A Comprehensive Test Strategy for Network Protocols in Diverse Environment

Akihito Hiromori*, Hirozumi Yamaguchi* and Teruo Higashino*
* Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka, Japan 565-0871
{hiromori,h-yamagu,higashino}@ist.osaka-u.ac.jp

***Abstract*—In order to analyze important properties of network protocols, such as robustness and applicability, we may need exhaustive tests to observe effects of various factors under different settings. For recent protocols on dynamic, large-scale and environment-aware networks such as wireless sensor networks and mobile ad-hoc networks, we should consider many factors due to diversity of hardware profiles, upper/lower layer protocols and physical environments. In this paper, we propose a comprehensive test method to analyze the effect of (usually 10 or more) factors on such network systems. Our method takes a set of factors to be considered and their (representative) domain values as inputs, analyzes their effects on the systems, and determines dominant factors that have impact on the performance and their interactions. Instead of applying exhaustive tests that require all the combinations of domain values, we take a step-wise approach that examines step-by-step suspected sets of factors, which requires fewer combinations. We also justify this approach based on a reasonable fault model. The approach also contains an analytical method to identify the performance characteristics. Through realistic case studies, we show that we could find sets of dominant factors in wireless networks systematically.**

***Index Terms*—wireless network; performance evaluation; network simulation**

I. Introduction

Network protocols and applications originally involve some protocol parameters, system state variables, and many optional functions. Some of them take wide range values and some others have binary states. Especially, environments for wireless sensor networks and mobile ad-hoc networks are very diverse. They may be operated with different node mobility (speed and movement patterns, if some of nodes are mobile) and node density. There may also be numerous patterns for sensor node deployment according to the targeted geography. Furthermore, sensor nodes may have different battery lifetime and wireless ranges, and their hardware profiles may be heterogeneous. This means that, unlike the case for conventional wired network systems, we may need to consider more environmental factors for testing such networks [1], [2].

Hereafter, for simplicity of notations, parameters of which the tester would like to vary their values are called *factors* and their domain values are called *levels*. This terminology is taken from the design of experiments [3], which is discussed in Section II. Then each combination of levels of factors is called a *test case*. In popular performance evaluations, we set representative values (levels) for non-interested factors and vary the levels of interested factors. Example strategies to vary these levels include "best-guess" that changes the level of the most influent factor after each test and "one-factor-at-a-time" that changes the level of only one factor and sets the others to the pre-defined baseline levels at each test [3]. These approaches do not require so many test cases, but we cannot find out such factors that *together* reveal distinctive performance characteristics of the implementation under test. In other words, these factors miscellaneously interact with each other in affecting the performance and we call them *dominant factors*.

Identifying dominant factors and checking their effects on the system performance and behavior are very important for protocol designers, developers, testers and anyone who is engaged in design and development. There have been a few literature reporting that particular combinations of performance factors affect the entire performance of systems. For example, [4] investigates how ad-hoc routing protocols, MAC protocols, and mobility models affect network performance together. The results have shown that there is a strong relationship between ad-hoc routing protocols and MAC protocols. Thus, the paper has concluded that a pair of these protocols should be chosen carefully. In addition, [5] considers not only above three factors but also network traffic and QoS architecture as factors and performs statistical analysis to these five factors by a network simulator. This has also concluded that ad-hoc routing protocols and MAC protocols have a strong relationship for packet delay. Ref. [6] applies many factors to statistical approaches like ANOVA to find parameters that affect packet delivery ratios. The paper has shown that the number of sources, source-destination pairs, packet transmission rate, and propagation model have much greater impact on packet delivery ratios than node speed, node pause time and packet size, which had been considered to have more impact on those. Similar attempts have been made to show this kind of dependencies among multiple factors [7], [8].

Some of these methods require all possible combinations of levels to find dominant factors and reveal the particular performance characteristics. Since general analytical methods require a certain number of test results, they do not consider reducing the number of test cases. Such exhaustive approaches are not realistic for evaluating performance of networks (especially wireless networks). The reasons are as follows: First, running a single simulation or a single experiment of large-scale wireless networks may consume long time. Secondly, for N factors, each of which has M discrete values (levels), there exist M^N test cases.

In this paper, we propose a comprehensive test method to analyze the effect of (usually 10 or more) factors on network protocols in an efficient and systematic manner. The implementation under test (IUT) is a wireless network system that is composed of many nodes, which therefore includes many possible patterns for node deployment and their mobility, heterogeneity of hardware profiles, choices of protocols and their optional parameter settings. Since wireless network simulations or experiments take long time or need much effort, we try to find out dominant factors with fewer test cases. In order to find out dominant factors with fewer test cases, we assume a reasonable fault model so that we can take a stepwise approach that avoids examining all possible test cases. Furthermore, to enable quantitative and automatic detection of factor effects on system performance, we introduce the notion of rank correlation to characterize performance dynamics and provide an appropriate decision policy. We have also conducted two experiments to validate the effectiveness of our method. First, we have tested FTP transmission over TCP on MANET, and could identify high correlation between the TCP version and TCP transmission buffer size. Secondly, we have tested LAR [9], the location aided routing, under different configurations of node deployment, void area sizes and locations, and could find strong relationship among communication range, forwarding zone size, node density and void area size.

II. Related Work and Comparison

Combinatorial tests have been investigated for long time, and its concept has been incorporated into a lot of industrial research and development. The traditional and well-known approach for testing effects of multiple factors is known as *factorial experiments* in the experimental design. In the factorial experiments, the *factorial design* considers all (or a part of) combinations of the levels, and the experimental results by these combinations are analyzed using regression methods, ANOVA (analysis of variances) or some others to observe effects of multiple factors. The well-exploited factorial design is the 2^k factorial design, which deals with k factors and two levels for each one. These levels may be qualitative like "the RWP model" or "the random walk model" in the mobility model selection factor, or may be quantitative like "100" or "200" in the number of mobile nodes.

In Ref. [7], this 2^k factorial design and ANOVA have been applied to analyze the behavior of MANETs. Also in Ref. [5], MANETs have been tested and main effects and interaction of five factors have been analyzed. Some others target different network architectures and protocols [6], [8], [10].

Compared with the above work, our contribution is summarized as follows. First, we provide a comprehensive test strategy with multiple factors and multiple levels for network protocols. Taking into account a reasonable fault model, we try to reduce the total number of test cases. Meanwhile, the focus of the above articles is to reveal or prove particular performance characteristics of particular networks. Therefore, they do not provide methodologies but apply existing analytical methods like ANOVA to analyze the main effect and interactions with help of design experts and with the result of almost all test cases. In the design of experiments, some extensions of 2^k factorial design have been considered to deal with multiple factors and multiple levels. This method can not only find all sets of dominant factors but also show how these sets affect the performance in detail. Therefore, they do not consider reducing the number of test cases since analytical methods require a certain number of test results. On the other hand, our method aims to find only sets of dominant factors that affect the performance strongly without a huge set of test cases. Secondly, we formally define the dominant factors and their interactions, and performance characteristics using the rank correlation coefficient. By these definitions, we can provide a systematic test strategy. Thirdly, these advantages are shown by a realistic case study. We have investigated a case study to see the effects of factors on data transmission over TCP on MANETs.

As far as we know, such a comprehensive test strategy in which factor effects and their interactions are automatically analyzed with the reasonable number of test cases has not been considered in the existing literature.

TABLE I
Performance Factors from Each Network Layer

Layer	Protocol	Factor	Level
MAC	IEEE802.11	*QueueLength*	50, 100, 200 (Packets)
Transport	UDP	*PacketSize*	256, 512, 1024 (Bytes)
Application	CBR Rate	*SendRate*	10, 20, 40 (kbps)

III. Preliminaries

A. Problem Formulation

We consider N attributes $F = (f_1, f_2, ..., f_N)$ which are considered to affect the performance of the target system. They are called *performance factors* (or simply *factors*). A performance factor may be a parameter of a protocol or system such as TCP segment size and maximum wireless range. It may be environmental settings such as a mobility model or even a choice of a protocol (*e.g.* a choice of routing protocols). A subset of the factors is described as $F_s (F_s \subseteq F)$. The difference between sets F and F_s is described as $F \backslash F_s$. We assume that the value domain of each performance factor f is a set of discrete values which are called *levels*, and $L(f)$ denotes the set of levels of f. A *test case* is an assignment of levels to performance factors, and a set of test cases is called a *test suite*. A test case that assigns levels to the factors of F is called a test case $over\ F$. The implementation under test (IUT) is modeled as I taking a test case t as an input and returns the result $I(t)$ for the test case. $I(t)$ is called a *performance measurement value* such as the average throughput and the average end-to-end delay, which can be obtained by (single or replicated) simulations or experiments under the given test case. Table I shows an example of performance factors and levels. We consider evaluating the throughput of UDP transmission with CBR over IEEE802.11 MAC and PHY, varying the CBR rate, UDP packet size and the queue length in IEEE802.11 MAC. Then we choose the three parameters as performance factors, and prepare three levels for each performance factor. A test case for this can be an assignment like (*QueueLength*,

PacketSize, *SendRate*) = (100, 1024, 10), and 27 ($= 3^3$) test cases are possible as a total.

Our objective is to find out all sets of *dominant factors*. Even though it seems difficult to formally define the properties of such sets in general, we try to formalize the problem. We introduce several relations for simplicity of descriptions.

Definition 1: (F_s-equivalence of test cases)
Two test cases t and t' over F are said to be F_s-*equivalent* with respect to the subset F_s *iff* the assignments of levels by t and t' to the factors of F_s are equivalent (and we do not care about levels assigned to the other factors). For example, $t_1 = \{1, 2, 3\}$ and $t_2 = \{3, 2, 3\}$ are F_s-equivalent where F and F_s are $\{f_1, f_2, f_3\}$ and $\{f_2, f_3\}$ respectively. Also they are said to be F_s-*inequivalent* with respect to F_s *iff* there exists at least one factor f of F_s such that the levels assigned to f by t and t' are different.
We introduce binary relations $\equiv_{F_s}$ and $\not\equiv_{F_s}$ to represent F_s-equivalence and F_s-inequivalence, respectively. Next, we introduce "reductions" of test cases and suites.

Definition 2: (F_s-reductions of test cases/suites)
Test case t_s over the subset F_s is said to be F_s-reduction of a test case t over F *iff* t includes t_s in its part. For example, $t_s = \{2, 3\}$ is the F_s-reduction of test case $t = \{1, 2, 3\}$ where F and F_s are $\{f_1, f_2, f_3\}$ and $\{f_2, f_3\}$ respectively. Similarly, a test suite T_s over the subset F_s is said to be F_s-*reduction* of a test suite T over F *iff* for any $t_s \in T_s$, there exists a test case t in T where the F_s-reduction test case of t is equivalent to t_s and for any $t' \in T$, there exists a test case t'_s in T_s where the F_s-reduction test case of t' is equivalent to t'_s.

Definition 3: (t_s-homogeneity of test suite)
For test case t_s over the subset F_s, a test suite T over F is said to be t_s-*homogeneous iff* for any test case t'_s in F_s-reduction test suite of T, t'_s equals t_s. Informally, if T is t_s-homogeneous, then T consists of test cases that include t_s as their parts.

Definition 4: (F_s-completeness of test suite)
Hereafter, we let F^* denote the test suite that contains all possible test cases over F. This is called complete test suite over F. A test suite T is said to be F_s-complete *iff* F_s-reduction of T is complete.

Our definition of dominant factors is following.

Definition 5: A subset F_s is suspected to be a set of dominant factors iff we cannot observe "distinctive" performance characteristics for a variety of test case pairs where each pair of t and t' satisfies $t \equiv_{F_s} t'$.
In other words, we cannot observe distinctive performance characteristics under the test cases that assign the same levels to the factors of F_s. The test cases shown in Table II satisfy the above conditions with respect to $F_s = \{f_1, f_2\}$. If we do not see any change of performance characteristics using the test cases, we can expect that f_3 and f_4 are not dominant factors.

We might also consider another definition that is more straightforward.

Definition 6: A subset F_s is suspected to be a set of dominant factors *iff* we can observe "distinctive" performance characteristics for a variety of test case pairs where each pair of t and t' satisfies $t \not\equiv_{F_s} t'$ and $t \equiv_{F \setminus F_s} t'$.
The test cases shown in Table II satisfy the above condition with respect to $F_s = \{f_3, f_4\}$. If we can observe significant change of performance characteristics with the same test cases, we can expect that f_3 and f_4 are not dominant factors.

TABLE II
A EXAMPLE TEST SUITE TO VALIDATE DOMINANT FACTORS

	f_1	f_2	f_3	f_4
t_1	1	1	1	1
t_2	1	1	1	2
t_3	1	1	2	1
t_4	1	1	2	2

B. Challenges in Finding Dominant Factors

To find sets of dominant factors according to the above definition, we need to do the followings; (i) we may need to apply many tests to the target system to examine if each possible F_d is a dominant factor or not, and (ii) we need to identify "distinctive" performance characteristics for given two test cases.

The performance measurement value $I(t)$ for a single test case t may be an aggregated value obtained from a number of simulations or field experiments with settings determined by test case t. For example, let us assume that I is the average throughput of TCP connection over MANET. Then for each given test case t, we may repeat simulations with the settings determined by t varying random seeds to observe the well-averaged value. This indicates that we need a considerable amount of time to get $I(t)$ for each t. In addition, in large-scale ad-hoc wireless networks, each simulation itself may take long time since it consumes much computer resources to calculate collision by interference in geographical region, mobility of nodes and so on. Also field experiments need much more efforts to set up and control wireless terminals in real environments. Consequently, obtaining each $I(t)$ needs a considerable amount of time even though simulation technologies have been improved recently and computing capability has grown rapidly.

Also, for the second problem, we should provide a reasonable and deterministic policy to observe the distinctive performance characteristics semi-automatically. In other words, we should design the test cases and the corresponding decisions without ambiguity. This is deeply related with the "good" test case selection under the limitation of their total amount.

Considering the above discussions, our goal is to design a comprehensive method to find out all sets of dominant factors with a reasonable number of possible test cases. In the following section, we exemplify dominant factors. After that, we briefly introduce pairwise test generation methods that are used in a part of our algorithm.

C. Example of Dominant Factors

In this section we exemplify dominant factors. We use a well-known result on the throughput fluctuation in TCP transmission over MANET [11], [12]. In more details, TCP over MANET may become unstable due to frequent retransmission

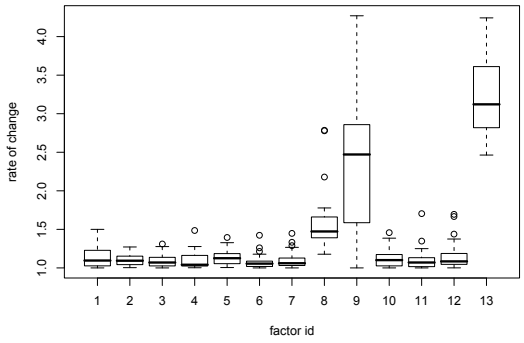

Fig. 1. Ranges of Normalized Jitters (in varying levels of single factor)

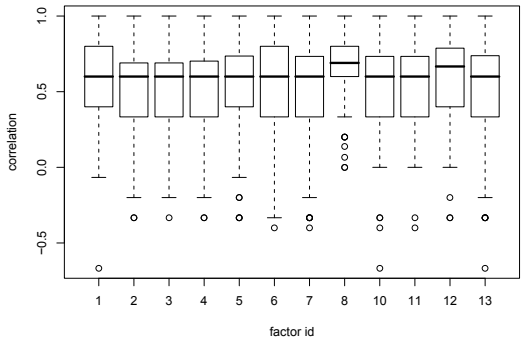

Fig. 2. Spearman Rank Correlation Coefficients (SRCC); on i-th label of X-axis, SRCCs of jitters from test suites where *AdvWin* and factor i are equivalent are plotted.

caused by MAC level collisions, route discovery in the network layer, or network partitions by node mobility. In previous researches, it has been reported that the control of the TCP advertise window and the TCP segment size are very important for stable communication. To evaluate the stability of TCP connections, we confirm that the TCP advertise window and the TCP segment size form a set of dominant factors. Through simulation experiments, we have measured the median of packet jitters at the receiver as the performance measurement. We have conducted simulations with the performance factors and their levels presented in Table III. In these simulations, only one TCP connection between two end nodes located at the opposite corners is established through stationary nodes deployed in the simulation field.

We first examine if each single factor is in dominant factors or not. For this purpose, we show the dynamic range of the packet jitters under the test cases where only the levels of a single factor are different. For each factor f_i, we prepare a test suite T_i where for each test case we set the representative levels to the factors except f_i and one of the levels to f_i. Therefore, T_i is a test suite that satisfies $t \equiv_{F \setminus \{f\}} t'$ for $\forall t, t' \in T_i$. Fig. 1 shows a box-whisker chart in which the measured jitters using T_i are plotted on the i-th label of the x-axis. The jitters plotted on the i-th label are normalized by the minimal value among them and the box shows their first

TABLE III
PERFORMANCE FACTORS AND THEIR LEVELS (FOR TCP OVER MANET WITH DSR AND IEEE802.11MAC)

MAC Sub-Layer (IEEE 802.11 DCF)		
Factor ID	Factor	Levels
1	QueueLength	50, 100, 200, 400
2	CWMin	31, 63, 127, 255
3	CWMax	1023, 2407, 4095, 8191

Network Layer (DSR)		
Factor ID	Factor	Levels
4	RouteCacheTimeout	4, 8, 16, 32, 300
5	SendBufferTimeout	15, 30, 60, 120
6	SendBufferSize	16, 32, 64, 128
7	EnableRingSearch	true, false

Transport Layer (TCP Tahoe)		
Factor ID	Factor	Levels
8	SegmentSize	256, 512, 1024, 2048
9	AdvWin	256, 512, 2048, 4096, 8192, 16384
10	RtoMin	0, 1, 2, 4
11	RtoMax	20, 40, 80, 160

Environmental Settings		
Factor ID	Factor	Levels
12	NodeSize (number of nodes in $100m \times 100m$)	4, 8, 16
13	RegionSize (m × m)	200 × 200, 400 × 400, 600 × 600

and third quarters. From this graph and Definition 6, we can say that { Factor8 (*SegmentSize*) }, { Factor9 (*AdvWin*) } and { Factor13 (*RegionSize*) } are dominant factors since we can observe distinctive performance variance on the 8th, 9th and 13th labels of the x-axis[1].

Then we focus on the correlation of *AdvWin* and *SegmentSize* to see that they form a set of dominant factors as reported in Refs. [11] and [12]. According to Definition 5, in order to confirm that we cannot observe the distinctive performance characteristics under any pair of test cases t and t' where the levels of *AdvWin* and *SegmentSize* are equivalent *i.e.* $t \equiv_{\{AdvWin, SegmentSize\}} t'$, we have measured the jitters for a variety of test suites where satisfy $T \equiv_{\{AdvWin, SegmentSize\}} T'$ and T. We have also calculated the *Spearman rank correlation coefficient* (SRCC) of them. To explain the set of dominant factors intuitively, here we roughly say that SRCC represents a relationship between the performance measurement values measured by two test suites T and T', and its value is close to -1 or 1 if these values have strong relationship (that is, T and T' cause no or little distinctive performance characteristics), and close to 0 otherwise (that is, T and T' cause distinctive performance characteristics).

Fig. 2 shows a box-whisker chart of coefficients where on i-th label of x-axis the SRCCs of jitters from each pair of test suites are plotted where *AdvWin* and factor i are equivalent. From this result, we can see that on the 8th label that plots the cases where *AdvWin* and *SegmentSize* (factor 8) are equivalent, the first quarter (1Q) is around 0.6.

[1]According to Definition 6, as T_i, we actually need to consider all the possible test cases that satisfy $t \equiv_{F \setminus \{f\}} t'$ and $\forall t, t' \in T_i$. Meanwhile for the purpose of exemplifying dominant factors, we only consider single T_i using default (representative) levels.

TABLE IV
EXAMPLE OF PAIRWISE TEST SUITE

Case	*QueueLength*	*PacketSize*	*SendRate*
1	100	1024	10
2	200	256	40
3	50	512	10
4	200	1024	20
5	50	256	20
6	100	512	40
7	200	256	10
8	50	1024	40
9	100	256	20
10	200	512	20

Meanwhile, for the other parameters, the first quarter is around 0.3. This means that using test suites which are equivalent over $\{AdvWin, SegmentSize\}$, we cannot see distinctive performance characteristics because SRCCs are high. From this fact, we can say that $\{AdvWin, SegmentSize\}$ is a set of dominant factors.

D. Pairwise Test

For efficient test case generation, we use the idea of pairwise test in part of our algorithm. Although using the pairwise method is not essential to reduce the test cases in our case, we explain it to clarify our algorithm explanation given later.

In traditional software development, it is considered that specific combinations of parameters can reveal most faults. Then a k-wise test suite, in which any k-tuple of parameters can be found, has been applied for such purpose. Even though the size of k depends on the scale of software, a k-wise test suite is considered effective through several case studies [13]. In other application domains, it has been reported that k-wise test methods are applied to functional test of compilers [14] and interoperability test of network interfaces [15].

Formally, for a test suite R and a positive integer k $(1 \leq k \leq N)$, if any assignment of levels to k-factors appears in at least one of test cases in R, then R is called a k-wise test suite. For example, we consider the previous example of Table I with three performance factors (from the MAC sub-layer, the transport layer and the application layer) each of which has three levels. An example 2-wise test suite is shown in Table IV. We focus on two factors, *QueueLength* and *PacketSize*. Their possible assignments are followings; (50, 256), (100, 256), (200, 256), (50, 512), (100, 512), (200,512), (50,1024), (100,1024) and (200, 1024), and we can find them in test cases No. 5, 9, 2 (and 7 also), 3, 6, 10, 8, 1 and 4, respectively. Here the important feature is that for any pair of factors, that is, (*QueueLength*, *PacketSize*), (*QueueLength*, *SendRate*) or (*PacketSize*, *SendRate*), any assignment to the pair can be found in at least one of the test cases if the test suite is a pairwise test suite. The complete test suite includes 27 (3^3) test cases, while this 2-wise test suite includes only 10 test cases.

IV. PROPOSED TEST METHOD

A. Basic Idea and Outline

1) How to Generate Reasonable Test Cases for Dominant Factor Examination: For a given subset F_d $(\subseteq F)$, we prepare a test suite T_d over F satisfying the following conditions to examine if F_d is a set of dominant factors or not.

1) T_d is F_d-complete. This means that, for any combination of levels of factors in F_d, T_d contains at least one test case that contains the combination.
2) $\forall f \in F \backslash F_d$, $\forall u \in \{f\}$-reduction of T_d;
   a) $\exists u' \in \{f\}$-reduction of T_d;
      $(u \equiv_{F_d} u') \cap (u \not\equiv_{F \backslash (F_d \cup \{f\})} u')$.
      We ignore a factor f that is not in F_d. Then for any test case t, there exists a test case u' where u and u' have the common assignment of levels to F_d and have different assignments of levels to at least one factor not in F_d (except f).
   b) $\{u\} \times \{f\}^* \in T_d$ where $T_i \times T_j$ denotes the set of all test cases generated by the product of two (partial) test suites T_i and T_j.
      This means that, for u and f, all the combinations of u and the levels of f are in T_d.

TABLE V
A PART OF TEST SUITE TO EXAMINE F_d ($F_d = \{f_1, f_2\}$)

	f_1	f_2	f_3	f_4		f_1	f_2	f_3	f_4
t_0	0	0	2	0	t_{27}	0	0	1	0
t_1	0	0	2	1	t_{28}	0	0	1	1
t_2	0	0	2	2	t_{29}	0	0	1	2
t_3	0	1	1	0	t_{30}	0	1	0	0
t_4	0	1	1	1	t_{31}	0	1	0	1
t_5	0	1	1	2	t_{32}	0	1	0	2
t_6	1	0	2	0	t_{33}	1	0	1	0
t_7	1	0	2	1	t_{34}	1	0	1	1
t_8	1	0	2	2	t_{35}	1	0	1	2
t_9	1	1	0	0	t_{36}	1	1	2	0
t_{10}	1	1	0	1	t_{37}	1	1	2	1
t_{11}	1	1	0	2	t_{38}	1	1	2	2
...		...			...		...		
t_{24}	2	2	1	0	t_{51}	2	2	2	0
t_{25}	2	2	1	1	t_{52}	2	2	2	1
t_{26}	2	2	1	2	t_{53}	2	2	2	2

TABLE VI
$\{f_4\}$-REDUCTION OF TEST SUITE OF TABLE V

Case ID	f_1	f_2	f_3		f_1	f_2	f_3
u_0	0	0	2	u_{27}	0	0	1
u_3	0	1	1	u_{30}	0	1	0
u_6	1	0	2	u_{33}	1	0	1
u_9	1	1	0	u_{36}	1	1	2
...		...		...		...	
u_{24}	2	2	1	u_{51}	2	2	2

As an example, let us assume $F = \{f_1, f_2, f_3, f_4\}$, $F_d = \{f_1, f_2\}$, and $L(f_i) = \{0, 1, 2\}$ $(1 \leq i \leq 4)$. In Table V, we show a part of test suite T_d. We can see that this T_d satisfies condition 1 since all the possible combinations of levels of f_1 and f_2 appear. Also its $\{f_4\}$-reduction is shown in Table VI where each u_i corresponds to t_i, t_{i+1} and t_{i+2}. We can see that condition 2-(a) is satisfied for the case of $f = f_4$ since $u_i \equiv_{\{f_1, f_2\}} u_{i+27}$ and $u_i \not\equiv_{\{f_3\}} u_{i+27}$ for $i = 0, 3, 6, .., 24$. Similarly, condition 2-(b) is satisfied since for each u_i of $\{f_4\}$-reduction, $\{u_i\} \times \{f_4\}^*$ corresponds to test cases t_i, t_{i+1} and t_{i+2} in T_d. In the same way, we can generate the rest of T_d that satisfies condition 2 for the case of $f = f_3$ (we omit those test cases for the limitation of space).

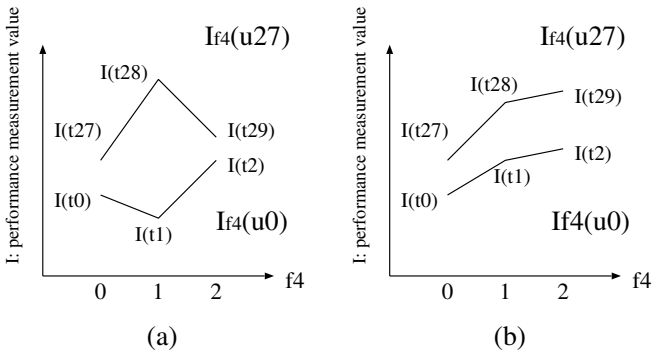

Fig. 3. Comparison of Performance Sequences $\vec{I}_{f_4}(u_0) = (I(t_0), I(t_1), I(t_2))$ and $\vec{I}_{f_4}(u_{27}) = (I(t_{27}), I(t_{28}), I(t_{29}))$

By condition 1, T_d is ensured to have test cases that include all the possible assignments of levels to F_d. Additionally with condition 2-(a), for each level assignment to F_d, it is ensured to have at least one pair of F_d-equivalent test cases with different level assignments to the factors of $F \backslash F_d$. If F_d-equivalent test cases with a common level assignment to F_d do *NOT* reveal "distinctive" performance characteristics, we can say that F_d is suspected to be a set of dominant factors according to definition 5.

The challenge here is how we generate such a set of test cases that satisfy condition 1 and condition 2-(a) with reasonable "resolution" and with a reasonable number of test cases. The "resolution" requirement is that for each level assignment to F_d, we should have a variety of assignments to $F \backslash F_d$ to examine the effects of $F \backslash F_d$ excluding the effects of F_d. To generate such test cases, we partly apply k-wise test methodology. Actually, for F_d with k factors, $(k+1)$-wise test suite over $F \backslash \{f\}$ satisfies condition 1 and condition 2-(a) since the test suite is F_d-complete from the definition of the k-wise test suite and there exist at least two test cases t and t' where $t \equiv_{F_d} t'$ and with different assignments to a factor of $F \backslash (F_d \cup \{f\})$. This is the idea of generating test cases to examine a given set F_d of factors.

2) How to Identify Performance Characteristics: We exploit condition 2-(b) to identify performance characteristics caused by F_d. If T_d satisfies condition 2-(b), the combination of each $\{f\}$-reduction test case (say u) and each level of f is in T_d. This means that we can obtain the "sequence" of test results by applying the products of u and all the levels of f. Such a sequence is called a *performance sequence*. Hereafter $\vec{I}_f(u)$ denotes a performance sequence obtained by varying the levels of factor f and the assignment of levels to the other factors is defined by test case u. In other words, this sequence is composed of each $I(t)$ where t is in u-homogeneous test suite. To identify performance characteristics more clearly, for each pair of $\{f\}$-reduction test cases u and u' where $u \equiv_{F_d} u'$, we compare $\vec{I}_f(u)$ and $\vec{I}_f(u')$ by calculating their *Spearman Rank Co-relation Coefficient* (SRCC in short) [16]. SRCC is given below;

$$\rho = \frac{\sum_{i=1}^{n} \left(r_i - \frac{n+1}{2}\right)\left(s_i - \frac{n+1}{2}\right)}{\sqrt{\sum_{i=1}^{n} \left(r_i - \frac{n+1}{2}\right)^2 \left(s_i - \frac{n+1}{2}\right)^2}} \quad (1)$$

where r_i is the rank of x_i (incremental order) in x_1, ..., x_n and s_i is the rank of y_i (incremental order) in y_1, ..., y_n. In order to obtain SRCC of two performance sequences, we let i-th values of performance sequences $\vec{I}_f(u)$ and $\vec{I}_f(u')$ correspond to x_i and y_i, respectively. The coefficient is between -1 and 1. If it is close to 0, the relationship between two sequences is weak. If the coefficient is close to -1 or 1, there is a strong relationship between them. In this case, the factors of F_d affect the performance in the sense that this similarity is brought by sharing the level assignment to F_d. For example, in Fig. 3, we show two sequences $\vec{I}_{f_4}(u_0) = (I(t_0), I(t_1), I(t_2))$ and $\vec{I}_{f_4}(u_{27}) = (I(t_{27}), I(t_{28}), I(t_{29}))$. In Fig. 3 (a), their SRCC is not close to -1 or 1 since the ranks of $\vec{I}_{f_4}(u_0)$ and $\vec{I}_{f_4}(u_{27})$ are (2,3,1) and (3,1,2), respectively. This means that these different treads of performance dynamics are brought by other factors than f_1 and f_2, and in this sense $\{f_1, f_2\}$ is not dominant over the others. Meanwhile, in Fig. 3 (b), it is high since their ranks are (3,2,1). This means that $\{f_1, f_2\}$ may be dominant over the others. To confirm the domination of F_d with the other assignment, we calculate the coefficient of $\vec{I}_f(u)$ and $\vec{I}_f(u')$ for every pair of factor $f \in F \backslash F_d$ and $u, u' \in \{f\}$-reduction of T_d, and we take the minimum coefficient among them. If this minimum coefficient is still above the predetermined threshold, then the coefficient of any other pair is larger than the threshold, and we can prove that the assignment of common levels to the factors of F_d have strong effect to make the performance characteristics very similar. Before this process, we should choose an appropriate value to the predetermined threshold for the target system so that we can know whether there is a strong relationship between two sequences. Otherwise, we may not be able to find sets of dominant factors, which are useful for performance evaluation. This is the idea on identifying the effects of F_d.

3) How to Reduce the Total Number of Test Cases: It is still too expensive with respect to the number of test cases if we generate an independent set of test cases for each possible F_d, since we have $\sum_{i=1..n} \frac{N!}{i!(N-i)!}$ candidate sets of factors and for each candidate set we need a certain amount of test cases. Then we take a stepwise approach. In this approach, at step k we examine the candidate sets with k factors, but non-suspected sets have already been excluded using the results of examination at the previous step $k-1$. However, to employ such a stepwise approach where the candidate sets are narrowed down at each step, we need to justify the cut-down process. In our case, we assume the following reasonable property as a fault model.

Property 1: (Fault Model) For any set of k dominant factors, at least one subset of $k-1$ factors is also a set of dominant factors.

This is in general true in the following reason. We do not assume that specific assignment of levels drastically increases or decreases the performance. Instead, we assume that each dominant factor has impact on the performance by itself to some or a great extent. For example, in many types of communications, multiplicity of packet size and bitrate should not be greater than the channel capacity, and the throughput

will decrease gradually beyond this capacity. In this case, larger packet size itself (or higher bitrate as well) may increase the throughput. Assuming the above property, first, each set of a single factor is examined. Then for each F_d of k-factors that is suspected to be a set of dominant factors, we examine each F'_d of $k+1$ factors where $F'_d \supset F_d$. This contributes to enable the stepwise examination starting from $k=1$ and to reduce the number of test cases compared with some other factorial design methods that need to apply all or a part of combinations before analysis.

B. Algorithm Description

Our test procedure takes as inputs (i) a given set F of factors, (ii) a given set $L(f)$ of levels for each factor $f \in F$ and (iii) the target system I that is a function of a test case. We assume $|F| > 2$, and we introduce a parameter k and a family DS_k of sets over F (*i.e.* $DS_k \subseteq \mathcal{P}(F)$). DS_k consists of sets of k dominant factors, and $\bigcup_{k=1..|F|} DS_k$ is the output of the algorithm. We let T_k denote a test suite over F used in the k-th iteration of the algorithm.

Initially, we start the algorithm with $k=1$, $DS_1 = \bigcup_{f\in F}\{\{f\}\}$, $DS_i = \emptyset$ $(i \geq 2)$ and $T_0 = \emptyset$. The formal description of the test procedure is given as follows. We note that later we will validate this algorithm according to the discussion in Section IV-A.

1) We prepare the test suite which consists of test cases derived by the production of (i) the complete test suite over $F_d \in DS_k$ where F_d is a candidate set of factors to be examined (DS_k has been determined by the (k-1)-th iteration), (ii) 2-wise test suite over $F\backslash(F_d \cup \{f\})$ (denoted as R), and (iii) the complete test suite over $\{f\}$. To exclude the same test cases, we generate only the test cases that are not included in T_{k-1}.

$$T_k \leftarrow T_{k-1} \cup \left\{ \bigcup_{F_d\in DS_k, f\in F\backslash F_d} F_d^* \times R \times \{f\}^* \right\}$$

2) We apply each test case t of $T_k \backslash T_{k-1}$ to the target system to obtain $I(t)$.
3) For each pair of $f \in F$ and test case t' in $\{f\}$-reduction of T_k, we obtain the performance sequence $\vec{I}_f(t')$.
4) For each $f \in F$, each $F_d \in DS_k$ and test cases t' and t'' in $\{f\}$-reduction of T_k where $t' \equiv_{F_d} t''$, calculate the Spearman rank correlation coefficient (SRCC) of two performance sequences $\vec{I}_f(t')$ and $\vec{I}_f(t'')$. This SRCC is denoted by $C[\vec{I}_f(t'), \vec{I}_f(t'')]$.
5) For each $F_d \in DS_k$, we define

$$\begin{aligned} &min[\equiv_{F_d}] \\ &= \min\{\, abs(\, C[\vec{I}_f(t'), \vec{I}_f(t'')]\,) \mid \\ &\quad \forall f \in F,\ \forall t', t'' \in \{f\}\text{-reduction of } T_k,\ t' \equiv_{F_d} t'' \,\} \end{aligned}$$

where $abs(x)$ is the function that returns the absolute value of x. For each $F_d \in DS_k$, we let F_d^{+1} denote each set of factors where $F_d^{+1} \supset F_d$ and $|F_d^{+1}| = k+1$. For each F_d^{+1}, if $min[\equiv_{F_d^{+1}}] \sim min[\equiv_{F_d}]$ and $min[\equiv_{F_d}] < Th_{high}$ where Th_{high} is a lower threshold of SRCC,

$$DS_k \leftarrow DS_k \backslash \{F_d\}$$

else if $min[\equiv_{F_d^{+1}}] \not\sim min[\equiv_{F_d}]$ and $min[\equiv_{F_d}] < Th_{high}$, where $|u^+| = k-1$ and $min[\equiv_{F_d^{+1}}] - min[\equiv_{F_d}] > \Delta Th$, where ΔTh is a lower threshold of SRCC increment,

$$DS_{k+1} \leftarrow DS_{k+1} \cup \{F_d^{+1}\}$$

6) If $DS_{k+1} = \emptyset$, exit this procedure with return value $\bigcup_k DS_k$. Otherwise jump to the first step with $k \leftarrow k+1$.

To validate the algorithm, we show that the algorithm tests the given system using a test suite that satisfies all the conditions in Section IV-A. For this purpose, we show that each T_k satisfies the conditions in Section IV-A. Obviously, it satisfies condition 1 and condition 2-(b) since it contains the complete test suite F_d^* and $\{f\}^*$ for each $f \in F\backslash\{F_d\}$. In addition, since it contains 2-wise test suite R over $F\backslash\{F_d \cup \{f\}\}$, which is combined with the complete test suite F_d^*, condition 2-(a) is also satisfied.

Then we explain that the algorithm finds sets of dominant factors assuming Property 1, which is the fault model. Due to Property 1, any set F_d of k factors can be found if all the sets of $k-1$ factors have been found and if we examine the sets of k factors which include these $k-1$ dominant factors. We note that we would like to check at step 5 of k-th iterations whether the set of k factors has similar performance characteristics or not. This is necessary since we need to determine whether we should continue to examine larger sets or not. For this purpose, we let R be a 2-wise test suite so that T_k has the sufficient test cases to examine the sets of $k+1$ factors.

V. Case Study

A. FTP over AODV and IEEE 802.11

At first, in order to show how our method works, we have applied our method to file transfer over MANETs. As performance factors, we have considered several internal parameters of AODV, TCP and IEEE802.11 MAC. We have also considered the choice of node density and region size as the performance factors. These factors accompanied by their levels are summarized in Table VII. As we can see, we have dealt with 29 factors ($|F| = 29$). In the experiments, we located two nodes in near the corners of the field and have measured jitters at the receiver with given parameters by QualNet [17].

We started the algorithm with $k=1$ and $DS_1 = \{\{f\} \mid \forall f \in F\}$. 3,500 test cases were generated as T_1 and these jitters are measured by varying the levels of a single factor. For each $F_d = \{f\}$, we have calculated SRCCs of performance sequences where the assignments of levels to f are the same. From the results, SRCCs were high where F_d is {TCP}, {MSS}, {SEND-BUFFER} or {MAP}. Thus, DS_1 became {{TCP}, {MSS}, {SEND-BUFFER}, {MAP}} after

TABLE VII
PERFORMANCE FACTORS AND THEIR LEVELS (AODV PROTOCOL)

MAC Layer (IEEE 802.11 DCF with RTS/CTS)		
ID	Factor	Level
1	LONG-PACKET-TRANSMIT-LIMIT	1, 4 , 7, 13
2	RTS-THRESHOLD	0, 730. 1460

Routing (AODV)		
ID	Factor	Level
3	NET-DIAMETER	10, 35, 50
4	NODE-TRAVERSAL-TIME(ms)	10, 40, 160
5	ACTIVE-ROUTE-TIMEOUT(s)	1, 3, 10
6	MY-ROUTE-TIMEOUT(s)	1, 6, 10
7	HELLO-INTERVAL(s)	1, 3, 10
8	ALLOWED-HELLO-LOSS	1, 2, 4
9	RREQ-RETRIES	1, 2, 4
10	ROUTE-DELETION-CONSTANT	1, 5, 10
11	PROCESS-HELLO	NO, YES
12	LOCAL-REPAIR	NO, YES
13	SEARCH-BETTER-ROUTE	NO, YES
14	BUFFER-MAX-PACKET	50, 100, 200
15	BUFFER-MAX-BYTE	0, 1000, 100000
16	OPEN-BI-DIRECTIONAL-CONNECTION	YES, NO
17	TTL-START	1, 5, 10
18	TTL-INCREMENT	2, 4, 8
19	TTL-THRESHOLD	5, 15, 25

Transport (TCP)		
ID	Factor	Level
20	TCP	TAHOE, RENO, LITE, SACK, NEWRENO
21	DELAY-ACKS	YES, NO
22	DELAY-SHORT-PACKETS-ACKS	NO, YES
23	USE-NAGLE-ALGORITHM	YES, NO
24	USE-KEEPALIVE-PROBES	YES, NO
25	USE-PUSH	YES, NO
26	MSS	256, 512, 1024, 1460
27	SEND-BUFFER	1024, 4096, 16384, 65535

Environment		
ID	Factor	Level
28	NODE DENSITY (per 100m × 100m)	4, 8, 16
29	MAP (m × m)	200 × 200, 400 × 400, 600 × 600

the first test. In addition, we have found that SRCCs of $F_d^{+1} = \{\{\text{TCP}\}, \{\text{SEND-BUFFER}\}\}$ were higher than these SRCCs of F_d={TCP} or F_d={SEND-BUFFER}. This means that { TCP, SEND-BUFFER } might be a set of dominant factors and should be tested in the next step. Therefore, we let $k = 2$ and $DS_2 = \{\{\text{TCP, SEND-BUFFER}\}\}$, and went into the second iteration.

The second iteration started with $DS_2 = \{\{\text{TCP, SEND-BUFFER}\}\}$. We have examined if $F_d = \{\text{TCP, SEND-BUFFER}\}$ was a set of dominant factors or not. For this purpose, we have generated test suite T_2 where $T_2 \backslash T_1$ had only 2,060 test cases, and calculated SRCC for each pair of performance sequences $\vec{I}_f(t)$ and $\vec{I}_f(t')$ where t and t' are in the {f}-reduction of T_2 and $t \equiv_{F_d} t'$ for each f. Since they had similar SRCCs, we concluded that {TCP, SEND-BUFFER} was the maximum set of dominant factors which includes {TCP, SEND-BUFFER}. Thus we obtained $DS_3 = \emptyset$, the algorithm terminated at the end of the second iteration. Through this case study, {TCP, SEND-BUFFER}, {MAP} and {MSS} are the sets of dominant factors that affect the packet jitters between two nodes on MANETs.

TABLE VIII
PERFORMANCE FACTORS AND THEIR LEVELS (LOCATION-AIDED ROUTING PROTOCOL)

MAC Layer		
ID	Factor	Level
1	CommunicationRange (m)	100, 125, 150, 200, 250

Routing (Location-Aided Routing)		
ID	Factor	Level
2	ExtraSize (of Request Zone) (m)	0, 10, 25, 50, 100, 150

IP		
ID	Factor	Level
3	QueueLength	25, 50, 100
4	FragmentSize	512, 1024

Application		
ID	Factor	Level
5	PacketLength	128, 256, 512, 1024
6	PacketInterval (ms)	1, 5, 10, 25, 50, 100

Environment		
ID	Factor	Level
7	NodeDensity (per 100m × 100m)	1, 2.5, 5.0, 7.5, 15.0
8	VoidSize (m)	0, 50, 100, 150, 200, 250, 300

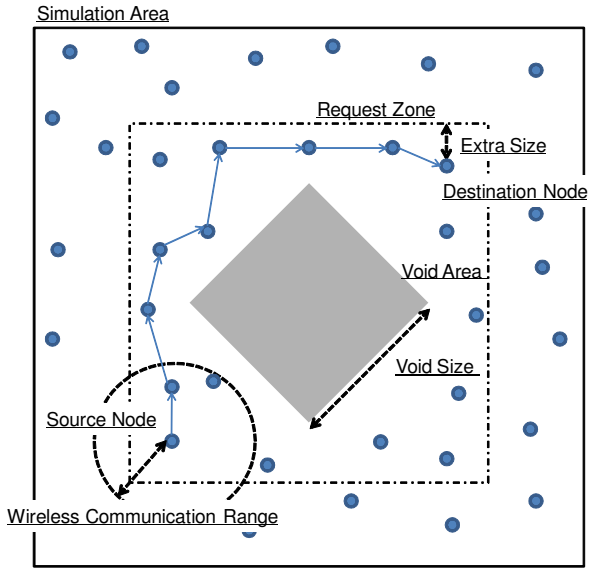

Fig. 4. Simulation Environment

B. Location-Aided Routing Protocol

We have also applied our method to test the Location-Aided Routing protocol [9]. As shown in Fig. 4, in the square region of 900m × 900m, we put a source node (src), a destination node (dst) and many other relay nodes. We artificially generated a square region without nodes called a void area, and its side length is denoted by VoidSize. If src would like to deliver data to dst, src transmits 100 packets of a certain size denoted by PacketLength, at regular time intervals denoted by PacketInterval. In this case study, the nodes transmitted their packets on the routes constructed by the LAR scheme 1. We have fixed the location of all nodes. In

order to vary the number of relay nodes, we have changed the size of the expected zone denoted by ExtraSize, which is the additional length of the standard zone size. Varying the levels of the above and other factors summarized as Table VIII, we have measured the average delay from src to dst using the Qualnet [17].

At the first iteration with $k = 1$, SRCCs of QueueLength and FragmentSize were 0.991 and 0.981, respectively. In addition, SRCCs of any pair of QueueLength and the other factor are almost same as that of QueueLength. Similarly, SRCCs of any pair of FragmentSize and the other factor are almost same as that of FragmentSize too. This means that these factors have less effects on the delay. QueueLength is the length of the packet queues of the nodes. If QueueLength is smaller, some packets might be dropped if a relay node cannot keep packets in its queue. However, no packet was dropped at the relay nodes through the simulations and QueueLength did not affect the delay in this case study. Similarly, packet fragmentation was not observed since the sizes of most packets were smaller than FragmentSize, which is the maximum size of packets in the network. For the other factors, their SRCCs were high but below 0.9, and the algorithm brought them to the next step with $k = 2$.

At the second iteration with $k = 2$, SRCC of {PacketSize, PacketInterval} was 0.983, which means a strong relationship. This is natural since forwarding delay at each relay node depends on the total amount of packets, which is determined by PacketSize and PacketInterval. Since SRCC of this combination was almost 1, we stopped investigating further combinations and determined that {PacketSize, PacketInterval} was a set of dominant factors. Meanwhile, the algorithm brought the other factor combinations to the next step with $k = 3$.

At the third iteration with $k = 3$, SRCC of {CommunicationRange, NodeDensity, VoidSize} was 0.913, that of {ExtraSize, NodeDensity, VoidSize} was 0.963 and that of {CommunicationRange, ExtraSize, NodeDensity} was 0.936. For their superset {CommunicationRange, ExtraSize, NodeDensity, VoidSize}, its SRCC was almost 1.0. Therefore, the algorithm stopped its investigation and {CommunicationRange, ExtraSize, NodeDensity, VoidSize} was regarded as another set of dominant factors.

VI. Conclusion

In this paper, we have proposed a comprehensive and systematic test strategy for network protocols that are operated in diverse environment. Our goal is to find out dominant factors with less test cases, since (i) IUT may be a wireless network system that is composed of many nodes, which therefore includes many possible patterns for node deployment and their mobility, heterogeneity of hardware profiles, choices of protocols and their optional parameter settings, and (ii) wireless network simulations or experiments take long time or need much effort. For this objective, we provide a comprehensive test strategy with multiple factors and multiple levels for network protocols. Taking into account a reasonable fault model, we try to reduce the total number of test cases. Meanwhile, the focus of existing papers is to reveal or prove the particular performance characteristics of particular networks. Additionally, we formally define the dominant factors and their interactions, and performance characteristics using the rank correlation coefficient to provide a systematic test strategy. The advantages of our method are shown by several realistic case studies.

Our ongoing work includes the development of GUI for our toolset to improve the usability and to attract the wide variety of simulator users. Also, we are going to test other systems like wireless sensor networks. They have their specific performance factors regarding hardware settings like battery capacity, memory capacity, wireless range and so on, and the corresponding performance metrics may be different.

References

[1] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET simulation studies: the incredibles," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50–61, 2005.

[2] V. Rodoplu and A. Aminzadeh Gohari, "Challenges: automated design of networking protocols," in *Proc. of the 14th ACM Int. Conf. on Mobile computing and networking (Mobicom2008)*, 2008, pp. 271–278.

[3] D. C. Montgomery, *Design and Analysis of Experiments*, 7th ed. Wiley, 2008.

[4] C. Barrett, M. Drozda, A. Marathe, and M. Marathe, "Characterizing the interaction between routing and MAC protocols in ad-hoc networks," in *Proc. of ACM MobiHoc*, 2002.

[5] K. Vadde and V. Syrotiuk, "Factor interaction on service delivery in mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1335–1346, Sept. 2004.

[6] S. Kurkowski, W. Navidi, and T. Camp, "Discovering variables that affect MANET protocol performance," in *Proc. of GLOBECOM '07*, 2007, pp. 1237–1242.

[7] M. W. Totaro and D. D. Perkins, "Using statistical design of experiments for analyzing mobile ad hoc networks," in *Proc. of 8th ACM Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM2005)*, 2005, pp. 159–168.

[8] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," in *Proc. ACM MobiCom*, 1999, pp. 195–206.

[9] Y. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.

[10] A. Goulart and W. Zhan, "A design of experiment (DOE) analysis of the performance of uplink real-time traffic over a 3G network," in *Proc. of IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communication (WiMob2008)*, 2008, pp. 466–471.

[11] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, 2001.

[12] S. Xu and T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks," *Computer Networks*, vol. 38, no. 4, pp. 531–548, 2002.

[13] R. Kuhn, D. R. Wallace, and A. M. G. Jr., "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 418–421, 2004.

[14] R. Mandl, "Orthogonal Latin squares: an application of experiment design to compiler testing," *Communications of the ACM*, vol. 28, no. 10, pp. 1054–1058, 1985.

[15] A. W. Williams and R. L. Probert, "A practical strategy for testing pair-wise coverage of network interfaces," in *Proc. 7th International Symposium on Software Reliability Engineering (ISSRE '96)*, 1996, pp. 246–254.

[16] J. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 578–580, 1972.

[17] "Qualnet," http://www.scalable-networks.com/.