The Transport Sample Protocol (TSP) white paper

Eric NOULARD
BT Consulting & Systems Integration
Software Architect

eric.noulard@syntegra.com

Yves DUFRENNE
EADS Astrium
Expert in RealTime Test Benches

yves.dufrenne@astrium.eads.net

TSP the Transport Sample Protocol

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

1. History

TSP was born from the collaboration of BT C & SI (formerly Syntegra) (http://www.bt.com/consulting) and EADS Astrium (http://www.astrium.eads.net) 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 real-time test beds. The TSP specificication comes from the need for an efficient and versatile sample protocol which may bring better reusability among real-time testbeds projects (not only in the satellite domain).

2. A simple design

TSP is a simple protocol whose design focuses on simplicity and efficiency. TSP is easy to use and has low resource needs (especially regarding CPU and network bandwidth).

2.1. A provider/consumer model

TSP defines a 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 evolves (pseudo)-periodically. For example a TSP provider may provide 3 symbols:

Table 1. A TSP provider example

Symbol name	Minimal Sampling 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 50 Hz [1].

The TSP consumer may ask to any TSP provider for the list of symbols he provides and other kind of information like its base frequency. As soon as the consumer obtains 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 asks 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 and no meta-data.

2.2. TSP communication channels

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

2.2.1. 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 commands, called TSP requests, the network link between both sides may have been down without any impact.

Each time a TSP consumer sends a TSP request to a TSP provider, the consumer receives a TSP answer from the provider specifying the way 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 follows:

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

The current TSP implementation uses ONC RPC [4] as a command link interface. In fact TSP is open (by design and thanks to the current C implementation) to different kinds of request handlers. A foreseen TSP improvement in this area is to implement several request handlers kinds:

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

2.2.2. Data channel: TSP stream

The TSP data channel is used for efficient "sampling data flow" as soon as the consumer asks for the 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 side to consumer side. For example, if a consumer asks for 5 double precision symbols produced at 16Hz, the data link will contain 5x8x16 = 640 bytes/second (double precision values are 8 bytes values). Since both provider and consumer knows what the content of the TSP_request_sample was, there is no need to transport meta-data. Data sent by the data channel are XDR [5] encoded in order to avoid byte ordering troubles between provider and consumer. XDR is used by ONC RPC [4], we may have used CDR which is used by CORBA-GIOP [12]. Note that using other data channel encodings may be added to TSP very easily since TSP_request_information and TSP_request_sample contain a "requested features" bitset with unused place which may be used to negotiate forthcoming features.

The sample request dialog phase enables the TSP provider to precisely specify to the TSP consumer (*prior to send any sample data*) the exact order of the raw data it will receive. The provider describes this data sequence in TSP_answer_sample [2], so that when the consumer reads the data flow he only has to pop the data in the specified order. This minimizes the work at both sides during sampling.

3. 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 simple Java consumer has been written in no more than a hundred lines, comments included. TSP is simple but its technical content is rich since TSP is meant to be a technical development and an experimentation framework.

The technical objectives of TSP are described hereafter.

3.1. Portability

From its specification through the different implementations TSP aims at the best source portability, in order to be usable on a wide range of hardware platforms.

The portability elements of TSP are:

- ANSI C and Java API
- Platform independent data stream (XDR)
- Multi-threaded model
- Standards based : POSIX (thread, semaphores, . . .), XML, TCP/IP, XDR, ONC RPC

- Multi-protocol (ongoing work): CORBA, XML-RPC, SOAP
- Multi-platforms: Solaris 32 et 64bits, OSF, Linux 2.4/2/6, VxWorks (provider), Java (consumer), Win32 (ongoing work).

3.2. Robustness

TSP is uses simple concepts which should be selftested in the TSP SDK itself. TSP should be robust, since it is meant to be used in long run simulations (from several hours to many weeks).

3.3. Efficiency

Efficiency was a primary goal since TSP first design. Resource requirements during sampling, are minimized in terms of network bandwidth and CPU cycles on both consumer and provider sides.

3.4. Maintenance

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

Initial industrial partners took the road to "Open Source" in order to help TSP spreading outside the company. Free software is a source of motivation for TSP stakeholders (developers and other contributors). TSP is hosted at the FSF project hosting solution: Savannah (http://savannah.nongnu.org/projects/tsp)

4. TSP objectives

TSP is a multi-purpose project.

4.1. A standard sampling protocol

TSP may become a standard data sampling protocol.

4.2. Wide range of application

- Real-time or accelerated simulation

- Test beds / test benches
- Evolutionary phenomenon monitoring (real or simulated)
- Data format translation

5. Forseen developments & evolutions

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 contributions are welcomed, however the preferred way is to join the TSP Savannah registered team in order to avoid redundant developments.

The foreseen improvements (some of them already on-going) are:

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

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

- Implement several TSP requests handlers (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 an XDR writer (TSP consumer).

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

- Implement an CDF writer (TSP consumer) and CDF reader (TSP provider).

  Technical content: CDF [14], C and/or Java programming.

- Win32 Port.

  Technical content: Pthread-Win32, C programming with Win32 API . . .

- TSP provider lib in Java.

Technical content: Java 1.4, thread, . . .

- TSP consumer GUI in Java.

  Technical content: Java 1.4, thread, Swing/SWT . . .

- TSP data stream over UDP.

  Technical content: UDP unicast and UDP multicast

- Scripting API for TSP.

  Technical content: SWIG, python, perl, ruby, tcl . . .

- Asynchronous Remote Read/Write.

  Technical content: TSP Core, rpc, pthread . . .

- Synchronous Write.

  Technical content: TSP Core, pthread . . .

A. Contributors and helpers

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

Here is an incomplete list of contributors: Cesare BERTONA (The tche), Frédéric DEWEERDT (Just do it), Yves DUFRENNE (Do it more simple), Stéphane GARAY (gtk wizard), Stéphane GALLES (Java guru), Jérôme LACHAIZE (Studio TSPixWorks), Eric NOULARD (Big brain), Robert PAGNOT (Do It More Complex), ...

Here is an incomplete list of helpers: Julien BRUTUS, Jean-Paul CHAUMIER, José DEKNEUDT, Christophe LAPLUME, Rachid TALAALOUT, Alexandre TARAYRE, Sébastien GUILBAUD, Stéphane CANAVY, Pedro F. GIFFUNI, ...

Bibliography

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

[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 (http://www.rfc-editor.org/rfcsearch.html) .*

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

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

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

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

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

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

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

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

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

[14] *The NSSDC Common Data Format: http://nssdc.gsfc.nasa.gov/cdf/cdf_home.html (http://nssdc.gsfc.nasa.gov/cdf/cdf_home.html).*

Notes

1. Minimal sampling period is 2 out of a 100Hz base frequency provider
2. Doxygen: http://www.doxygen.org" Doxygen for example