A Proposal for Modifying TCI-CH Interface to Facilitate Implementation of Decentralized Self-organizing TTCN-3 Test Platform

Min Shan, Xianrong Wang, Xingming Ye, Lili Guo, Lijun Zhao
College of Computer Science, Inner Mongolia University
Huhhot, 010021, P.R. China
{csshanmin, cswxr, xmy, cszhaolj,csguoll}@imu.edu.cn

(present at T3UC 2010)

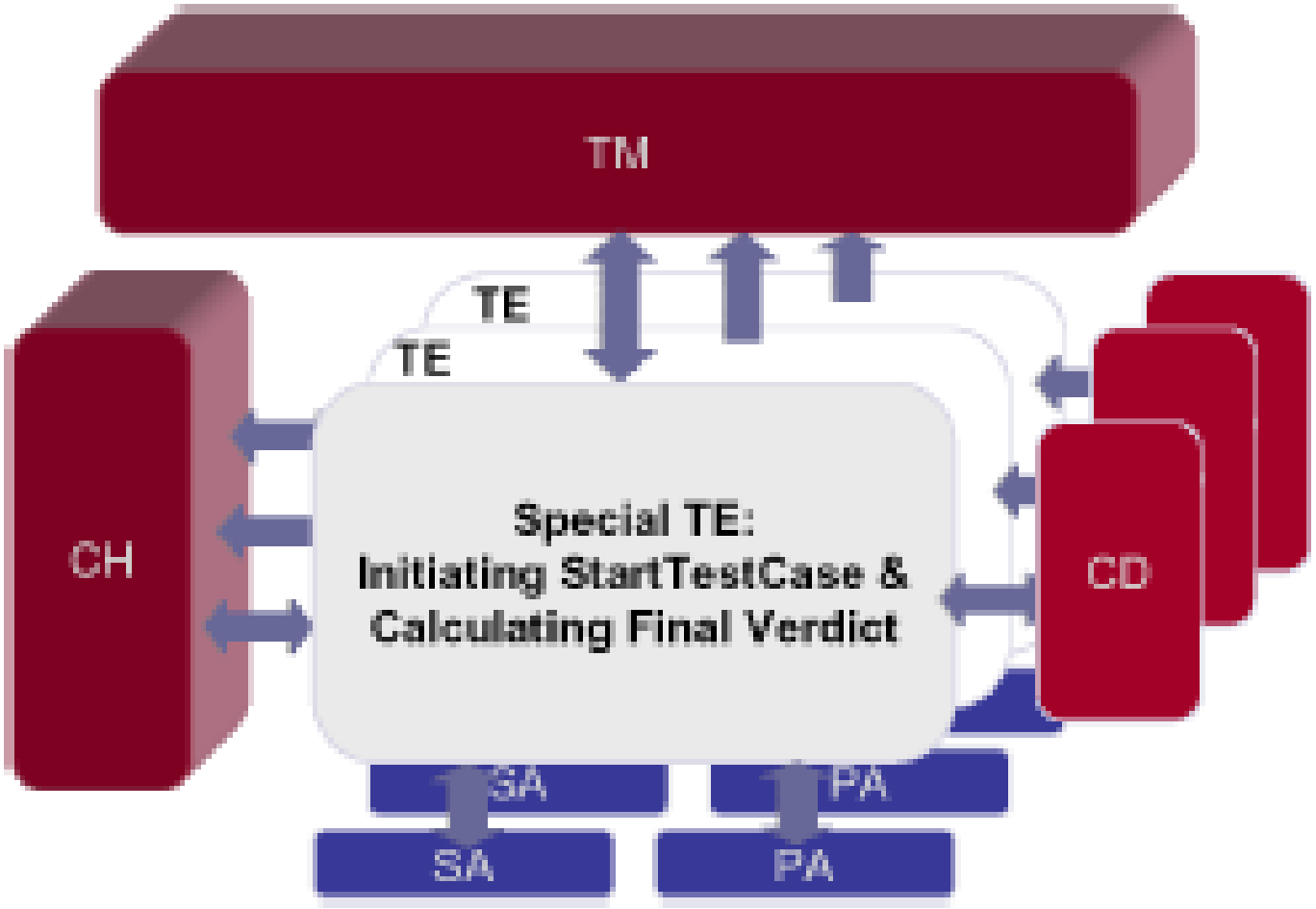
abbreviation

- SUT **S**ystem **U**nder **T**est
- ETS **E**xecutable **T**est **S**uite
- TE **T**est **E**xecutable
- CH **C**omponent **H**andling
- TCI **T**TCN-3 **C**ontrol **I**nterface
- MTC **M**ain **T**est **C**omponent
- PTC **P**arallel **T**est **C**omponent

Motivation

- CH : handling component operation
- CH interactive with TE through standard **TCI-CH Interface**
  - TCI-CH provided
  - TCI-CH required

Motivation



The distributed test system architecture defined in TCI standard

Motivation

- What will happen when executing some test suites for **testing** some **high throughput** systems from industry?
  - a large number of test nodes are required
  - time-consuming

Motivation

- drawback of a **centralized** test platform containing **a large number of** test nodes using in **time-consuming** test suite execution:
  - scalability problem
- solution: **decentralized self-organizing** TTCN-3 test platform
- modifications on TCI-CH interfaces

Outline

- the modifications on TCI-CH interface to:
  - solve a problem on **taking distributed snapshot** when executing ***done*** and ***killed*** operation
  - represent ***all component*** and ***any component***
  - **apply test nodes** for executing a test suite
  - **release test nodes** after finishing test suite execution

Taking Distributed Snapshot

```
testcase tc1 tc1 runs on MTC_Type system SYS_Type
{
    var ptctype c1, c2, c3;
    …
    alt
    {
        []ptc1. killed
        {
            …
        }
        []any component.done
        {
            …
        }
    }
    …
}
```

Taking Distributed Snapshot

- side effect
- applicable solutions for **taking distributed snapshot**:
  - CH ensures that the status of all test components **do not change** during taking distributed snapshot operation
  - local TE **fetches the status** of relevant test components from CH, and the match operations are based on the status.

Taking Distributed Snapshot

- drawback of existing TCI-CH interface for taking distributed snapshot:
  - TCI standard has not define any operation for TE to inform starting and finishing taking distributed snapshot
  - the signature ***tciTestComponentKilledReq*** and ***tciTestComponentDoneReq*** can not **directly** get the status of test components. **Incorrect judgment** may be caused by **using the two operations together** to judge the status of test components if the status of the component **changes between the two operations**

Taking Distributed Snapshot

- alternative modification :
  - add new signatures to inform CH starting/finishing taking snapshot
    - TCI-CH provided:
      void *tciStartTakingDistributedSnapshotReq*()
      void *tciFinishTakingDistributedSnapshotReq*()
  - add a new signature to **directly** fetch the status of test component
    - abstract data types:
      tciComponentStatusType
    - TCI-CH provided:
      tciComponentStatusType *tciGetTestComponentStatus*(
      TriComponentIdType component)

Represent *all component* and *any component*

- **any** and **all** refer to PTCs only, i.e. the MTC is not considered. (TTCN-3 core language v4.1.1, Table 20, page152)
- Who knows all of PTCs? (local TE? tastcase? CH?)
- **local TE** only knows the PTCs **deployed at** or **created from local** test node

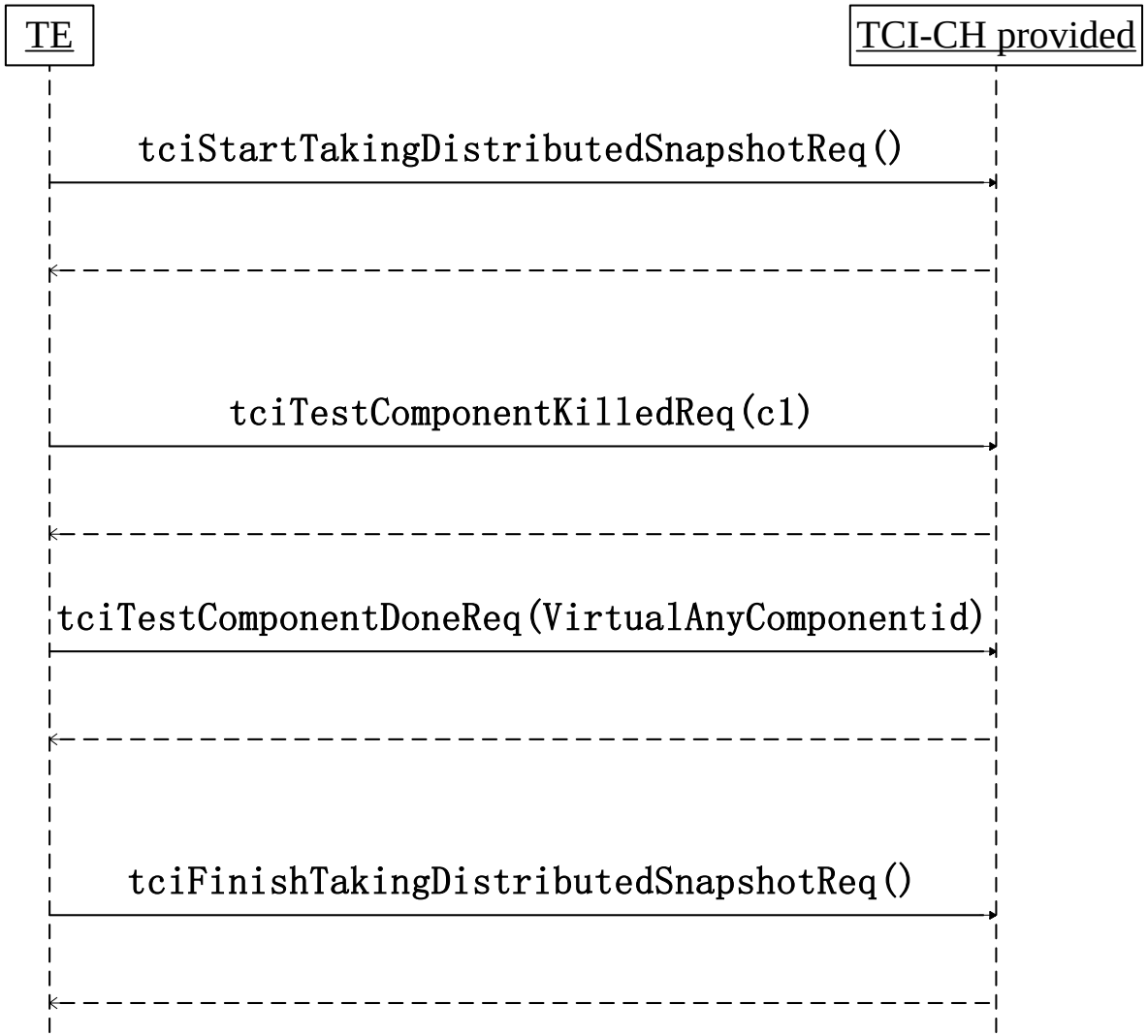
Represent *all component* and *any component*

- *all component* & *any component* are only allowed using from **MTC** (TTCN-3 core language v4.1.1, Table 20, page152)
- Does **testcase** know all of PTCs?
  - Components can be created at any point in a behaviour definition providing full flexibility with regard to dynamic configurations (i.e. **any component can create any other PTC**) (TTCN-3 core language v4.1.1, page145)
- **testcase** does **not** know PTCs **created by other PTCs**

Represent *all component* and *any component*

- only **CH** knows all of PTCs
- when executing an configuration operation for *all PTCs* or *any PTCs*, **TE should inform CH** that the operation is for all PTCs or any PTCs, so that CH can dispose the operation into multiple operations, each of which is for an individual PTC
  - introduce two **virtual component IDs** in TCI:
    - **VirtualAllComponentid**
    - **VirtualAnyComponentid**

```
testcase tc1
    runs on MTC_Type
    system SYS_Type
{
  var ptctype c1, c2, c3
  …
  alt
  {
    []ptc1. killed
    {
       …
    }
    []any component.done
    {
       …
    }
  }
  …
}
```



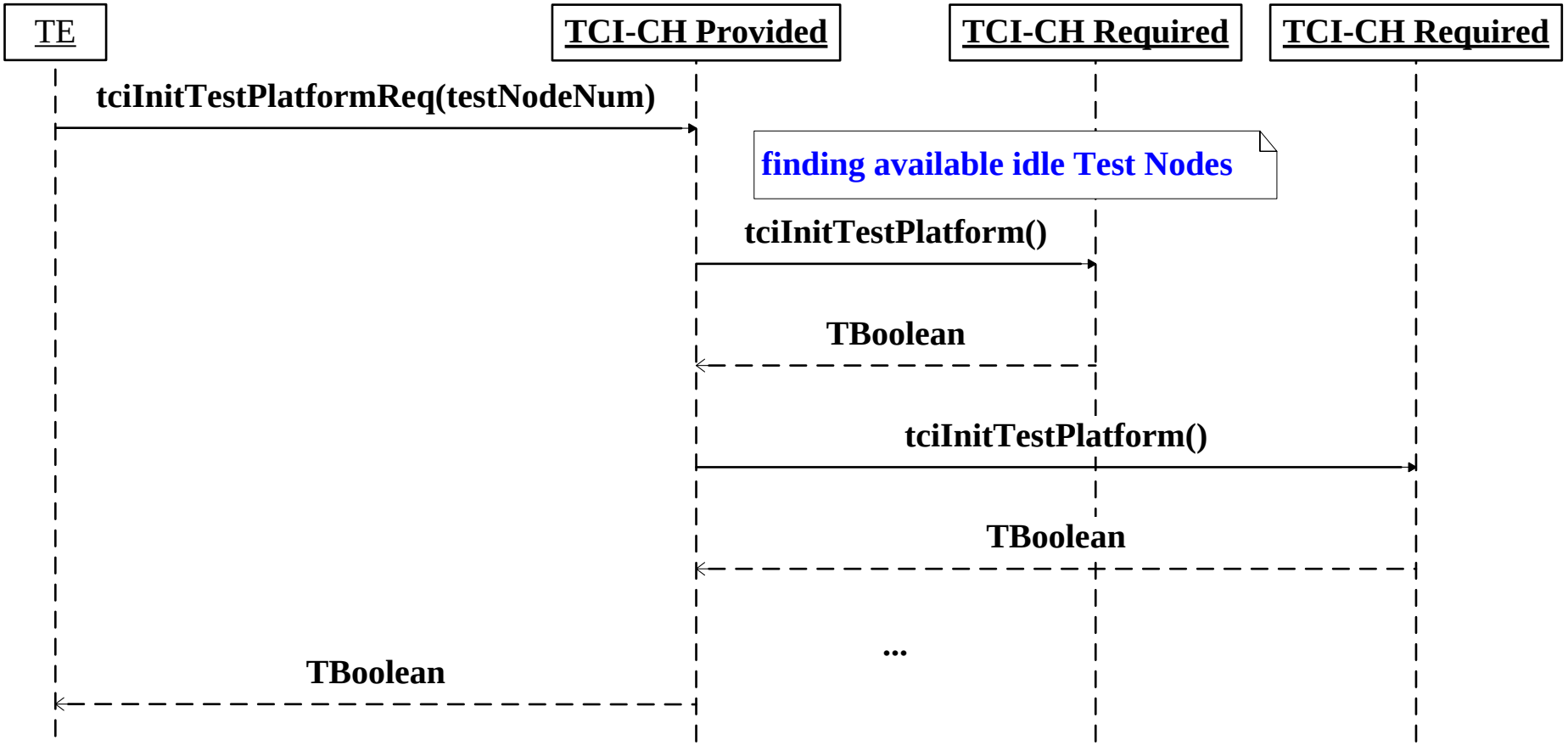
Apply Test Nodes

- the extended signatures for applying test nodes when initialization
  - TCI-CH provided:

    TBoolean *tciInitTestPlatformReq* (in TInteger TestNodeNum)
  - TCI-CH required:

    TBoolean *tciInitTestPlatform* ()
- **idle**&**busy** test node

Apply Test Nodes



Use scenario – apply test nodes when initialization

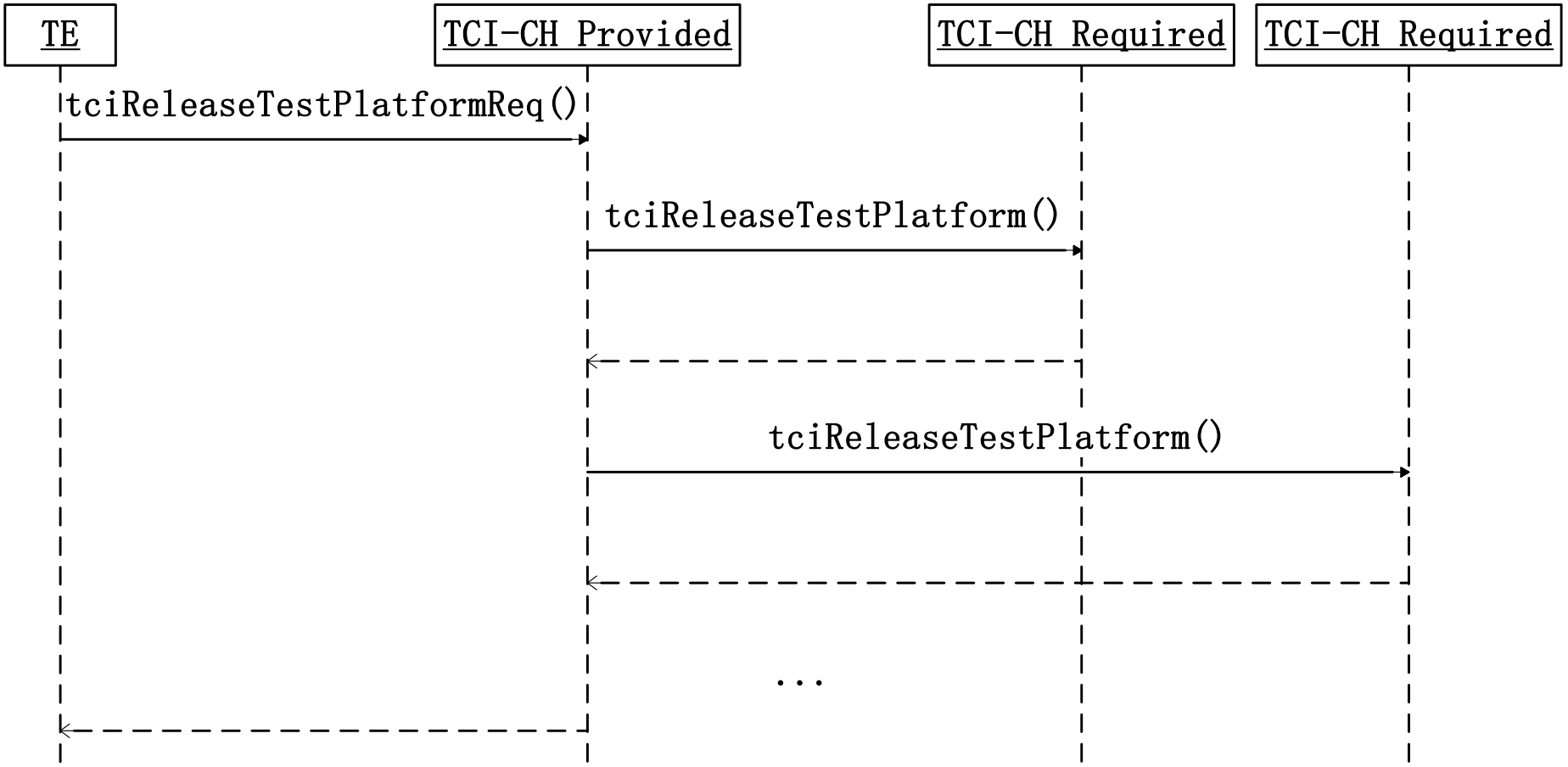
Release Test Nodes

- the extended signatures for releasing test nodes after finishing test suite execution
  - TCI-CH provided:

    void *tciReleaseTestPlatformReq* ()
  - TCI-CH required:

    void *tciReleaseTestPlatform* ()

Release Test Nodes



Use scenario - release test nodes after finishing test suite execution

two open questions

- Are the TCI-CH required operations for alive/running/done/killed necessary? (why do not storing status in CH)
- Why most of operations in TCI-CH interface (such as operations for *start*, *connect*, *send*) have no return value, whereas an operation in TRI often returns a value of *TriStatusType* to indicate whether the execution of the operation is successful or failed

Thank you!

Any Question?