The Transport Sample Protocol (TSP) white paper

Eric NOULARD
British Telecom
Consulting & Systems Integration

eric.noulard@syntegra.com

TSP the Transport Sample Protocol

TSP is a data sampling protocol which aims at simplifying sampling and observation of evolutionary data. Today two implementations in C and Java for different architectures exist. Since TSP is a protocol specification, it is very easy to write other implementations with other languages. TSP is an *open source project.*

Table of Contents

History

TSP was born from the collaboration of BT C & SI (formerly Syntegra)[1] and EADS Astrium [2] in the domain of validation tests benches for satellite assembly, integration and validation or test (AIV or AIT). Both parties worked together for years, so they can both build on firm technical experience in the domain of realtime test beds. The TSP specificication cames from the need for a efficient and versatile sample protocols which may brings a better reusability among realtime testbeds projects (not only in the satellite domain).

A simple design

TSP is a simple protocol whose design focused on simplicity and efficiency. TSP is easy to use and has low ressource needs (most notably CPU and network bandwith).

A provider/consumer model

TSP defines relationship between *providers* (TSP provider) and consumers (TSP consumer) of data. A TSP data provider is very simple, it is a process defining a list of named symbols (TSP symbols) with an associated value. The name of the symbol is a text string, the associated value is evolving (pseudo)-periodically. For example a TSP provider may provide 3 symbols:

Table 1. A TSP provider example

Symbol name	Minimal sample period	Type
time	1	double precision
temperature	10	double precision
nb_stars_in_FOV	2	integer
Base frequency = 100 Hz		

The provider has a *base frequency* which in this case is 100Hz. Then, the provider specifies for each TSP symbol its minimal admitted sample period. In the previous example the *time* symbol may be provided at 100Hz whereas *nb_stars_in_FOV* could only be provided at 50hz [3].

The TSP consumer may ask to any TSP provider for the list of the symbols he provides and other kind of information like its base frequency. As soon as the consumer obtained a TSP session id from a provider, he may ask the provider for receiving sample values. When both consumer and provider are informed of a valid sample configuration, the consumer ask the provider for beginning sampling, then raw data value are sent from provider to consumer side at the specified pace. At this step only *necessary raw data* are sent on the link, no symbol, no meta-data.

TSP communication channels

Communication between providers and consumers use two different paths: *command channel* and *data channel* .

Command channel: TSP request

The TSP command channel is the mean for consumer and provider to negotiate the sample configuration. It's an *asynchronous* command link which is logically *unconnected* . In fact between 2 asynchronous command, called TSP request, the network link between both side may have been down without any impact.

Each time a TSP consumer send a TSP request to a TSP provider, the consumer receive a TSP answer from the provider specifying how the request was processed by the provider.

Table 2. The TSP request

Request name	Role	Answer name
TSP_request_open	Ask for TSP session Id	TSP_answer_open
TSP_request_infos	Ask for provider informations	TSP_answer_sample
TSP_request_sample	Ask for sample configuration	TSP_answer_sample
TSP_sample_init	Ask for sampling process to start	TSP_answer_init
TSP_sample_finalize	Ask for sampling process to terminate	TSP_answer_finalize
TSP_request_close	Close a TSP session (the previous TSP session Id may not be used anymore)	TSP_answer_close

A typical TSP request sequence is following:

1. TSP_request_open
2. TSP_request_infos
3. TSP_request_sample
4. TSP_request_sample
5. TSP_request_sample
6. TSP_request_sample_init
7. TSP_request_sample_finalize
8. TSP_request_close

Current TSP implementation use ONC RPC [4] as command link interface. In fact TSP is made open (by design and current C implementation) to different kind of request handler. A forseen TSP improvement in this area is to implement several request handler kind:

- XML-RPC [11]
- SOAP [10]
- CORBA [12]

Data channel: TSP stream

The TSP data channel is used for efficient sample data flow as soon as the consumer ask for sampling process to start (TSP_request_sample_init). The TSP sampling data flow is optimal in the sense that only the necessary raw data will flow from provider to consumer side. For example, if a consumer ask for 5 double precision symbols produced at 16Hz, the data link will contains 5x8x16 = 640 bytes/second (double

precision values are stored on 8 bytes). Since both provider and consumer know what was the content of the TSP_request_sample, there is no need to transport meta-data. The data sent by the data channel are XDR [5] encoded in order to avoid byte ordering trouble between povider and consumer. XDR is used by ONC RPC [4], we may well have used CDR which is used by CORBA-GIOP [12]. Note that using other encoding for the data channel may be added to TSP very easily since TSP_request_information and TSP_request_sample contains a "requested features" bitsets with unused place which may be used to negotiate forthcoming features.

The sample request dialog phase enables the TSP provider to precisely specifies to the TSP consumer *before to send any sample data* the exact order of the raw data it will receive. The provider describes this data sequence in the TSP_answer_sample [2], so that when the consumer read the data flow he only have to pop the data in the specified order. This minimize the work of both sides during sampling.

A technical experimentation platform

TSP is easy to use since building a provider and/or a consumer from the TSP software library (C or Java) is simple. A sample Java consumer has been written in no more than a hundred lines, comment included. TSP is simple but its technical content is rich since TSP is meant to be a technical development and experimentation framework.

The technical objectives of TSP are described hereafter.

Portability

From its specification through the different implementation TSP aims at a maximum source portability, in order to be usable on a wode range of hardware platform.

The portability elements of TSP are:

- ANSI C and Java API
- multi-threaded model
- standards based : POSIX (thread, semaphores, . . .), XML, TCP/IP, XDR, ONC RPC
- multi-protocol (ongoing work): CORBA, XML-RPC, SOAP
- multi-plateforms: Solaris 32 et 64bits, OSF, Linux 2.4/2/6, VxWorks, Java, Win32 (ongoing work).

Robustness

TSP is using simple concept which should be autotested in the TSP SDK itself. TSP should be robust, since it is meant to be used long term running simulation (from several hours to many weeks).

Efficiency

Efficiency was a primary goal since TSP first design. Ressource requirement during sampling, are minimized in terms of network bandwith and CPU cycle on both consumer and provider side.

Maintenance

A TSP implementation *must be documented* in order to fullfil the TSP specifications [2]. Code documentation should should be processed automatically using a documentation extraction tool. [4]

Initial industrial partners did take the road to "Open Source" in order to make TSP cross the companies frontier. Free software is a source of motivation for TSP stakeholders (developpers and other contributors). TSP is hosted at the FSF project hosting solution: Savannah[5]

TSP objectives

TSP is a multi-purpose project.

A standard sampling protocol

TSP may become a standard data sampling protocol.

Wide range of application

- realtime or accelerated simulation
- test beds / test benches
- evolutionary phenomenon monitoring (real or simulated)
- data format conversion

Forseen development & evolution

TSP is usable now, but there is some place for improvement and evolution which would enable more widespread use. A non-exhaustive list of tasks is shown below, each task will be discussed and assigned by the TSP registered team at Savannah. External contribution are welcomed, but the preferred way is to join the TSP Savannah registered team in order to avoid redondant development.

The forseen improvement are following:

- Multi type and clean array handling (in C and/or Java).

  Technical content: multi-threaded C/Java programming ...

- Implements several TSP request handler (in C and/or Java).

  Technical content: XML-RPC, ONC-RPC, multi-threaded C/Java programming, CORBA-IIOP, ...

- Design cryptographic and/or compression support to TSP data channel.

  Technical content: SSL, TLS, socket programming, firewall, compression algorithm, ...

- Implement a XDR writer TSP consumer.

Technical content: XDR, XML, C and/or Java programming.

- Win32 Port.
  Technical content: Pthread-Win32, programmation C sous Win32 ...

- TSP provider lib in Java.
  Technical content: Java 1.4, thread, ...

- TSP consumer GUI in Java.
  Technical content: Java 1.4, thread, Swing/SWT ...

A. Contributors and helpers

Many people have contributed to TSP since its conception at different level.

Here is an incomplete list of contributors: Cesare BERTONA, Frédéric DEWEERDT, Yves DUFRENNE, Stéphane GARAY, Stéphane GALLES, Eric NOULARD, Robert PAGNOT...

Here is an incomplete list of helpers: Julien BRUTUS, Jean-Paul CHAUMIER, José DEKNEUDT, Christophe LAPLUME, Rachid TALAALOUT, Alexandre TARAYRE...

Bibliography

[1] *The TSP project at Savannah: http://savannah.nongnu.org/projects/tsp* [1] .

[2] *The TSP User Requirement Document (URD).*

[3] *The TSP Architectural Design Document (ADD).*

[4] *RFC 1831, RPC: Remote Procedure Call Protocol Specification, Version 2 : http://www.rfc-editor.org/rfcsearch.html* [2] .

[5] *RFC 1832, XDR: External Data Representation Standard: http://www.rfc-editor.org/rfcsearch.html* [3] .

[6] *CCSDS 102.0-B-5, Packet Telemetry : http://www.ccsds.org/all_books.html#telemetry* [4].

[7] *POSIX (1003.1-1990, 1003.1b-1993, 1003.1c-1994, . . . : http://www.opengroup.org/austin/* [5] .

[8] *Single Unix Specification version 2 or version 3 : http://www.unix-systems.org/version3/* [6].

[9] *ESA PSS-04-107 Issue 2 (April 1992): http://esapub.esrin.esa.it/pss/pss-cat1.htm*[7].

[10] *Simple Object Access Protocol (SOAP) 1.1: http://www.w3.org/TR/SOAP/* [8] .

[11] *XML-RPC Specification: http://www.xmlrpc.org/spec* [9].

[12] *Object Management Group: http://www.omg.org/*[10].

[13] *The Apache ANT Project: http://ant.apache.org/* [11].

Notes

1. http://www.bt.com/consulting
2. http://www.astrium.eads.net
3. minimal sample period is 2 out of a 100Hz base frequency pri ovder
4. Doxygen: http://www.doxygen.org" Doxygen for example
5. http://savannah.nongnu.org/projects/tsp
1. http://savannah.nongnu.org/projects/tsp
2. http://www.rfc-editor.org/rfcsearch.html
3. http://www.rfc-editor.org/rfcsearch.html
4. http://www.ccsds.org/all_books.html#telemetry
5. http://www.opengroup.org/austin/
6. http://www.unix-systems.org/version3/
7. http://esapub.esrin.esa.it/pss/pss-cat1.htm
8. http://www.w3.org/TR/SOAP/
9. http://www.xmlrpc.org/spec
10. http://www.omg.org/
11. http://ant.apache.org/