TARGET COMMUNICATIONS FRAMEWORK: ONE LINK TO RULE THEM ALL

Anna Dushistova
anna.dushistova@gmail.com

PROBLEM

- Almost every device software development tool has its own method of communication with target system
- Communication methods often require individual setup, configuration and maintenance, and add unnecessary constraints to the toolchain

SOLUTION

"One link to rule them all,

One link to find them,

One link to bring them all

And in the ether bind them..."

Target Communications Framework

- Universal, extensible, simple, lightweight, vendor agnostic framework for tools and targets to communicate for purpose of debugging, profiling, code patching and performing other device software development tasks
- Single configuration per target (not per tool per target as today in most cases), or no configuration when possible
- Small overhead and footprint on target side
- Transport-agnostic channel abstraction
- Auto-discovery of targets and services

HISTORY

- **2007-10-25**
  - WindRiver Systems makes initial contribution of code to the Eclipse Target Management project
- **2008-06-25**
  - Initial release. Releasing core protocol documentation, initial Java framework, C agent and examples, RSE integration, and debugger integrations for CDT and TCF
- **2011-06-22**
  - 0.4 release. Adding Terminal service, disassembly, watchpoints. Initial addition of Python binding and Target Explorer
- **2012-06-27**
  - 1.0 release. Independent build and downloads. Promotion of Python binding and Target Explorer. Addition of Lua shell to agent

WIND RIVER | intel

XILINX®

montavista

SOME COMPANIES BEHIND THE PROJECT

WIND RIVER

intel

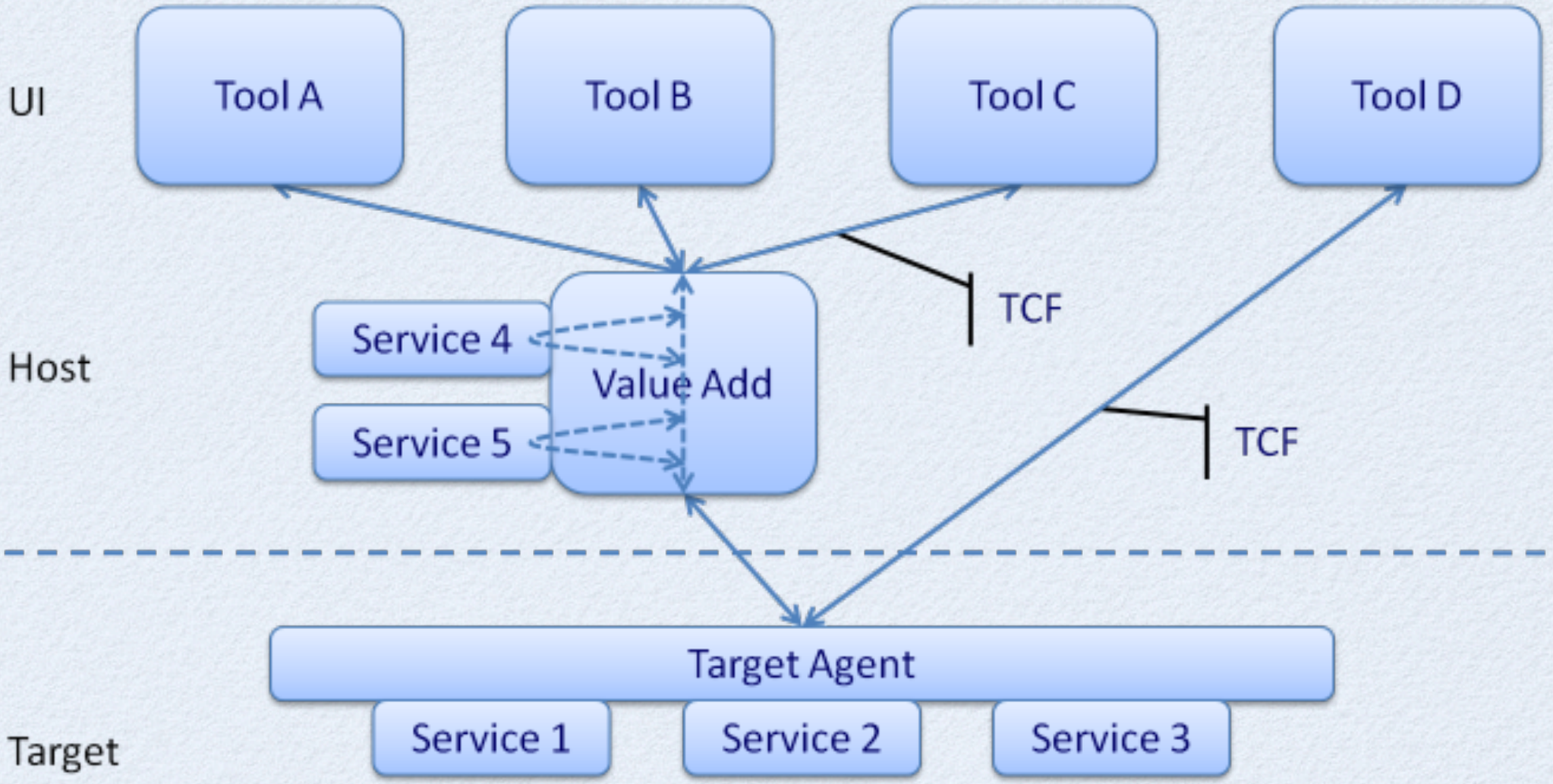
XILINX®

montavista

LICENSING

- All TCF code is licensed under the Eclipse Public License (EPL) v1.0
- C agent code is also licensed under the Eclipse Distribution License (EDL) v1.0 (dual-licensing)

ARCHITECTURE



DEFINITIONS

- ***Peer:***
  - Communication endpoint. Both hosts and targets are called peers. A peer can act as a client or a server depending on the services it implements.
- ***Service:***
  - Group of related commands, events and their semantics define a service. A service can be discovered, added or removed as a group at a communication endpoint.
- ***Channel:***
  - Communication link connecting two endpoints (peers). A single channel may be used to communicate with multiple services. Multiple channels may be used to connect the same peers, but no command or event ordering is guaranteed across channels.

COMMUNICATION PROTOCOL

- Defines data packet properties and roles common for all services
- Defines contents of a part of a packet; the rest of the packet is treated as array of bytes
- Provides:
  - Multiplexing – opening multiple channels per peer
  - Proxy – packet forwarding on behalf of other hosts

PACKET TYPES

- commands (requests)
  - ⇒ C • <token> • <service name> • <command name> • <byte array: arguments>
- results (responses)
  - ⇒ N • <token> •
  - ⇒ R • <token> • <byte array: result data>
  - ⇒ P • <token> • <byte array: progress data>
- Result packets start with the string “P” for intermediate result, “R” for final result, and “N” if the command is not recognized.There should be exactly one “R” or “N” result for each command

PACKET TYPES (CONTINUED)

- Events
  - ⇒ E • <service name> • <event name> • <byte array: event data>
  - Events are used to notify clients about changes in peer state. Services should provide sufficient variety of events for clients to track remote peer state without too much polling
- Flow Control Events
  - One side of communication channel might produce messages faster than they can be transmitted over the channel or can be consumed by another side. This will cause channel traffic congestion (flooding). Clients can use flow control events to implement advanced techniques to handle traffic congestion; for example, message coalescing, switching to less detailed messages, etc.
    - ⇒ F • <int: traffic congestion level> •

AUTO-DISCOVERY

- The Locator service uses the transport layer to search for peers and to collect data about the peer's attributes and capabilities (services)
- The only required service

What is already there?

- A plain-C implementation of a lightweight extendable target agent
- Java client API (usable standalone or on top of Eclipse)
- Python and Lua client APIs
- A complete debugger UI implementation in Eclipse
- A CDT integration for debugger launching
- A Target Management / Remote Systems Explorer integration for file system and process browsing
- Target Explorer – A lightweight UI for remote file system and process browsing, terminal access and debugger launch
- Documentation and usage examples

TCF Agent - Available Services

- Locator Service
- Memory Service
- Processes Service
- Run Control Service
- Registers Service
- Stack Trace Service
- Breakpoints Service
- Memory Map Service
- Path Map Service
- File System Service
- System Monitor Service
- Terminals Service
- Streams Service
- Disassembly Service
- Context Query Service

DEMO SETUP

- Host:
  - Yocto standalone cross toolchain 1.2.1 for x86(32 bit)
  - Eclipse IDE for C/C++ Developers package Juno SR2
- Target(qemu x86):
  - Stable kernel 3.6.2 from kernel.org
  - Yocto core-image-sato-sdk filesystem image for x86 qemu
  - Latest TCF agent

References

- http://wiki.eclipse.org/TCF
- http://git.eclipse.org/c/tcf/org.eclipse.tcf.git/plain/docs/
- tcf-dev mailing list: http://dev.eclipse.org/mhonarc/lists/tcf-dev/