transport) : CsnStatus ;

PROCEDURE Close (VAR t: Transport) : CsnStatus ;

PROCEDURE RegisterName (t: Transport; name: ARRAY OF CHAR) : CsnStatus ;

PROCEDURE LookupName (VAR n: NetId; name: ARRAY OF CHAR) : CsnStatus ;

PROCEDURE Tx (t: Transport; n: NetId; a: ADDRESS; l: CARDINAL) : CsnStatus ;

PROCEDURE Rx (t: Transport; VAR n: NetId; a: ADDRESS; l: CARDINAL;
              VAR ActualReceived: CARDINAL) : CsnStatus ;

PROCEDURE TxNb (t: Transport; n: NetId; a: ADDRESS; l: CARDINAL) : CsnStatus ;

PROCEDURE RxNb (t: Transport; a: ADDRESS; l: CARDINAL;
               VAR ActualReceived: CARDINAL) : CsnStatus ;

PROCEDURE Test (t: Transport; flags: CsnFlags; timeout: CARDINAL;
               VAR n: NetId; VAR a: ADDRESS; VAR s: CsnStatus) : CsnStatus ;

```

Figure 1: The core CSN prototypes.

BUILDING THE PAPER

To build this skeleton paper you need to use the command line and firstly change to the `uksim-skeleton` directory and the type:

```

$ cd uksim-skeleton
$ make view

```

The command `make view` invokes a postscript previewer which updates the rendered image whenever you save the file `paper.ms`. Finally when you are happy with the paper content you can shutdown the postscript previewer and type:

```

$ make clean all

```

which will generate: `xhtml`, `PostScript` and `PDF` versions of your paper.

Notice that you can write your title and section headings in lower case troff will capitalise it for the PDF version and pass your text unaltered to the `xhtml` version.

Also note to add references to your paper you can extend the file `refs` in the skeleton directory. Here is the first reference in the file:

```

%A R.M. Stallman
%T Using and Porting the GNU Compiler Collection
%I Free Software Foundation
%C 51 Franklin Street, Boston, MA 02110-1301, USA
%D 2001
%K gcc

```

The `%A` is the author, `%T` the title, `%I` the institution, `%C` the address (city), `%D` is the date and finally `%K` is the keyword used to pull in the reference in the main text. So for example if we could cite the reference above by:

```

etc etc that to build some software
we suggested using GCC
.[
gcc
.]

```

and this would result in the following output text.

etc etc that to build some software we suggested using GCC [Stallman, 2001].

Notice that the PDF and XHTML outputs differ appropriately. The PDF version conforms to the format desired by `uksim` whereas the XHTML output uses plain numerical referencing.

CONCLUSIONS

Insert your conclusions here and finally leave the following three lines alone as they instruct groff to collect all references together and emit them at this point.

REFERENCES

Inmos 1984, *IMS T414 Reference Manual*, Inmos Ltd, Prentice Hall.

Inmos 1988, *Transputer Reference Manual*, Inmos Ltd, Prentice Hall.

Inmos arch, "IMS T800 architecture," Technical Note 6, Inmos Ltd.

Intel 1992, *i860 Microprocessor family programmer's reference manual*, Intel.

Sparc International 1992, *The SPARC Architecture Manual*, Version 8, Prentice Hall.

Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R. and Sunderam V. 1994, *PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, Scientific and Engineering Computation.

Stallman R.M. 2001, *Using and Porting the GNU Compiler Collection*, Free Software Foundation, 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.