

Towards an IP-oriented Testing Framework

the *IPv6 Testing Toolkit*

Ariel Sabiguero^{1,2} Anthony Baire² Alexandra Desmoulin²
Annie Floch² Frédéric Roudaut² César Viho²

¹Instituto de Computación, Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay
asabigue@fing.edu.uy

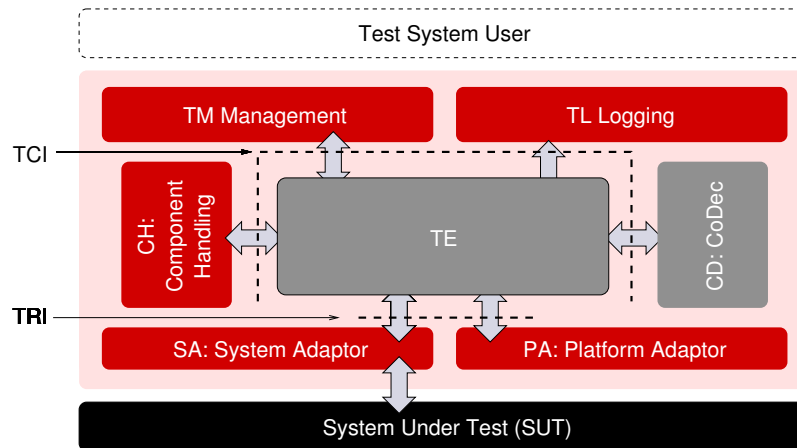
²IRISA
Campus de Beaulieu
35042 Rennes CEDEX, France
{asabigue, abaire, adesmoul, afloch, roudaut, viho}@irisa.fr

01/06/2006

Outline

- 1 Introduction
- 2 Using Ethereal for CoDec development
- 3 CoDec development
- 4 The CoDec Generator
- 5 IPv6 Testing Toolkit
- 6 Summary

Core complexity addressed



Underlying IPv6 protocol requirements

Technical Requirements

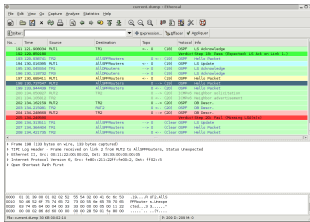
- Bit-oriented handling of content parts
- Complex messages (need to process the whole Ethernet frame)
- Unknown message size
- Unknown number of options in messages
- Unknown order of options in messages
- Process unexpected messages

Operational Requirements

- Flexibility
- Reliability

Using Ethereal: the benefits

Why ?



- To avoid the complexity of decoding packets
- Already existing tool
- Extensively used in the IP world
- Maintained by a big community of users
- Evolved by it's users
- Validated
- ... it is Open/Free

Interfacing with Ethereal

Input

- Packets were already in TRI Binary String format...
- ... have to be converted back into some Ethereal format

Output

- PDML format selected
- XML packet representation
- XML has to be parsed... libxml
- ... new dependencies
- XML's content has to be processed

CoDec specific problems

CoDec development, integration and maintenance is an expensive task

- Manual synchronization of types in TTCN-3 and C++
 - **Lot** of redundancy between TTCN-3 and C++ code
 - C++ code is derived from TTCN-3 type definitions
- TCI-CD has a flat design
- TCI-CD data presentation is not handy

Handling TCI Values

Example: decoding and instantiating an 8-bit field

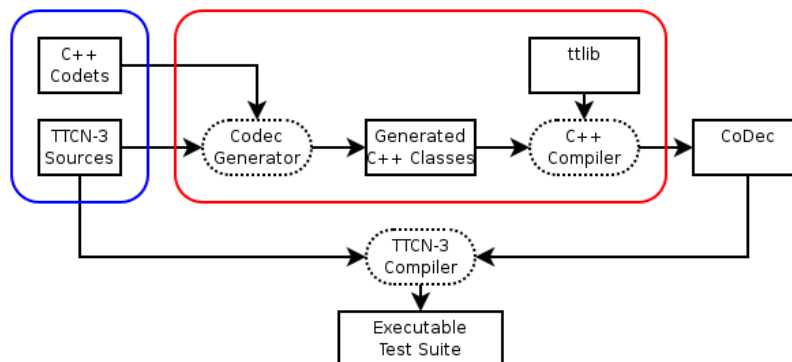
```
integer    => tciSetIntAbs      (val, "42");
           tciSetIntSign      (val, 1);
octetstring => tciSetOStringValue (val, "'2A'O'");
bitstring  => tciSetBStringValue (val, "'00101010'B'");
charstring => tciSetHStringValue (val, "\"B\"");
...
```

- Not natural ways of manipulating data in C
- Must know the type of **every** field (and subfield)

CoDec motivated decisions

- Develop a library that provides a comfortable framework for TTCN-3 data handling from C++
 - Develop a tool that extracts available type information from TTCN-3 code
 - Convert most of the CoDec development into an automatic task
 - Provide automatic way to integrate missing logic into the CoDec
- => The CoDec expert would never write the CoDec directly

Build Process



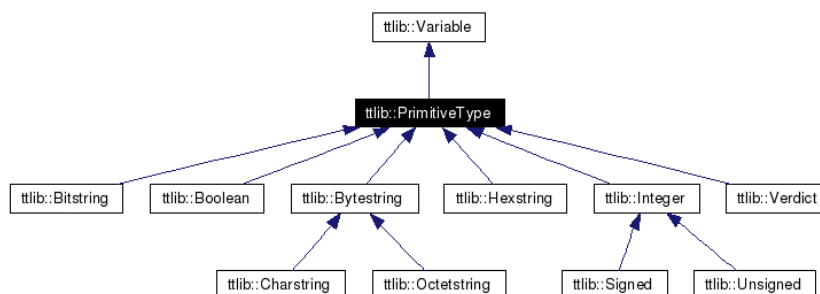
ttlib: the corner stone for the CoDec generator

The testing toolkit library provides:

- an implementation of TCI-CD (validated with tools from 2 vendors)
- a framework of C++ classes for manipulating TTCN-3 data with:
 - different levels of abstraction
 - default encoding/decoding mechanisms
 - codec exception handling & debugging functions
 - hooks for integrating custom *codets*

ttlib: a type handling helper

Basic type hierarchy



Codets

```

type union ICMPv6OptionSingleType {
    SLLOptionType          SLLOpt,
    MTUOptionType          MTUOpt,
    .....
}

void ICMPv6OptionSingleType::PreDecode (Buffer& buffer)
    throw (DecodeError) {
    UInt8 type;
    int position = buffer.GetPosition();
    buffer.Read (type, 8);
    buffer.SetPosition (position);
    SetHypChosenId (map_icmpv6_opttype.GetValue(type));
}

```

Codet usage

Codets may be used for:

- making decisions in the decoder
 - variable size contents (string, record of, ...)
 - union variants / optional fields
- additional processing on binary contents (eg. checksum, ...)
- handling exceptions

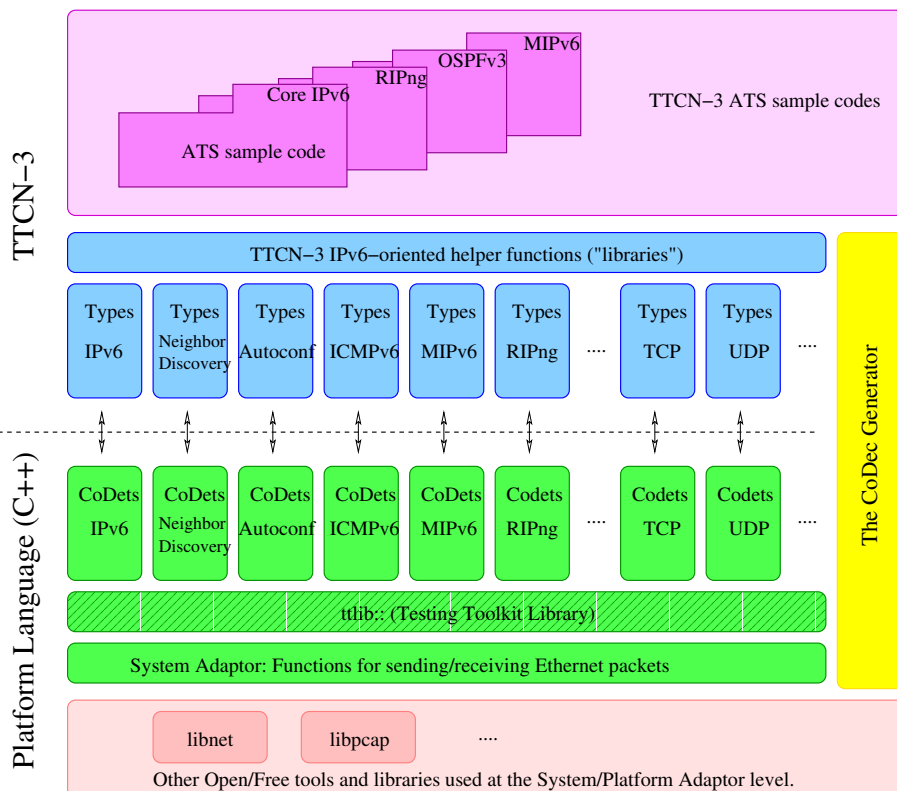
Codets may be inserted in:

- the decoder: PreDecode, PreDecodeField, PostDecodeField and PostDecode
- the encoder: PreEncode, PreEncodeField, PostEncodeField and PostEncode

Toolkit description

Our "IPv6 Testing Toolkit" is a set of data, functions and basic mechanisms dedicated to TTCN-3 test development and execution of IPv6 test suites

- *Off-the-shelf* data types and functions
- Adequate level of abstraction
- Easily extensible
- Tool provider independent (C++ platform language only now)



Summary

- We presented IRISA's "*IPv6 Testing Toolkit*"
- Spot and propose solutions into grey areas of TTCN-3 execution issues
- The CoDec Generator was presented, which automatises CoDec development task
- We produced tools that allows adequate software engineering practices in test suites development in our laboratory

Thank you for your time

Questions?

Paper available in the proceedings or at:

<http://www.irisa.fr/tipi/publi/t3uc2006paper.pdf>