Verification of Transport Protocol's Parallel Routing of a Vehicle Gateway System

Hassan Mohammad and Piyush Patil
Body & EE
MBtech Group GmbH & Co. KGaA
Sindelfingen, Germany
Hassan.mohammad@mbtech-group.com
Piyush.patil@mbtech-group.com

Abstract—Transport protocol (TP) routing is a routing functionality of a vehicle gateway system enabling TP data to be transferred to different types of connected networks. This routing functionality is required for intra- as well as inter-vehicular communications such as flashing new software onto Electric Control Units (ECUs) and collecting status information (large data packets) and routing them for diagnostic purposes. Transport protocol's parallel routing is the scenario of establishing multiple TP routing instances in parallel in order to save time and resources. This article addresses the issue of verifying parallel routing of TP for a gateway system. To achieve this goal, a new recursive test case selection and generation strategy along with a suitable input parameter model is introduced. The new strategy is a combination test strategy guided by the category partition method to overcome the combinatorial explosion problem raised by testing of TP parallel routing functionality. Additionally, the new strategy enables user to analyze the performance of the gateway related to TP parallel routing. This is achieved by providing statistical information in diverse scenarios and determining the maximum number of guaranteed TP parallel routing instances. The statistical information can help in optimizing the configuration or the implementation of the transport protocol by providing hints about error causes.

Keywords- Parallel Routing of Transport Protocols, Combination Test Strategies, Input Parameter Models (IPM).

I. INTRODUCTION.

Today's vehicles Electric/Electronic (E/E) system is designed as a distributed system in order to overcome the increasing complexity and meet the diversity of requirements such as performance, comfort, safety and costs. In a vehicle distributed system, gateways are indispensable. They enable Electric Control Units (ECUs) within connected networks to interchange information necessary for accomplishing specified functionalities. A modern E/E system has multiple gateways, e.g., central gateway, telematic gateway, etc. During information interchange, the gateway routes data between its connected networks although they work on different communication protocols.

Mainly, two types of data routing can be established over the gateway. The first type is frame routing and concerns with routing of data that fit into one frame. This kind of routing is simple and out the focus of the article. The second type is TP routing and concerns with routing of data packets which do not fit into one frame. This routing functionality is required for intra-vehicular communications such as flashing new software onto Electric Control Units (ECUs), variant coding, and software update or even reading vehicle status. For such use cases, an external device "Tester" is connected via an external interface "OBD-connector" to the central gateway in order to access and communicate with the ECUs in the network. In inter-vehicular communication, TP routing is also required when large data packets need to be exchanged between the ECUs. For this type of routing, the gateway utilizes transport protocols, such as CAN transport protocol [1] and FlexRay transport protocol [2], which provide features for segmentation, reassembling, flow control and error detection. TP parallel routing is the scenario of routing TP data between multiple communicating ECUs located on different networks in parallel in order to save time and resources, as for example the case of flashing multiple ECUs in parallel.

Verifying TP parallel routing of a gateway system is not a trivial problem, since a huge number of possible combinations of communicating ECUs can be built when test cases have to be selected, especially if the combination of communicating ECUs demands transport protocol change, i.e., converting one transport protocol to another, when different types of protocols are involved during the routing process.

This article presents a new recursive test case selection and generation strategy along with a suitable input parameter model (IPM) to overcome the combinatorial explosion problem raised by testing TP parallel routing on a vehicle gateway system while still having a very good thoroughness of the test.

The remainder of this article is organized as follows. Section II describes the combinatorial explosion problem of TP parallel routing test, its differentiation from described problem in literature and techniques existed to fight it. The proposed approach is presented in details in Section III. Section IV discusses the approach and Section V concludes the article.

 ISBN: 978-1-61208-348-3

II. BACKGROUND AND RELATED WORK

A. The Combinatorial Explosion Problem of TP Parallel Routing Test

The gateway is part of a distributed network system. It communicates with its environment over communication channels via buses. Communication channels of a gateway system, e.g., CAN or FlexRay, are mostly heterogeneous, i.e., each has its own characteristics and behavior. A group of up to c channels shall be defined as the Communication Channels of a gateway system (1).

$$Communication\ Channels = (Ch_1, Ch_2, ..., Ch_c) \quad (1)$$

The environment consists of multiple ECUs, e.g., engine Control Module, interacting with the gateway and among each other. Each ECU is located on a specific channel and utilizes it to establish the required communication. If an ECU connected on a channel of the gateway needs to communicate with another ECU located on another channel, then the gateway establishes a routing process between the two ECUs. It receives the data from sending ECU on the source channel and routes it to the receiving ECU on the destination channel. The environment of the gateway shall be described as a group of e ECUs connected to Communication Channels (2).

$$Environment = (ECU_1, ECU_2, ..., ECU_e) \quad (2)$$

$$e >= c \quad (3)$$

Generally, ECUs in the environment exchange data over the gateway in predefined Fashions. Each Fashion is characterized through a set of configuration parameters which are required to establish TP routing relationships between communicating ECUs. A Fashion F shall be described as a set of configuration parameters P (4). (see [1] for configuration parameter of CAN TP).

$$Fashion_F = (P_{F_1}, P_{F_2}, ..., P_{F_p}) \quad (4)$$

As described before, ECUs in the environment communicate over the gateway in diverse Fashions. Fashions are a superset of Scenarios, where one Scenario is constructed with the set of parameters for a Fashion. Scenarios are related to ECUs, i.e., some ECUs could communicate only in some Scenarios. All possible Scenarios shall be described as a group of s Scenarios (5).

$$Scenario = (Scenario_1, Scenario_2, ..., Scenario_s) \quad (5)$$

A Connection Channel is an instance of a Fashion and it has the same set of parameters defined for that Fashion. A Connection Channel in the Fashion F shall be described (6).

$$Connection\_Channel_x = (P_{F_{1x}}, P_{F_{2x}}, ..., P_{F_{px}}) \quad (6)$$

A Routing Instance is a relationship between a specific Connection Channel and a possible Scenario and shall be described (7).

$$Routing\_Instance_x = (P_{F_{1x}}, P_{F_{2x}}, ..., P_{F_{px}}, Scenario_x) \quad (7)$$

The gateway can be configured to serve y Routing Instances in parallel. The number y of parallel Routing Instances is a configuration parameter which needs to be verified. In the case of errors, the next verified y should be determined. To verify parallel routing of y instances, the combinatorial explosion problem arise. By considering all variables described before and assuming that all ECUs communicate with the same number of Scenarios; the theoretical number of combinations for routing instances can be calculated (8).

$$X = \left(\frac{e!}{y!(e-y)!}\right).y^s + \left(\frac{e!}{(y-1)!(e-(y-1))!}\right).(y-1)^s + ... + \left(\frac{e!}{1!(e-1)!}\right).1 \quad (8)$$

Let us take a simple example and assume that the E/E system has 50 ECUs, the gateway is configured to support only 3 Routing Instances in parallel and each Connection Channel can be mapped to only 3 Scenarios. The number of theoretical combinations in this example is 5390050. Assume that to test each possible combination 10 seconds are needed, testing the theoretical number of combinations will require 62.39 days. This duration is not acceptable.

B. Combination Testing Strategies

To overcome the combinatorial explosion problem of testing distributed systems that consist of a number of interacting elements, combination testing strategies have been devised in literature [3]-[11]. A chronological overview with a comparison of diverse strategies can be found in [12]. Combination testing strategies are category partition [13] based methods that supports the finding of a trade off between test coverage and available resources by providing techniques for selecting combinations of parameter.

The combinatorial explosion problem mentioned in literature shall be explained as in the following example:

Assume a distributed system consisting of a central unit interacting over communication channels with n units of the environment $u_1, u_2, \ldots, u_n$. Each unit u_i in the environment uses a defined parameter p_i for communication. The parameter p_i shall have v_i possible configuration values. By assuming that configuration values of parameters are independent from each other, the number of configuration possibilities of the system would be $v_1 \times v_2 \times \ldots \times v_n$. If each possible configuration requires c test cases to verify it, the number of test cases for exhaustive test would be $c \times v_1 \times v_2 \times \ldots \times v_n$. In a nontrivial software system, the values

 ISBN: 978-1-61208-348-3

of n and v_i are large which leads to a huge number of possible combinations of parameter values.

In order to find a trade off between test thoroughness and test resources, combination test strategies define coverage criteria needed to be satisfied. Coverage criteria can be varied between *1-wise* to *N-wise*.

1-wise coverage criterion requires that, each interesting value of each parameter must be included at least in a test case to reach 100% coverage. Whereas, *N-wise* coverage or exhaustive test requires that all possible combinations of interesting values must be included in generated test cases. Decision on which coverage criterion to be used depends on several factors, such as the effort required to construct each test case, time, resources and budget. Studies on reported bugs and failures [14][15] have shown that *2-wise* or *pair-wise* combinations are very effective in finding failures of parameter interaction. However, as shown in [16], the quality of *2-wise* combination testing is affected strongly by diverse factors which can be only partially influenced by the technique.

Related to testing TP parallel routing, the goal of test is to measure the performance of the gateway to handle multiple parallel Routing Instances. The problem is more complex because:

- n is not a constant. It is a configuration parameter which can be different for every new release of the system.
- Each unit of the environment can have multiple sets of configuration parameters that can be used in defined scenarios to establish a communication.
- Sets of configuration parameters also include timing parameters and the interactions between different values of timing parameters are difficult to resolve.
- The number y of parallel instances, which is also a configuration parameter, can be any subset of n. In case of errors, one of the test goals is to find the next verified y.
- In TP parallel routing, each additional instance will consume resources of the system and may lead to errors. Hence, it is not only a specific combination of input parameter values which can affect the behavior and may reveal errors, but also the number of included input parameter sets and their values.

Based on these factors and other uncontrolled factors mentioned in [16], the proposed combination testing strategies revealed in literature are not suitable for fighting the combinatorial explosion problem in our case.

C. Related Work

During the literature research, no combination testing technique was found that is designed to support testing parallelism of applications. Only systems accepting a fixed number of input parameters and techniques to solve the problem of handling combinations of interesting values for those parameters have been discussed in literature.

To solve the combinatorial explosion problem raised by testing TP parallel routing, a new recursive test case selection and generation approach is proposed. The approach is based on the category partition method and utilizes *N-wise* coverage criterion. A suitable IPM [17]-[20] is also defined and serves as an input for the test case generation strategy.

Recursively generation and running of combinatorial test cases gives the ability to analyze the results from executed test cases and collect symptoms. Information gained can help deciding the next parameter sets to combine.

Although the proposed approach is guided by the category partition method, it differs from it in diverse aspects. One aspect is that the proposed approach deals with testing of parallelism, which is described through combinations of instances. Another aspect is the definition of interesting parameter values in category partition, which is different in the proposed approach.

An important difference to existing combination test strategies is the usage of semantic information in a recursive approach. Semantic information is not part of existing combination test strategies in their basic form. It is utilized here to build an IPM and to guide the selection of combinations which can reduce the test suite size.

III. PROPOSED STRATEGY FOR TEST CASE SELECTION AND GENERATION

In this section, steps for the test case selection and generation methodology are discussed. The methodology shall select and generate test cases to test the gateway system for its user test requirements confined to TP parallel routing with an efficient number of test cases. The methodology is categorized in 5 steps as depicted in Fig. 1.

1. Determining existing scenarios and defining constraints.

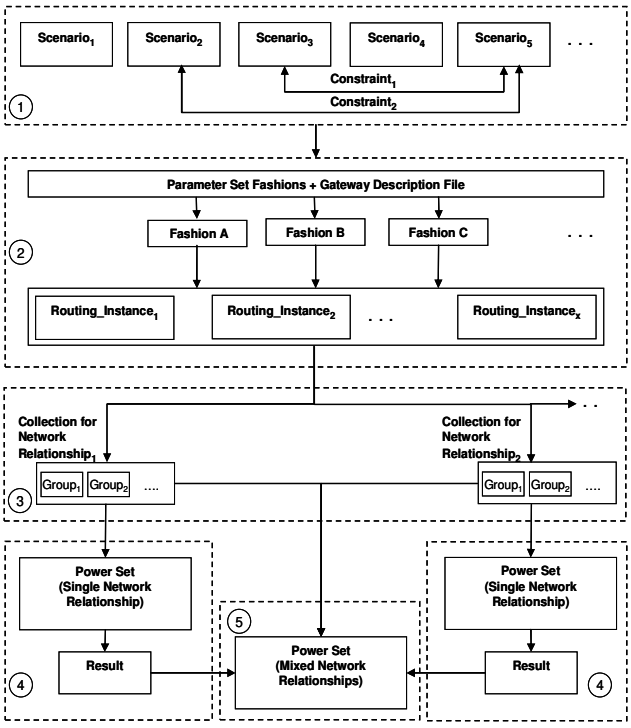

Figure 1. Test Case Selection and Generation Steps.

2. Mapping TP Scenarios onto Connection Channels in order to construct Routing Instances.
3. Collection and completion based on similarity criteria.
4. Testing TP parallel routing for Single Network Relationships (SNRs).
5. Testing TP parallel routing for Mixed Network Relationships (MNRs).

The first 3 steps are concerned with constructing an IPM and the following steps with the new combination testing strategy.

A SNR mentioned in step 3 is an abstract term, which describes all Routing Instances between two specific networks of the gateway.

A MNR in step 5 comprises Routing Instances from different SNRs.

A. Determining Existing Scenarios and Defining Constraints

In this step, TP routing scenarios are discussed and analyzed with persons practicing TP functionalities. At the end of this step, real use case scenarios are defined and described in a selectable format. Several advantages are gained from this step:

- Real use case scenarios are mostly not described and can not be recognized or built automatically from gateway description file.
- Analyzing can help in avoiding scenarios which are not practiced but theoretically conceivable.
- Future extensions for scenarios can be discussed and defined.

Along with determining existing scenarios, constraints can also be defined.

- Constraints for combinable or non-combinable Scenarios.
- Constraints for combinable or non-combinable Connection Channels.
- Configuration Constraints, e.g., maximum configured parallel routing instances.

The user has the ability to construct *preventing* or *allowing* Constraints. *Preventing* Constraints are responsible for preventing specific combinations to be constructed and included in a test case. Whereas, *allowing* constraints define conditions used to build certain combinations.

The decision on using *preventing* or *allowing* constraints depends on the case study. If the number of *allowing* constraints is bigger than the number of *preventing* constraints, then it is better to use *preventing* constraints for selection in order to reduce manual effort. Constraints are crucial for combination selection. They can reduce the number of combinations to a large extend. Defined constraints shall be described in a suitable format.

Examples for *preventing* constraints between scenarios:

$$Scenario_2 + Scenario_5 = NC \quad (9)$$

$$Scenario_3 + Scenario_5 = NC \quad (10)$$

NC: Not Combinable.

Examples for *preventing* constraints between Connection Channels:

$$Connection\_Channel_x + Connection\_Channel_z = NC \quad (11)$$

Example for configuration constraints:

$$Maximum\_Parallel\_Instances\_CAN\_TP = y \quad (12)$$

B. Mapping TP Scenarios onto Connection Channels in Order to Construct Routing Instances

The gateway is described on a certain abstraction level by means of a description file. ECUs communicating with the gateway have parameters defined in the description file. These parameters define the behavior of ECUs from the gateway point of view. If an ECU communicates using the transport protocols in a specific scenario, a related set of configuration parameters called *Connection_Channel* are utilized.

Mapping TP Scenarios onto Connection Channels is a step in which TP parameters of defined Connection Channels are extracted and then mapped to TP Scenarios. As a result of this step, each Connection Channel included in the gateway description file must be related to minimum one specific TP Scenario. Resulted relationships are called Routing Instances. Examples of mapping can be formulated as in the following (13) (14).

$$Routing\_Instance_A = (P_{F_{11}}, P_{F_{21}}, ..., P_{F_{p1}}, Scenario_x) \quad (13)$$

$$Routing\_Instance_B = (P_{F_{12}}, P_{F_{22}}, ..., P_{F_{p2}}, Scenario_x) \quad (14)$$

C. Collection and Completion Based on Similarity Criteria

The goal of collection is to cluster similar Routing Instances which stimulate the same or similar behavior in the gateway when TP routing is established. In the proposed approach, collection is performed in two steps as depicted in Fig. 2.

- Creating groups out of Routing Instances. Routing Instances of each created group must have the same values for all related parameters such as routing parameters, network relationships and mapped scenarios.
- Creating SNR collections out of constructed groups. Groups of a SNR collection must have the same network relationships, i.e., the same source and destination networks for all of their Routing Instances. SNR collections are the base for TP parallel routing test of single network relationships.

Collection process is part of designing the IPM and helps in reducing the number of combinations required for test.

 ISBN: 978-1-61208-348-3

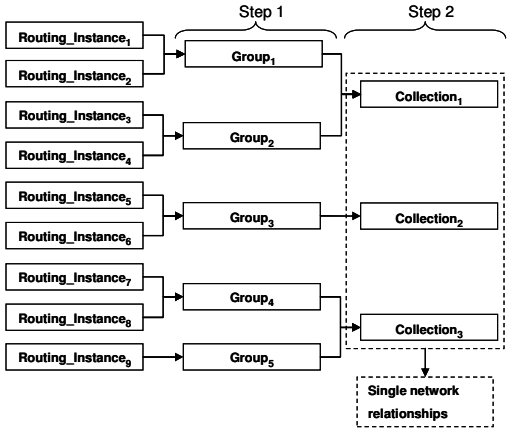

Figure 2. Steps of Collection

The following example explains reduction achieved when groups are constructed:

Assume that 4 Routing Instances A, B, C and D are constructed and the gateway is configured to support 2 Routing Instances in parallel. The number of possible combinations of 2 Routing Instances out of 4 would be 6 (the order has no effect). Groups can be constructed based on similarity criteria such that $Group_1$ consists of instance A and instance B, $Group_2$ consists of instances C and D. After grouping, the number of combinations could be rather reduced to 3, because all other possible combinations would resemble the same effects on the gateway, i.e., combinations of instances (A, C), (A, D), (B, C) and (B, D) are all similar and can be replaced by only one representative as in example (A, C). (see Fig. 3).

In completion, Similarity Numbers and Stress Factors are assigned to constructed groups. The same Similarity Number will be assigned to groups if their Routing Instances have the same routing parameters, the same scenarios and the same characteristics for network relationships. Concerned characteristics are the protocol type and channel bandwidth.

The Stress Factor is calculated initially based on aspects such as the expected processing time, memory usage, network bandwidth and other network specific aspects. Stress Factor shall be also adjusted during the test run based on variance. Table 1 represents an output example of the grouping and completion process.

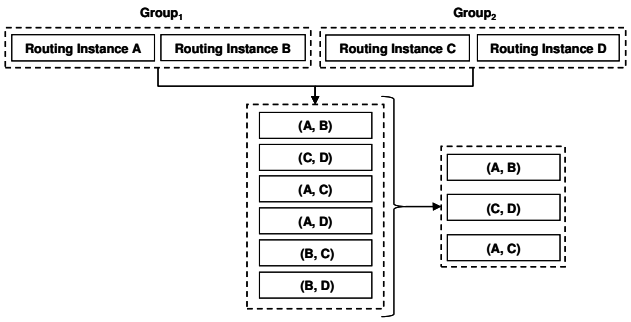

Figure 3. Advantages of Building Groups

Similarity Numbers and Stress Factors are required for the combination selection during TP parallel routing test for MNRs.

The number of formed groups in step 1 of collection depends on the complexity of the gateway under test with respect to connected networks, their heterogeneous level, the number of configured Connection Channels and the heterogeneous level of their parameters.

Grouping process has several benefits in addition to reducing the number of combinations required for testing:

- It assists in analyzing the parallel routing behavior of the gateway under diverse combinations of configuration parameters or combinations of network relationships.
- It can facilitate the search for error causes by enabling the user to compare routing results under particular scenarios, and gain information about the relationship between specific attributes of the included Routing Instances and raised errors.

D. Testing TP Parallel Routing for Single Network Relationships (SNRs)

The proposed test case selection and generation approach is a recursive technique consisting of two main test phases. The first phase deals with TP parallel routing test for SNRs by means of constructed SNR Collections. The second phase deals with TP parallel routing test for MNRs based on Similarity Numbers and Stress Factors. This separation is very practical for analyzing the behavior of the gateway and can provide information about possible reasons for errors if they can be revealed.

In the first phase, formed SNR Collections are picked up successively. For each SNR Collection, a power set of its groups shall be constructed. Power set is the set of all subsets of input elements without the empty set, and serves as a medium to check if the coverage criteria can be completely achieved. Subsets are then categorized into levels, where each level consists of all subsets with the same number of elements (see Fig. 4 as an example of a SNR Collection with 3 groups). Subsets on a specific level substitute implicitly all subsets on the successive levels. This feature shall be used to

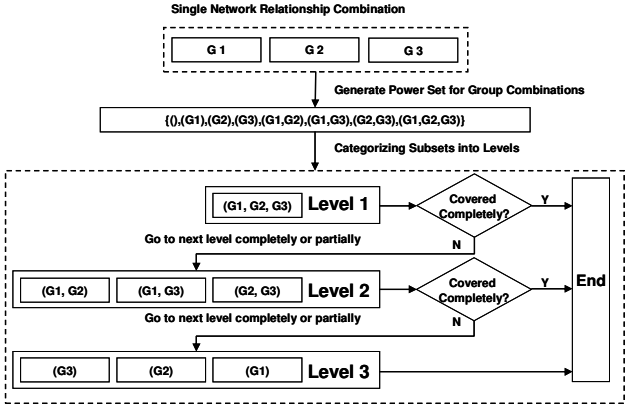

Figure 4. Checking the Combination Coverage with the Help of Power Set

TABLE I. OUTPUT EXAMPLE OF FIRST GROUPING STEP

	Parameter$_1$	**Parameter$_2$**	**Parameter$_3$**	**Parameter$_4$**	**Scenario**	**Similarity number**	**Stress Factor**
Group$_1$ (n_1 Routing Instances)	1,2	Any	Any	-	Scenario 1	1	3
Group$_2$ (n_2 Routing Instances)	1,2	Any	Any	-	Scenario 1	2	1
Group$_3$ (n_3 Routing Instances)	1,2	Any	Any	-	Scenario 1	3	2
Group$_4$ (n_4 Routing Instances)	1,2	Any	Any	-	Scenario 1	4	4
Group$_5$ (n_5 Routing Instances)	1,3	Any	Any	Any	Scenario 2	1	3

reduce the combinations in the successive levels when test cases for all combinations on a specific level can be built.

Levels shall be handled using a top to bottom processing strategy. Since combinations on one level comprise implicitly all combinations on successive levels, the capability of generating test cases for combinations on a specific level would be sufficient to finish the parallel routing test for the related SNR. Considering the example in Fig. 4, if the combination (G1, G2, G3) on level 1 can be built in a test case, the processed SNR test can be completed because all other combinations on levels 2 and 3 are included implicitly in the combination on level 1.

Generally, when test cases are generated, two possibilities can be distinguished for each combination of groups on a processed level. Either a test case can be generated for that combination completely or partially.

A completely generated test case describes the situation when all Routing Instances of all groups for a selected combination is included in one test case (constraints and gained information are criterion for the construction of test cases). A partially generated test case describes the situation when Routing Instances of groups for a selected combination can only be partially included. For such situations, the algorithm shall proceed to the next lower level to cover missing combinations.

Reasons for utilizing power set are:

1. In the formulation of subsets, the order of elements has no effect.
2. The count of elements in subsets varies between one to a maximum number.

These two features are correlated to parallel routing, because the order of Routing Instances in a test case has no effect on the test and the count of Routing Instances can be varied from one to a maximum configured number.

Introducing a recursive testing technique for parallel routing can help in analyzing the results from executed test cases. By analyzing the results, Stress Factors shall be corrected if a variance is observed. Additionally, groups that stress the gateway more than others shall be isolated for testing TP parallel routing for MNRs.

E. Testing TP Parallel Routing for Mixed Network Relationships (MNRs)

Parallel routing test for SNRs helps in correcting group's Stress Factors. From groups of each SNR Collection, a representative with the best Stress Factor shall be marked when combinations have to be selected for testing TP parallel routing for MNRs. Since networks can also have similarities among each other, the number of combinations can be further reduced by omitting similar combinations for MNRs. This can be achieved based on the Similarity Number of SNR collection's groups. In the selection of combinations for MNRs, the power set of available networks shall be constructed and similar subsets shall be deleted. Resulted combinations for MNRs shall be the base to check if the coverage criteria can be completely achieved.

The described concept for testing TP parallel routing for MNRs shall be explained in the example in Fig. 5:

Network 1 (N1) consists of formulated groups G1, G2 and G3 along with their respective Similarity Numbers (Si. N.) and Stress Factors (S.F.). Network 2 (N2) and Network 3 (N3) also contains similar information. Representative groups shall be selected based on the groups having best Stress Factors from N1, N2 and N3. Thereby, N1 (G1), N2 (G1) and N3 (G2) can be selected for optimizing the number of combinations. Other groups shall be omitted because of following reasons:

- Groups excluding representatives have higher Stress Factor and shall affect the behavior of gateway and hence they should be used for defining worst case scenarios.
- Representative groups implicitly resemble the excluded groups from each of the networks and hence can reduce the duplication of the process.

Power set shall be formulated from the representatives. Based on the Similarity Numbers, subsets from the power set shall be omitted, thereby resulting into an optimized formulated power set. This optimized power set shall be further used for categorizing into levels as explained in the previous section.

Testing TP parallel routing for MNRs then follows the same concept as of testing TP parallel routing for SNRs.

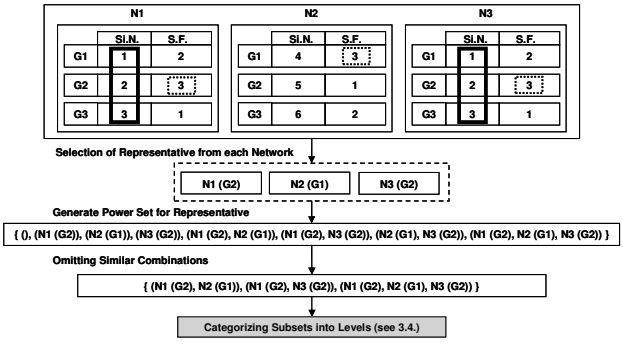

Figure 5. Optimizing Power Set for Mixed Network Relationships

Finally, the number of combinations required to test TP parallel routing for a given system shall be calculated from the number of combinations for testing SNRs along with the number of combinations for testing MNRs.

IV. DISCUSSION

Depending on the grade of diversity in parameters for Connection Channels and for the gateway connected networks, the number of resulted groups can increase. The idea is to use combinations of groups instead of combining Routing Instances to reduce the number of generated combinations. If the number of groups still higher, then Stress Factors shall be required within groups for testing SNRs. Another drawback of this approach is the need of system functionality expertise to define similarity criterion and calculate the Stress Factors of the groups. However, this needs to be performed only once. Later on, combinations to be tested and test cases can be generated automatically for each new release of the system.

V. CONCLUSION AND FUTURE WORK

In this paper, a recursive test case selection and generation methodology has been proposed to overcome the combinatorial explosion problem in testing TP parallel routing on a gateway. Based on similarities between parameter values of Connection Channels, the methodology collects Connection Channels into groups which serve as input for building combinations to verify TP parallel routing for SNRs. Similarities between networks together with Stress Factors gained from verifying SNRs provides the base for building combinations to testing TP parallel routing for MNRs. The two phases for testing TP parallel routing are very practical and can provide information for optimizing the TP configurations and error analysis. After group selection, power set is used to construct combinations of groups which is required to completely achieve the *N*-wise coverage criteria. Subsets of power set are divided into levels to give orientation for constructing combinations and contribute in reducing combinations for testing. An Implementation of the Methodology is currently under development to test TP parallel routing on a central gateway with five networks (3 CAN networks with 500 kilo baud, 1 CAN network with 250 kilo baud and 1 FlexRay network with 10 Mbps).

REFERENCES

[1] "Road vehicles-Diagnostics on Controller Area Network (CAN)-," ISO 15765:2004(E), Switzerland, 2004.

[2] "Road vehicles-Communication on FlexRay-," ISO 10681:2010(E), Switzerland, 2010.

[3] P. E. Ammann and A. J. Offutt, "Using Formal Methods to Derive Test Frames in Category Partition Testing," In Ninth Annual Conference on Computer Assurance (COMPASS'94), Gaithersburg, MD, Jun. 1994, pp. 69–80.

[4] D. M. Cohen, S. R. Dalal, A. Kajla, and G. C. Patton, "The Automatic Efficient Test Generator (AETG) System," Proc. of the 5th International Symposium on Software Reliability Engineering, Monterey, CA, Nov. 1994, pp. 303–309.

[5] M. B. Cohen, J. Snyder, and G. Rothermel, "Testing Across Configurations: Implications for Combinatorial Testing," Proc. of the 2nd Workshop on Advances in Model Based Software Testing, Raleigh, North Carolina, USA, Nov. 2006, pp. 1–9.

[6] C. J. Colbourn, M. B. Cohen, and R. C. Turban, "A Deterministic Density Algorithm for Pairwise Interaction Coverage," Proc. of the IASTED Intl. Conference on Software Engineering, Innsbruck, Austria, 2004, pp. 345–352.

[7] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A General Strategy for T-Way Software Testing," Proc. of the 14th Annual IEEE Intl. Conf. and Workshops on the Engineering of Computer-Based Systems, Tucson, AZ, Mar. 2007, pp. 549–556.

[8] Y. Lei and K. C. Tai, "In-Parameter-Order: A Test Generation Strategy for Pairwise Testing," Proc. of the 3rd IEEE Intl. High-Assurance Systems Engineering Symposium, Washington, DC, Nov. 1998, pp. 254–261.

[9] Y. K. Malaiya, "Antirandom Testing: Getting the Most Out of Black-Box Testing," Proc. Of the 6th International Symposium on Software Reliability Engineering, Toulouse, Oct. 1995, pp. 86–95.

[10] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing," Proc. of the 28th Annual Intl. Computer Software and Applications Conference (COMPSAC 2004), Hong Kong, Sept. 2004, pp. 72–77.

[11] K. C. Tai and Y. Lei, "A Test Generation Strategy for Pairwise Testing," IEEE Transactions on Software Engineering 28, Jan. 2002, pp. 109–111.

[12] M. Grindal, J. Offutt, and S. F. Andler, "Combination Testing Strategies: A survey," Software Testing, Verification, and Reliability, 2005, pp. 167–199.

[13] T. J. Ostrand and M. J. Balcer, "The Category-Partition Method for Specifying and Generating Functional Tests," Communications of the ACM, 31(6), New Yourk, Jun. 1988, pp. 676–686.

[14] D. R. Kuhn and M. J. Reilly, "An Investigation of the Applicability of Design of Experiments to Software Testing," Proc. of 27th NASA Goddard/IEEE Software Eng. Workshop, Dec. 2002, pp. 91–95.

[15] D. R. Wallace and D. R. Kuhn, "Failure Modes in Medical Device Software: An Analysis of 15 Years of Recall Data," Int. J. of Reliability, Quality and Safety Eng., vol. 8, no. 4, 2001, pp. 351–371.

[16] J. Bach and P. J. Shroeder, "Pairwise Testing: A Best Practice that Isn't," Proc. of the 22nd Pacific Northwest Software Quality Conference, 2004, pp. 180–196.

[17] M. N. Borazjany, L. S. G. Ghandehari, Y. Lei, R. N. Kacker, and D. R. Kuhn, "An Input Space Modeling Methodology for Combinatorial Testing," Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference, Luxembourg, Mar. 2013, pp. 372–381.

[18] M. Grindal and J. Offutt, "Input Parameter Modeling for Combination Strategies," Proc. SE'07 Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, Anaheim, CA, 2007, pp. 255-260.

[19] M. Grindal, J. Offutt, and J. Mellin, "Handling Constraints in the Input Space when Using Combination Strategies for Software Testing," Technical Report HS-IKI-TR-06-001, School of Humanities and Informatics, University of Skövde. 2006.

[20] S. A. Vilkomir, W. T. Swain, and J. H. Poore, "Software Input Space Modeling with Constraints among Parameters," Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International, Seattle, WA, Jul. 2009, pp. 136–141.

 ISBN: 978-1-61208-348-3