A Systematic Test Case Generation Approach for Testing Message Length Variability

Kyungmin Go
Command & Control
R&D Lab
LIGNex1
Seoul, South Korea
kyungmingo@lignex1.com

Sungwon Kang
Department of
Computer Science
KAIST
Daejeon, South Korea
sungwon.kang@kaist.ac.kr

Myungchul Kim
Department of
Computer Science
KAIST
Daejeon, South Korea
mck@kaist.ac.kr

Jihyun Lee
Department of
Computer Science
KAIST
Seoul, South Korea
jihyunlee@kaist.ac.kr

Abstract— **Variable length messages have been in use for a long time for efficient delivery of information. As there are many different ways, with varying complexity, to utilize message length variability, it is crucial to thoroughly test the capability of the parsers of such messages to ensure that they correctly handle the variability. However, testing techniques for message length variability were developed only in fragments in the past and therefore test developers who are faced with the task of testing variable length messages are left with little guidance or few techniques for handling them systematically. This paper develops an approach for systematic test cases generation for testing the parsers of variable length messages. To do so, we develop taxonomy of message length variabilities, and derive test requirement patterns for them using the taxonomy. Then test requirements for particular protocols can be derived from the patterns. A case study is conducted to show that the proposed taxonomy and the test requirement patterns are effective in deriving test cases for actual protocols. The results showed significant improvement over the conventional approach. It revealed many missed requirements that were not identified with the test requirements developed using the conventional intuition-based approach.**

Keywords- message length variabiltiy; test requirements; test case generation

1. Introduction

Variable length message is a means of sending data in an optimal way so that communication protocols use network bandwidth minimally by utilizing message length variability. Due to this characteristic, variable length messages were used in various communication protocols such as IPv6 [1], Variable Message Format (VMF) [2], Asynchronous Transfer Mode (ATM) [3], Resource Reservation Protocol (RSVP) [4], ROHC [5] and etc. Message length variability can be exploited in many different ways. For example, IPv6 allows the number of message headers to be minimal by using variable number of extension headers, and in VMF [2] only fields that contain relevant information and data are transmitted by using Field Presence Indicator.

However, while variable length messages are used in many communication protocols, there was little research that addressed testing of message length variability.

As the surveys in [6] show, message length variability testing was treated only in fragments in communication protocol testing. For example, the ATM protocol testing carried out in [7], which was based on ATM Forum test standards, did not address message length variability systematically. The work by Lee et al. [8] covers communication protocol testing based on Finite State Machine (FSM). However, it didn't consider message features of the protocol and therefore the possible message length variations are not mentioned for testing a FSM of the protocol. The work related to network messages variability appear in [9, 10], but they address testing of variable length messages only in fragments. So for example no classification of message length variability is attempted and no treatment at the message fields is conducted. As another example, VMF messages used in the military domain, the Redondo System Company makes a VMF Test VTT program for message-level message length variability test [11], but its test generation approach is not known.

For the systematic testing of variable length messages, the variability of variable length messages should be prescribed with systematic taxonomy, and proper test requirements should be extracted from taxonomy. Previous work on communication protocol testing failed to explicitly address test case generation for message length variability either by considering only fixed length messages or by not systematically dealing with message length variability of the protocol.

Therefore, development of a systematic test generation method to ensure the correct operation of existing and new communication protocols using variable length messages is called for. However, there are two major challenges in test generation of variable length messages.

The first major challenge is that the length of a message is variable due to the presence fields and the recurrence fields for exchanging only the required information. The second major challenge is that message length variabilities are specified with context dependent rules as well as with context independent rules.

In this paper, we propose a systematic approach to testing message length variability.As the first step of the approach, we classify patterns for defining message length variability, relying on a survey of the methods used in communication protocols, thereby developing taxonomy of message length variabilities. As the second step of the approach, based on the taxonomy, test requirement patterns are derived. As the third step of our approach, the test requirements patterns are used to derive test requirements for particular protocols. Then finally actual test cases can be derived by using the common methods for test case derivation.

The key aspect of our approach is using patterns as ways of systematizing test requirements derivation and test cases derivation. As C. Alexander said in [12], a pattern is a description of a recurring problem and its solution in such a way that the solution can be used again and again. As F. Bushman et al. said in [13] this kind of thinking in problem-solution pairs is common in many different domains such as architecture, economics, and software engineering as it is a natural way of coping with any problem. This paper is an attempt to solve the problem of testing message length variability along this line of thinking as when there is no known general automatic solution, the best we can do is to systematically utilize experimental knowledge.

For the problem of testing message length variability, to our knowledge our work is the first systematic approach to solving it. The proposed taxonomy and the test requirement patterns of the approach are effective in deriving test requirements for actual protocols. Our approach provides very thorough test coverage and reveals errors related to message length variability that are not easily detected by the conventional intuition-based approach. This is demonstrated with a case study by applying our approach to testing of VMF parsers and comparing it with the conventional approach. The case study comparison results show that with our approach the numbers of test requirements and detected errors have almost doubled compared to those of the conventional approach.

The remainder of the paper is organized as follows. In Section 2, we present our approach to message length variability testing. In Section 3, taxonomy of message length variability is provided. In Section 4, as the stepping stone to test cases generation test requirement patterns are derived based on the taxonomy. In Section 5, for a case study our approach is applied to test cases generation for VMF parsers and is compared with the conventional approach. In Section 6, we discuss related works. Finally, Section 7 concludes the paper.

2. Approach to Message Length Variability Testing

Our approach to message length variability testing consists of two phases as depicted in Fig. 1. In the first phase, a theory is developed that is to be used in the second phase of actual test case derivation. For that, first, the taxonomy of message length variability is developed (T1). The taxonomy covers all of the message length variability patterns in communication protocols known to us to date. Then protocol independent test requirement patterns are derived from the taxonomy (T2). We call these two together *a theory* because it is applicable to test cases derivation of any protocols as long as they have message length variability.

In the second phase, actual test cases for particular communication protocols are derived. Based on the protocol independent test requirement patterns resulting from the first phase and the particular communication protocol at hand that allows message length to vary, test requirements specific to the protocol are first derived (E1). From the test requirements, using test case generation methods which suitable with protocol properties, protocol specific test cases are generated (E2). Then, finally, the test cases are applied to the parsers of the variable length messages to check their correctness with respect to their ability to interpret the message lengths (E3).

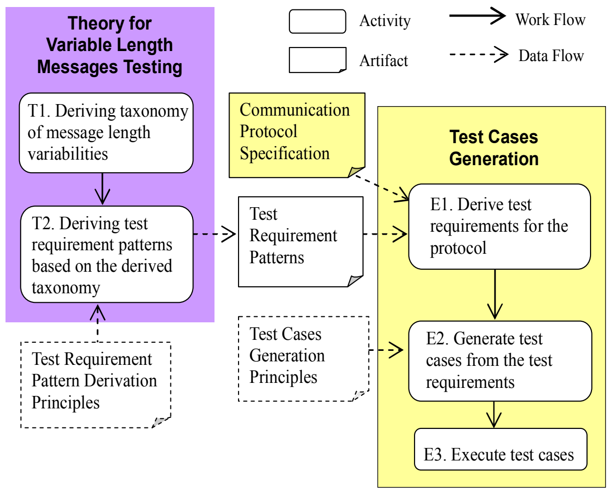

Theory for Variable Length Messages Testing
Activity
Artifact
Work Flow
Data Flow
T1. Deriving taxonomy of message length variabilities
T2. Deriving test requirement patterns based on the derived taxonomy
Communication Protocol Specification
Test Requirement Patterns
Test Cases Generation
E1. Derive test requirements for the protocol
E2. Generate test cases from the test requirements
E3. Execute test cases
Test Requirement Pattern Derivation Principles
Test Cases Generation Principles

Figure 1. The proposed approach to message length variability testing

The two dotted lined boxes in Fig. 1 represent the principles to be used for deriving test requirement patterns and generating test cases. The two sets of principles can be determined by the users of our approach as deemed appropriate.

3. Taxonomy of Message Length Variability Specification Mechanisms

In general, a message consists of a sequence of fields and can have a hierarchical structure. Hence, a field may have its subfields and a subfield in turn has a substructure in a recursive manner. A set of fields or subfields that have the same parent field or subfield is said to be at the same level. So the message itself is at level 0, its fields are at level 1, the immediate subfields of the fields at level 1 are at level 2, and so on. (Therefore the, subfields cannot be at level 1 since only fields that are not subfields of any field can exist at level 1.) In this paper, we use the term *field* to mean field or subfield.

A field can be classified as an *indicator field* or a *subsequent subfield*. An indicator field specifies the existence or length or type of the immediately subsequent subfield or a block of fields. For example, in a VMF K05.1 message, which announces location info, the existence of an Elevation Field is indicated by the flag field that comes just before it. If the value of the flag is 1, then the Elevation Field should follow and if it is 0, then it should not follow. Also, in order to allow a field to occur repeatedly, there is a flag that represents field repetition.

In order to develop taxonomy of message length variability specification mechanisms, we first differentiate two classes of mechanisms for specifying variable length messages. One is context independent mechanisms and the other is context dependent mechanisms. In the case of context dependent mechanisms, the variability of message length is not expressed at once for the whole message except for the case of the message termination pattern.

In the remainder of this Section, different types of variability defined by context independent rules and context dependent rules are introduced, respectively, in Sections 3.1 and 3.2.

3.1 Context Independent Mechanisms for Specifying Message Length Variability

For context independent mechanisms of specifying length message variability, there are three mechanisms: one using an indicator, one using a rule, and one using a pattern. As mechanisms using an indicator, there are the *Field Indicator* (FI), *Cluster Indicator* (CI), and *Repetition Indicator* (RI). As a mechanism using a rule, there is the *Condition Rule* (CR). As a mechanism using a pattern, there is the *Field End Pattern* (FEP).

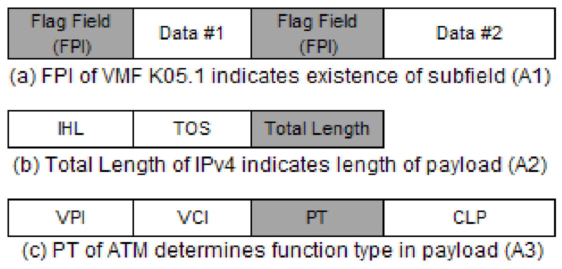
Fig. 2 gives a detailed description of these five mechanisms. In Fig. 2, '[]' indicates an optional field or subfield. The '*val*(I)' denotes the value of the indicator I, which is either 0 or 1 in the case of an existence indicator and any natural number in the case of length and type indicators. '$(F)^+$' and '(F)*' represent respectively one or more and zero or more repetitions of F. In the following, each mechanism is explained one by one.

1) Field Indicator

Field Indicator has three different kinds. They are either the *Field Existence Indicator* (FEI) or the *Field Length Indicator* (FLI) or the *Field Type Indicator* (FTI).

The FEI is used as a flag for presenting the existence of a subsequent subfield. If the value of FEI is '0', then the subsequent subfield should have no value. The FLI shows the length of the subsequent subfield. As is the case with FEI, if the FLI is 0, the subsequent subfield should have no value. If it is more than 0, the length of subsequent subfield should match it. The FTI shows a type of subsequent subfield. Even though the value of the subsequent subfield is only expressed with 0 and 1, it can be interpreted with different units such as 'inch' or 'meter'.

Fig. 3 shows an example of FI in VMF. In Fig. 3(a), VMF K05.1 FEI consists of 1 bit to indicate the existence of its subsequent subfield (*Rule A1*). In Fig. 3(b), FLI represents the length of the Subsequent subfield, depending on the total length field of IPv4. (*Rule A2*). In Fig. 3(c), FTI determines the type of its subsequent subfield, which defines the length and the data. Once the type is determined, the length and the way the value of the subsequent subfield is interpreted are fixed. Fig. 4 also shows another example of FLI in ATM protocol. The IE Length field defines the length of the IE data.


Flag Field (FPI)
Data #1
Flag Field (FPI)
Data #2
(a) FPI of VMF K05.1 indicates existence of subfield (A1)
IHL
TOS
Total Length
(b) Total Length of IPv4 indicates length of payload (A2)
VPI
VCI
PT
CLP
(c) PT of ATM determines function type in payload (A3)
Figure 3. VMF message examples with context independent structure

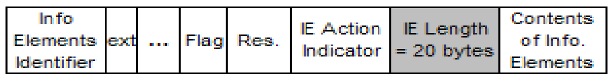


Figure 4. IE length field of the ATM protocol

A) Field Indicator (FI)

(A1) *Field Existence Indicator* (FEI) indicates *existence* of the subsequent subfield.

Usage Syntax: FEI + [DATA_FIELD]

→ *If val*(FEI) = 1 *then* DATA_FIELD *must* exist.

→ *If val*(FEI) = 0 *then* DATA_FIELD *must not* exist.

(A2) *Field Length Indicator* (FLI) indicates the *length* of the subsequent subfield.

Usage Syntax: FLI + [DATA_FIELD]

→ *If val*(FLI) = m > 0 *then* the *length* of the field DATA_FIELD *must* be m.

→ *If val*(FLI) = 0 *then* DATA_FIELD *must* not exist.

(A3) *Field Type Indicator* (FTI) indicates the *type* of the subsequent subfield.

Usage Syntax: FTI + [DATA_FIELD]

→ *If val*(FTI) = t *then* the *type* of the field DATA_FIELD *must* be t.

B) Cluster Indicator (CI)

Let CL = FEI + [DATA_FIELD] | FLI + DATA_FIELD | FTI + DATA_FIELD | DATA_FIELD
|CEI + CL* | CLI + CL* |CTI + CL$^+$

We call an instance of zero or more repetitions of CL a *cluster*.

(B1) *Cluster Existence Indicator* (CEI) flag indicates *existence* of the subsequent subfields.

Usage Syntax: CEI + CL*

→ *If val*(CEI) = 1 *then* CL* *must* exist.

→ *If val*(CEI) = 0 *then* CL* *must not* exist.

(B2) *Cluster Length Indicator* (CLI) indicates the *length* of CL*.

Usage Syntax: CLI + CL*

→ *If val*(CLI) = m > 0 *then* the *length* of CL* *must* be m.

→ *If val*(CLI) = 0 *then* CL* *must not* exist.

(B3) *Cluster Type Indicator* (CTI) indicates the *type* of the subsequent subfields.

Usage Syntax: CTI + CL$^+$

→ *If val*(CTI) = t *then* the *type* of CL *must* be t.

C) Repetition Indicator (RI)

(C1) *Field Repetition Indicator* (FRI) flag indicates *repetition* of a field.

Usage Syntax: (FRI + [DATA_FIELD])$^+$

→ *If val*(FRI) = 1 *then* DATA_FIELD *must* exist and 'FRI + [DATA_FIELD]' *must* be repeated.
If val(FRI) = 0, *then* its associated DATA_FIELD *must not* exist
and 'FRI + [DATA_FIELD]' *must not* be repeated any longer.

(C2) *Cluster Repetition Indicator* (CRI) indicates *repetition* of a cluster.

Usage Syntax: (CRI + CL*)$^+$

→ *If val*(CRI) = 1 *then* CL* *must* exist and '(CRI + CL*)' *must* be repeated.
If val(CRI) = 0, *then* CL* *must not* exist and '(CRI + CL*)' *must not* be repeated any longer.

D) Condition Rule (CR)

Condition Rule (CR) is defined with a condition indicator Cond and the fields for its consequents ConConsq$_i$.

Usage Syntax: Cond + CL$^+$ -- CL is ConConsq of Cond.

→ *If* Cond == case$_1$ *then val(*ConConsq$_1$) = x; *val*(ConConsq$_2$) = y; …
Else if Cond == case$_2$ *then val(*ConConsq$_1$) = z; *val*(ConConsq$_2$) = w; …
Else if …

E) Field End Pattern (FEP)

Field End Pattern (FEP) is a pre-defined data pattern that indicates *termination* of a field.

Usage Syntax: DATA_FIELD + FEP

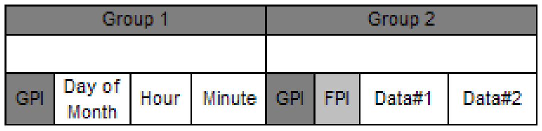
→ The given field ends by the *existence* of FEP regardless of its *length*.

Figure 2. Specification mechanisms for context independent message length variability

2) *Cluster Indicator*

A cluster is a block of fields. Typically the first field of a cluster is the indicator field and is called the *Cluster Indicator* (CI). It is either the *Cluster Existence Indicator* (CEI) or the *Cluster Length Indicator* (CLI) or the *Cluster Type Indicator* (CTI). The functionality of the CI is the same as that of FI with the only difference that CI has multiple subfields as a group.

Fig. 5 shows an example of CI for VMF K05.1. In Fig. 5, the Group Presence Indicator (GPI) is classified as CEI, which determines the existence of the subsequent subfields (Rule B1). The subsequent subfields should have values if the condition is 'GPI is 1' and none if the condition is 'GPI is 0'.



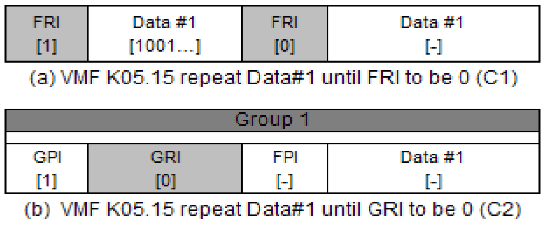
Group 1				Group 2			
GPI	Day of Month	Hour	Minute	GPI	FPI	Data#1	Data#2

Figure 5. GPI of the VMF K05.1 message

3) *Repetition Indicator*

The *Repetition Indicator* (RI) indicates whether a single field or a block of fields exists or not. If the *Field Repetition Indicator* (FRI) is '0', then it has a subsequent subfield, after which FRI does not appear again. If it is '1', then after the subsequent subfield appears, the FRI and its subsequent subfield should appear (*Rule C1*). Similarly, the *Cluster Repetition Indicator* (CRI) indicates repetition of a cluster (*Rule C2*).

Fig. 6 shows an example of the RI used for a VMF K05.1 message. In Fig. 6(a), the subsequent subfield, Data #1, is repeatedly inserted by setting the value of FRI. Through setting the first value of FRI to '1', it is indicated that Data #1 will be repeated. If the next occurrence is '0', there is no more repetition of the field. In Fig. 6(b), the FPI and Data #1 are the subsequent subfields of the preceding GRI. As with the case of FRI, the GRI field value of '1' indicates that its subsequent subfields are repeated, and a value of '0' indicates there will be no more repetitions.



FRI [1]	Data #1 [1001…]	FRI [0]	Data #1 [-]

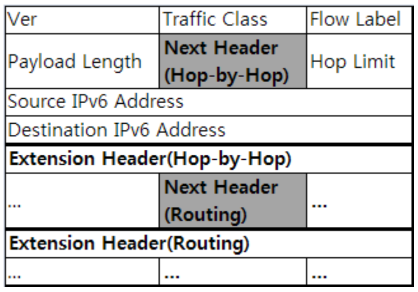
(a) VMF K05.15 repeat Data#1 until FRI to be 0 (C1)

Group 1			
GPI [1]	GRI [0]	FPI [-]	Data #1 [-]

(b) VMF K05.15 repeat Data#1 until GRI to be 0 (C2)

Figure 6. FRI and CRI

Fig. 7 shows another example of the use of repetition indicator in IPv6. The next header field of the IPv6 header is a kind of CRI field and is used to indicate which type of extension header follows after the current header. The number of following extension headers can be zero or more.



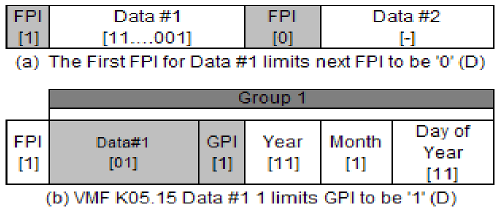
Ver	Traffic Class	Flow Label
Payload Length	**Next Header (Hop-by-Hop)**	Hop Limit
Source IPv6 Address		
Destination IPv6 Address		
Extension Header(Hop-by-Hop)		
...	**Next Header (Routing)**	...
Extension Header(Routing)		
...	...	...

Figure 7. Next Header Field in IPv6

4) *Condition Rule*

By defining certain conditions, values of some flag fields and subsequent subfields are constrained by the conditions. We call such message length defining mechanism the *Condition Rule* (CR). CR puts constraints on the relationship between two or more adjacent fields. CR is stated with if-then-else statements. One of the subsequent subfields or indicator fields specifies a condition, which constrains the values of other predetermined fields.

Fig. 8(a) shows an example of CR used for a VMF K05.1 message. The CR in VMF says that if the value of FI for Data #1 is '1', then the value of FI for Data #2 should be 0. Fig. 8(b) shows an example of a VMF K05.15 message. If the value of Data #1 is '01', the GPI value of Group 1 must be '1'. Therefore, it is a valid message satisfying this CR. RSVP protocol also uses FLI and FTI for specifying the length and type of the subsequent subfield.



FPI [1]	Data #1 [11….001]	FPI [0]	Data #2 [-]

(a) The First FPI for Data #1 limits next FPI to be '0' (D)

	Group 1				
FPI [1]	Data#1 [01]	GPI [1]	Year [11]	Month [1]	Day of Year [11]

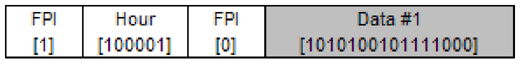
(b) VMF K05.15 Data #1 1 limits GPI to be '1' (D)

Figure 8. Condition Rule

5) *Field End Pattern*

This mechanism signifies the end of a field before it reaches the pre-defined length and is called a *Field End Pattern* (FEP). By using FEP, message length can be shortened.

Fig. 9 shows an example of FEP used for a VMF K05.1 message. Data #1 field can be as long as 1050 bits. However, if the Data #1 should be 350 bits long, a FEP of '1111000' is added at the end of the 350 bits.



FPI [1]	Hour [100001]	FPI [0]	Data #1 [1010100101111000]

Figure 9. Field End Pattern in the VMF K05.15 message

A) Case Definition Rule (CDR)

A CDR is defined with CDR fields Cdrs and the subordinate data fields $\mathrm{CdrsConsq}_i$, which are not necessarily sequentially placed.

Usage Syntax: Cdrs fields + …+ CL+ … + CL -- CL is CdrsConsq of Cond.

$\rightarrow$ *If* Cdrs == case_1 then *val*($\mathrm{CdrsConsq}_1$) = x; *val*($\mathrm{CdrsConsq}_2$) = y; …

If Cdrs == case_2 then *val*($\mathrm{CdrsConsq}_1$) = z; *val*($\mathrm{CdrsConsq}_2$) = a; …

Else if …

B) Data Compression Rule (DCR)

Data Compression Indicator (DCI) is used to represent *multiple fields with fixed-value* to reduce the length of a field.

Usage Syntax: DCI field + uncompressed fields

$\rightarrow$ *If* a DCI field is applied to the fields B and C in the following message

Field A(6 bits) + Field B(8 bits) + Field C(3 bits) + Field D(4 bits)

then *Field A(6 bits) + DCI (Field B&C)(2bits) + Field D(4 bits)* = 12

Figure 10. Specification mechanisms for context dependent message length variability

3.2 Context Dependent Mechanisms for Specifying Message Length Variability

There are two commonly used context dependent rules: the *Case Definition Rule* (CDR) and *Data Compression Rule* (DCR). These two mechanisms are formally described in Fig. 10.

1) *Case Definition Rule*

CDR is a rule specified with one or more FIs (or CIs), which together put certain constraint among subsequent subfields, FIs or CIs that depend on the factors external to the message. The subsequent subfields, FIs or CIs need not be in sequence.

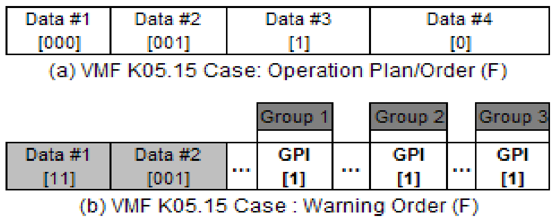


Figure 11. Case Definition Rule

Fig. 11 shows an example of CDR used for a VMF K05.15 message. In Fig. 11(a), multiple fields such as Data #1 and Data #2 should have the fixed values as defined by CDR. In Fig. 11(b), likewise, Data #1 should be '11' and at the same time Data #2 should be '001', and GPI 1, GPI 4 and GPI 7 should be '1'. In this case, GPI fields should occur with the predefined values as dictated by Data #2.

2) *Data Compression Rule*

Data Compression Indicator (DCI) uses a symbol field to replace multiple fields, which are collectively called the *profile*. In Fig. 12, it is possible to apply a profile to repeated IPv4 fields which have a fixed value during transmission. The entire length of the message will be reduced, but the contents will be saved by a profile.

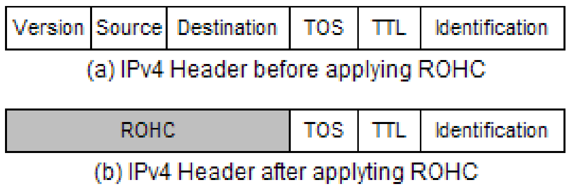

Version
Source
Destination
TOS
TTL
Identification
(a) IPv4 Header before applying ROHC
ROHC
TOS
TTL
Identification
(b) IPv4 Header after applying ROHC

Figure 12. IPv4 header compression with DCI

4. Deriving Test Requirement Patterns

In this section, we derive test requirement patterns based on the taxonomy developed in Section 3. For that two principles are used. The first principle is that test requirement patterns should include cases for both valid messages and invalid messages (*Principle P1*). The second principle is that the boundary values of a field, i.e. the maximum and the minimum values of a field, are the major aspect of testing in the case of a field with a range of values (*Principle P2*). Because we derive test requirement patterns based on the taxonomy presented in Section 3, the organization of Section 4 matches that of Section 3.

4.1 Test Requirement Patterns for Context Independent Message Length Variability

Test requirement patterns for the sub-cases of FI are shown in Table I. Depending on whether FI indicates presence or length or type, there are three cases. By applying Principle P1 the three cases are partitioned into validity test requirement patterns ((1), (2) and (3)) and invalidity test requirement patterns ((4), (5) and (6)). Since FI indicates ranges in the cases of (2) and (5), by Principle P2, (2) and (5) each generate two test requirement patterns, one for the maximum value and the other for the minimum value. The notation 'V' or 'IV' indicates the type requirements; valid or invalid case.

Table I. Test requirement patterns for FI

(1)	FEI _V1	*val*(FEI) = 1; A subsequent subfield should exist.
	FEI _V2	*val*(FEI) = 0; Subsequent subfield should not exist.
(2)	FLI _V1	*val*(FLI) ≥ 1; A subsequent subfield should exist and its length should be *val*(FLI).
	FLI _V2	*val*(FLI) = 0; Subsequent subfield should not exist.
(3)	FTI _V1	*val*(FTI) > 1; A subsequent subfield should exist and its type should be *val*(FTI).
	FTI _V2	*val*(FTI) =0; Subsequent subfield should not exist.
(4)	FEI _IV1	*val*(FEI) =1; Subsequent subfield does not exist.
	FEI _IV2	*val*(FEI) = 0; A subsequent subfield exists.
(5)	FLI _IV1	*val*(FLI) ≥ 1; A subsequent subfield exists but its length is not *val*(FLI).
	FLI _IV2	*val*(FLI) ≥ 1; Subsequent subfield does not exist.
	FLI _IV3	*val*(FLI) = 0; A subsequent subfield exists.
(6)	FTI _IV1	*val*(FTI) = 0; A subsequent subfield exists.
	FTI _IV2	*val*(FTI) ≥ 1; Subsequent subfield does not exist.
	FTI _IV3	*val*(FTI) = m; A subsequent subfield exists but its type is not *val*(m).

In contrast with the case of FI, there are one or more subsequent subfields to match CI. As CI is related to FI, a hierarchical message structure is formed. Thus test requirement patterns for CI should address the hierarchical structure of the form 'CI – FI – Subsequent Subfield(s)' together with the matching structure resulting from CI. Appendix A contains test requirement patterns for CI. Testing related to RI can be performed for the cases where its value is 1 and 0 for the Field and Cluster. So there are two test requirement patterns for FRI and CRI, respectively. Then test requirement patterns are partitioned to validity and invalidity test requirement patterns. FRI and CRI are flags indicating repetition of a field and a cluster, respectively. Both validity and invalidity test requirement patterns for them, therefore, focus on repetition of the dependent fields or clusters. In the case of FRI, which consists of one subsequent subfield per one flag, the number of cases is small. But in the case of CRI, which represents a group of fields, the number of cases is relatively high. Appendix B contains test requirement patterns for FRI and CRI.

In the case of test requirement patterns for CRs, the values of one or more fields in a message constitute the conditions and constraints related to the fields. Appendix C contains test requirement patterns for CR.

For the testing of FEP, in addition to the usual valid case, there are three invalid cases: 1) parsing a field terminated before it reaches the end of the field's predefined length without the occurrence of FEP, 2) there is an additional data after FEP and the additional data is interpreted, and 3) FEP occurs at the beginning of a field. Appendix D contains test requirement patterns for FEP.

4.2 Test requirement patterns for Context Dependent Message Length Variability

CDR describes data with patterns as prescribed by the context external to the message. For testing related to CR, it is necessary to verify that fields have pre-determined values based on the patterns. Appendix E contains test requirement patterns for CDR.

DCI is a symbol field for replacing one or more fields. By applying DCI to a block of fields, it is replaced by a reference that is previously defined in a table called a *Profile*. An entry of a Profile is a pair consisting of the data of a block of fields and a reference for the data. Appendix F contains test requirement patterns for DCI.

As can be seen in the case study, test requirements (and therefore test cases) with context dependent message length variability especially could be easily missed without the systematically prepared test requirement patterns.

5. A Case Study

In this Section, we conduct a case study of our approach with the testing of the parsers of the VMF messages [2]. Due to the flexibility provided by the message length variability and the precise definition mechanisms provided through its semantic message rules, VMF is considered essential for fast exchange of mission critical data in the tactical data systems [14]. Therefore it is crucial to check the correctness of the VMF parsers with respect to message length variability to ensure their proper operation.

As part of the project developing the system consisting of the VMF parsers and its associated testing tools, called *the AIV system*, we developed its testing tool component [15]. Fig. 13 shows an example of using the AIV system for the testing of VMF parsers.

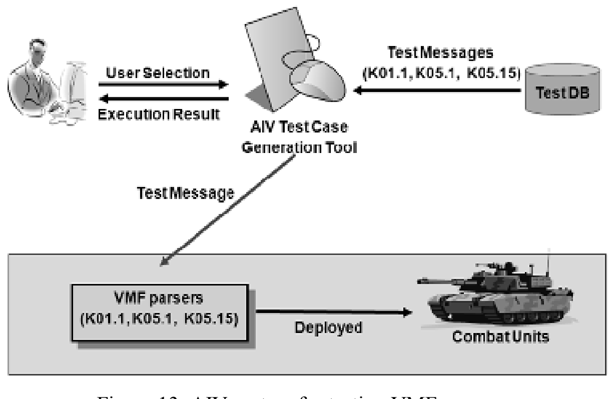

User Selection
Execution Result
AIV Test Case Generation Tool
Test Messages (K01.1, K05.1, K05.15)
Test DB
Test Message
VMF parsers (K01.1, K05.1, K05.15)
Deployed
Combat Units

Figure 13. AIV system for testing VMF parsers

For deriving test requirements and test cases for VMF messages, we first analyzed the syntax part of the VMF specification [2] and derived the test requirements and test cases from the syntax based on the Principle 1 and Principle 2 that were stated at the beginning of Section 4. So a systematic treatment of test cases derivation is conducted using Principles 1 and 2 but no systematic treatment is conducted on context dependent message rules or message length variability. We call this approach *the conventional approach* as it is one that would be followed without taking the kind of approach suggested in this paper. Using the conventional approach, the test requirements and test cases for VMF K01.1 using the conventional approach are derived as Tables II and III.

Table II. Test requirements for VMF K01.1 obtained using the conventional approach

No	Test Requirements
1	The subsequent subfield should be shorter than the field length limit.
2	FRI and subsequent subfield should exist together.
3	*val*(FRI) = 1; A subsequent subfield should exist, FRI and its subsequent subfield should repeat.
4	A pre-defined termination pattern (FEP) is present; Field should end early.

Table III. Test cases for VMF K01.1 obtained using the conventional approach

No	Test Cases
1	*Length limit*(Subsequent subfield) <= 1400; Insert to *val*(subsequent subfield) with text of 1000 characters 'xxxxxx....'.
2	*val*(FRI) = 0, *val*(subsequent subfield) = 'test'
3	*val*(FRI) = 1; *val*(Subsequent subfield) = 'test_test', *val*(Next FRI) = 0 , and *val*(Next subsequent subfield) = 'test_test'.
4	*val*(FRI) = 1, *invalid case*; *val*(Subsequent subfield) = 'test_test'
5	*val*(FEP) = '11100011'; *val*(Subsequent subfield) = 'test_test_11100011'.

For our approach to testing VMF messages, the procedure in Fig. 1 was followed to derive test requirements and test cases.

In Step E1, the test requirements for VMF messages K01.1, K05.1, and K05.15 were derived based on our test requirement patterns. From each type of VMF message we considered, i.e. K01.1, K05.1 and K05.15, elements with length variability are first identified, then applicable test requirement patterns are selected and finally test requirements for their parsers were derived from them. With our approach, for example, the test requirements and test cases for VMF K01.1 are derived as Table 4. The type column was added to the test requirements table. "N/A" in the type column indicates that a requirement is not part of message length variability requirement. Test requirements 5-9 are the ones that were not derived by the conventional approach and it is indicated by italicizing them.

Table IV. Test requirements for VMF K01.1 obtained using our approach

No	Type	Test Requirements
1	N/A	Same as Test Requirement 1 in Table II
2	N/A	Same as Test Requirement 2 in Table II
3	FRI_V1	Same as Test Requirement 3 in Table II
4	FEP_V1	Same as Test Requirement 4 in Table II
5	*FRI_IV1*	*val(FRI) = 1; Subsequent subfield do not repeat.*
6	*FRI_V2*	*val(FRI) = 0; The subsequent subfield exist, the FRI and the subsequent subfield should not repeat.*
7	*FRI_IV2*	*val(FRI) = 0; After a subsequent subfield appears, the fields with the same structure repeat.*
8	*FEP_IV1*	*FEP exists in the field without data.*
9	*FEP_IV2*	*FEP occurs after data that is shorter than the pre-defined length and data appear after FEP.*

Table V. Test cases for VMF K01.1 obtained using our approach

No	Type	Test Cases
1	N/A	N/A (Same as Test Case 1 in Table III.)
2	N/A	N/A (Same as Test Case 2 in Table III.)
3	FRI_V1	N/A (Same as Test Case 3 in Table III.)
4	FRI_IV1	N/A (Same as Test Case 4 in Table III.)
5	FEP	N/A (Same as Test Case 5 in Table III.)
6	*FRI_V2*	*val(FRI) =0;* *val(FRI) = 0, val(subsequent subfield) = 'tt'*
7	*FRI_IV2*	*val(FRI) = 0;* *val(Subsquent subfield) = 'tt_tt', val(Next FRI) = 0 , and val(Next subsequent subfield) = 'tt_tt'.*
8	*FEP_IV1*	*val(FEP) = '11100011';* *val(Subsequent subfield) = '11100011'.*
9	*FEP_IV2*	*val(FEP) = '11100011';* *val(Subsequent subfield) = 'tt_11100011_tt'.*

In Step E2, test cases of each message are generated based on the derived test requirements. Number of test cases is the number of test messages obtained by applying the relevant test requirements to the messages to be tested. The test cases for VMF K01.1 derived from the test requirements in Table IV are in Table V. Test cases 6-9 were not derived by the conventional approach.

The results of comparing the conventional approach and our approach are shown in Fig. 14, from which we can see that the numbers of test requirements for each message have almost doubled with our approach. This shows that our taxonomy and systematic test generation based on it significantly enhances test coverage.

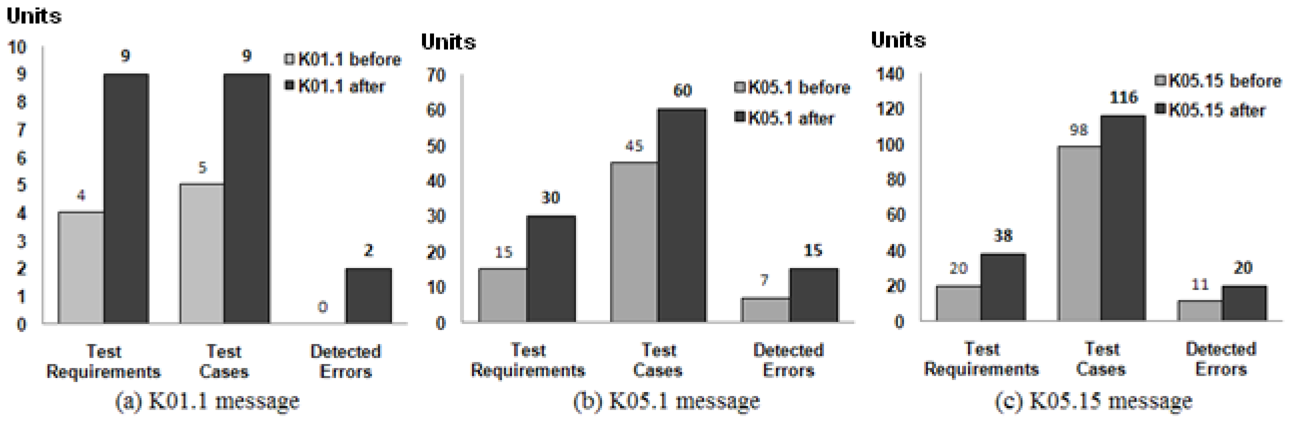


Figure 14. Comparison of the conventional approach and our approach

Our approach detected many errors related to message length variability in VMF parsers that were not identified by test cases derived using the conventional intuition-based approach. For test cases in Fig. p14, in our approach increments at Step E2 are relatively small compared to the increments at Step E1. That is, there are more additionally derived invalid test cases than additionally derived valid test cases. The reason is that, without a systematic guide as provided by our taxonomy and its consequent test requirement patterns, it is very hard to thoroughly consider all invalidity testing cases.

In conclusion, our approach showed a significant improvement over the conventional approach with respect to the coverage of test requirements and as its consequence in the numbers of generated test cases and detected errors.

6. Conclusion

This paper contains the following four contributions for message length variability testing. First, a systematic approach to message length variability testing as described in Fig. 1 was proposed. Second, as the first step of the approach, taxonomy of message length variabilities was developed. Third, based on the approach and the taxonomy, test requirement patterns that become the basis of actual test case derivation for specific protocols were derived. Fourth, through a case study, the efficacy of the approach has been demonstrated by comparing it with the conventional approach. The case study showed that the proposed taxonomy and the test requirement patterns are effective in deriving test cases for actual communication protocols and our approach is a big improvement over the conventional approach.

We developed our taxonomy to cover all message length variabilities known to us. But it is possible, although likely to be very rare, that a new pattern shows up that has not been anticipated in this paper. However, the method of taxonomy and the process to derive test requirement patterns and subsequent test case derivation proposed in this paper remains applicable and will accommodate a new type of message length variability in the same framework.

References

[1] The Internet Society, *RFC2460: Internet Protocol, Version 6 (IPv6) Specification*, December 1998.

[2] Department of Defense, *MIL-STD-6017, Variable Message Format*, 2004.

[3] Rainer Handel, Manfred N. Huber, Stefan Schroder, *ATM Networks: Concepts, Protocols, Applications*, Addison-Wesley Longman Ltd., Essex, UK, 1998.

[4] Zhang, L., Deering, S., Estrin, D, Shenker, S., and Zappala, D., “RSVP: A New Resource ReSerVation Protocol,” IEEE Network, pp 8-18, Sep. 1993.

[5] The Internet Society, RFC: 3095: *Robust Header Compression(ROHC): Framework and Four Profiles*, 2001.

[6] R. Lai, “A survey of communication protocol testing,” Journal of Systems and Software, Vol. 62, Issue 1, Pages 21-46, 1 May 2002.

[7] Kang, S., Seo, Y., Kang, D., Hong, M., Yang, J., Koh, I., Shin, J., Yoo, S., and Kim, M., "Development and Application of ATM Protocol Conformance Test System." In Proceedings of the IFIP TC6 12th International Workshop on Testing Communicating Systems: Method and Applications, September 01 - 03, 1999.

[8] Lee, David and Yannakakis, Mihalis, “Principles and methods of testing finite state machines - A survey,” Proc. of IEEE, 1996.

[9] S. Milliner, A. Delis, “Networking abstractions and protocols under variable length messages,” Third Int'l Conference on Network Protocols (ICNP'95), 1995.

[10] Y. T. Wu, J. F. Chang, "Collision resolution for variable-length messages", IEEE Transactions on Communications, Vol. 41 Issue 9, pp. 1281 - 1283, 1993.

[11] Redondo Systems Inc., http://www.redondosystems.com/vtt.html.

[12] C. Alexander, *The Timeless Way of Building*, Oxford University Press, 1979.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*, Wiley, 1996.

[14] D. Dusseau, C. Brock, "Network centric interoperability - using a variable message format (VMF) based data-link to improve situational awareness and close air support (CAS)", Proceedings of the 22nd Digital Avionics Systems Conference, pp.9A3.1 to 9A3.7, 2003.

[15] Jin-Gyu Kim, Sungwon Kang, Nohbok Lee, "Reliability Assessment of Translated VMF/Link-16 Messages," Proc. of the 19th IEEE Int'l Symposium on Software Reliability Engineering (ISSRE '08), Nov 11-14, 2008.

APPENDIX. TEST REQUIREMENT PATTERNS

A. Test requirement patterns for CI

CEI_V1	*val*(CEI) = 1; The number of subsequent subfields should be the same as the predetermined number.
CEI_V2	*val*(CEI) = 0; Subsequent subfields should not exist.
CLI_V1	*val*(CLI) ≥ 1; Subsequent subfields should exist and their total length should be *val*(CLI)
CLI_V2	*val*(CLI) = 0; Subsequent subfields should not exist.
CTI_V1	*val*(CTI) = m; Subsequent subfields of type m should exist.
CTI_V2	*val*(CTI) = 0; Subsequent subfields should not exist.
CEI_IV1	*val*(CEI) = 1; Subsequent subfields do not exist.
CEI_IV2	*val*(CEI) = 1; The number of subsequent subfields is not the same as the predefined number.
CEI_IV3	*val*(CEI) = 0; Subsequent subfields exist.
CLI_IV1	*val*(CLI) = 0; Subsequent subfields exist.
CLI_IV2	*val*(CLI) ≥ 1; Subsequent subfields do not exist.
CLI_IV3	*val*(CLI) ≥ 1; Subsequent subfields exist and its length is not *val*(CLI).
CTI_IV1	*val*(CTI) = m; Subsequent subfields do not exist.
CTI_IV2	*val*(CTI) = m; The type of subsequent subfields is not m.
CTI_IV3	*val*(CTI) = 0; Subsequent subfields exist.

B. Test requirement patterns for RI

FRI_V1	*val*(FRI) = 1; The subsequent subfield exist, the FRI and the field with the same structure should repeat.
FRI_V2	*val*(FRI) = 0; The subsequent subfield exist, the FRI and the subsequent subfield should not repeat.
CRI_V1	*val*(CRI) = 1; The subsequent subfields exist, the fields with the same structure should repeat.
CRI_V2	*val*(CRI) = 0; The subsequent subfields exist, but the fields with the same structure should not repeat.
CRI_V3	*val*(CRI) = 1; The FI(s) and subsequent subfields exist, the fields with the same structure should repeat.
CRI_V4	*val*(CRI) = 0; The FI(s) and subsequent subfields exist, but the fields with the same structure should not repeat.
FRI_IV1	*val*(FRI) = 1; Subsequent subfield do not repeat.
FRI_IV2	*val*(FRI) = 0; After a subsequent subfield appears, the fields with the same structure repeat.
CRI_IV1	*val*(CRI) = 1; Subsequent subfields do not repeat.
CRI_IV2	*val*(CRI) = 1; After subsequent subfields appear, some of the repeated fields are dropped.
CRI_IV3	*val*(CRI) = 0; After subsequent subfields appear, the fields with the same structure repeat.
CRI_IV4	*val*(CRI) = 1; FI(s) and subsequent subfields do not repeat.
CRI_IV5	*val*(CRI) = 1; After FI(s) and subsequent subfields appear, some of the repeated fields are dropped.
CRI_IV6	*val*(CRI) = 0; After FI(s) and subsequent subfields appear, fields with the same structure repeat.

C. Test requirement patterns for CR

CR_V1	Condition indicates type t; The type of the subsequent subfield(s) should be t.
CR_IV1	Condition does not indicate a pre-defined type.
CR_IV2	Condition indicates type t but the type of the subsequent subfield(s) is not t.

D. Test requirement patterns for FEP

FEP_V1	FEP should occur after data that is shorter than the pre-defined length.
FEP_IV1	FEP exists in the field without data.
FEP_IV2	FEP occurs after data that is shorter than the pre-defined length and data appear after FEP.

E. Test requirement patterns for CDR

CDR_V1	Case rule indicates type m; The subsequent subfields should have type m.
CDR_IV1	Case rule does not indicate type m but the subsequent subfields have type m.

F. Test requirement patterns for DCI

DCI_V1	DCI is used; Message should contain a reference in Profile.
DCI_IV1	DCI is not used; Message contains a reference in Profile.
DCI_IV2	DCI is used; No reference of Profile exists.
DCI_IV3	DCI is used; Message contains a reference in Profile but *length*(message) ≠ *length*(uncompressed message).