Leveraging P2P Networks to address the Test Scenario Explosion Problem

Mark Micallef, Conrad Attard, Andrea Mangion, Sebastian Attard
University of Malta
Malta

ABSTRACT

The behaviour of software is influenced by whatever environment it happens to be deployed in. Achieving a sufficient level of coverage for all deployment scenarios during lab testing is difficult for even the most resource-rich organisation. We refer to this as the Test Scenario Explosion Problem and propose the construction of a peer-to-peer network which facilitates the quick creation of large-scale virtual test labs that are representative of a company’s customer base. Following an outline of our initial ideas in this regard, a number of open research challenges are discussed.

Categories and Subject Descriptors
D.2.5 [Software Engineering]: Testing and Debugging; C.2.1 [Computer Communication Networks]: Network Architecture and Design

Keywords
Test Scenario Explosion Problem, Automated Testing, P2P Networks

1. INTRODUCTION

Modern software development is characterised by a strong customer focus and electronic delivery mechanisms which make it very easy for customers to buy and install a vendor’s software. However, it also makes it very easy for customers to buy and install software from competing vendors and as such it is more important than ever for deployed software to be as correct and bug-free as possible.

Whilst certain types of testing can be done in the lab with a high degree of confidence that results will hold when the software is deployed in the wild, in reality software systems are subject to influence from whatever environment they end up being deployed in. Varying factors in customer environments can include operating systems, services packs, device drivers, network connectivity, resource usage by other software, and so on. Any variation or combination of these factors can lead to a situation where a system deviates from its expected behaviour. The problem is amplified even further on mobile devices whereby devices can move between different networks, interrupt applications for phone calls, have varying screen sizes, have user interference in the form of turning features on and off to preserve battery power, vendor-specific operating system code, and so on. A conservative calculation indicates that a software system can be subjected to tens of thousands of different scenarios. Even if one were to execute just one test case against each scenario, obtaining any form of realistic coverage is infeasible for even the most resource-rich organisations. We refer to this as the Test Scenario Explosion Problem.

Companies traditionally approach this problem by utilising dog-fooding and alpha-release techniques whereby software is pre-released to company employees and early-adopters respectively with the intent of gathering feedback prior to an official release. Cloud and grid infrastructures have also been proposed as solutions to improving the scalability of software testing [3, 4, 7]. However, we argue that cloud computing systems are too homogenous and are thus not representative of real-world usage scenarios. We also argue that the millions of interconnected devices around the world have the potential of being leveraged as a virtual test lab that is highly heterogenous and more likely to be representative of a company’s customer base. Furthermore, if one were to connect and utilise such devices using a peer-to-peer (P2P) overlay, such a system would be arguably cost-effective to use. This is mainly due to the fact that such networks can grow organically and function with very little or no central control [8].

In this paper we outline the main concepts behind this idea and identify a number of research problems which would be interesting to solve in this context. The reader is kindly asked to note that this is an exploratory position paper with strict length restrictions and as such some of the ideas will be not be extensively explained and are subject to tweaking.

2. CONCEPT OVERVIEW

In order to better motivate the concepts being presented here, consider the following example.

Example: Joe is a software engineer who has developed a mobile application for the Android platform. The application makes use of the device’s GPS locator whilst a user is driving so as to detect when a toll road is coming up. It then automatically pays the toll road fare for the user and coordinates with the traffic management systems so the user
Figure 1: The core usecase of the proposed network.

is allowed to drive through a special prepaid lane. Payment
and coordination is done over a data connection. Joe has
tested the system on three devices in his local area and is
happy with the functionality. He has also developed an au-
tomated test suite for the application. However, he is wary
about releasing the application to the wild where it will be
exposed to a large number of devices and toll roads which
are as yet untested.

This example illustrates the nature of the problem we are
attempting to address. Although Joe has thoroughly tested
his application, he cannot confidently predict how it will
function in the wild. GPS locators on devices may vary in
accuracy, data connections in certain areas may be flaky, the
device may switch networks whilst a payment is being made,
and so on.

Figure 1 demonstrates how a P2P network could be used
to crowd-source scenario testing. Once a developer has de-
veloped a suite of automated tests\(^1\), the core usecase involves
him being able to (1) advertise a request on the network
for these tests to be executed in different scenarios and (2)
eventually receiving test results with respect to those sce-
narios. A scenario is abstractly defined as a set of condi-
tions of interest under which the developer would like to
ensure that a system behaves correctly. Building on the
example above, consider a situation whereby the developer
wanted to ensure that the system was tested on four par-
ticular brands (Samsung, Motorola, LG, HTC), three API
Levels (15,16,17)\(^2\), and six network conditions (GPRS, Edge,
3G, 4G, WiFi, Flaky). As indicated in table 1, this would
result in seventy-two distinct scenarios. Furthermore, the
amount of instantiations of the scenarios would be consid-
erably higher so as to include a variety of features which
are not explicitly specified. For example, the availability of
multiple Samsung device models would result in the specifi-
cation (Samsung, 17, Flaky) being tested multiple times so
as to cover a multitude of Samsung’s models.

Once a request has been submitted to the network, it
should be propagated to peers who are able to execute the
test suite against one or more of the required scenarios. The
notion of a peer matching a request is made possible by peer
profiles.

2.1 Peer Profiles

\(^1\)The development and maintenance of automated test suites
has become common practice in the ICT industry.

\(^2\)An API Level is an integer value that uniquely identifies the
framework API revision offered by a version of the Android
platform.

Peers in the network maintain profiles that advertise in-
formation relevant to test scenarios which they are able to
satisfy. It is beyond the scope of this work to explicitly
impose a taxonomy on the content of profiles but we envis-
age profiles a being a set of key-value pairs. Consider the
following tables representing the profiles of two peers:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>brand</td>
<td>samsung</td>
</tr>
<tr>
<td>model</td>
<td>galaxy s3</td>
</tr>
<tr>
<td>network</td>
<td>wifi,3g</td>
</tr>
<tr>
<td>api-level</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>brand</td>
<td>samsung</td>
</tr>
<tr>
<td>model</td>
<td>galaxy s3</td>
</tr>
<tr>
<td>location</td>
<td>germany</td>
</tr>
</tbody>
</table>

Also consider the following requests for tests:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>brand</td>
<td>samsung</td>
</tr>
<tr>
<td>model</td>
<td>galaxy s3</td>
</tr>
</tbody>
</table>

Request 1 is seeking peers that can provide a Samsung
Galaxy S3 device and given the peer profiles in this ex-
ample, both Peer 1 and Peer 2 can service this request. Request
2 on the other hand, simply requires peers that are located
in Germany and does not specify any further constraints. In
this particular example Peer 2 would be able to handle the
request. Whilst the example is simplistic, one can envisage
the power that this mechanism can provide in terms of re-
cruiting test populations. Given a sufficiently large network,
one can construct a virtual lab on the fly whilst being as spe-
cific or vague with respect to test scenarios as the context
dictates.

2.2 Reformulating as a search problem

Given a sufficiently large network of peers, a means of
executing automated tests, and a mechanism for matching
profiles with requests, test scenario explosion can now be
reduced to a search problem. This can happen in two di-
rections. Firstly, from the point of view of a developer the
problem consists of finding peers in the network whose pro-
file matches a particular request. On the other hand, it can
also be seen as a problem whereby individual peers actively
search out requests which they can service.

Search functionality in P2P networks has received sub-
stantial research interest with various techniques proving
effective in file sharing applications. Amongst these, dis-
tributed hash tables (DHTs) proved to be particularly re-

<table>
<thead>
<tr>
<th>#</th>
<th>Brand</th>
<th>API Level</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Samsung</td>
<td>15</td>
<td>GPRS</td>
</tr>
<tr>
<td>2</td>
<td>Samsung</td>
<td>15</td>
<td>Edge</td>
</tr>
<tr>
<td>3</td>
<td>Samsung</td>
<td>15</td>
<td>3G</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>71</td>
<td>HTC</td>
<td>17</td>
<td>Wifi</td>
</tr>
<tr>
<td>72</td>
<td>HTC</td>
<td>17</td>
<td>Flaky</td>
</tr>
</tbody>
</table>

Table 1: A partial enumeration of scenarios resulting
from the example.
silient to the dynamic nature of such networks. As indicated by their name, DHTs provide the functionality for putting and getting values to/from a hash table that is distributed over multiple nodes using a key-value pair approach. They are highly scalable and robust to faults, network churn and malicious activity. Our current work is focused on using distributed hash tables in such a way that keys constitute elements of peer profiles whilst values would be lists of addresses for peers whose profiles match that of the keys.

Figure 2 illustrates a typical user journey in the proposed system. When a peer first joins the network, it provides a profile which is then processed and stored on the P2P network. This essentially means that the profile ends up being stored as key-value pairs on a distributed hash table. The mechanism by which this happens is non-trivial and is a work-in-progress. At some point, a developer makes a request to the network and provides a set of scenarios of interest. The request should result in the developer receiving a set of addresses of peers whose profiles match the scenarios of interest. The developer then processes the list, selects and appropriate subset of peers, contacts them and makes a job offer. If a peer accepts a job offer, the developer sends over the test artefacts, which typically consist of automated tests for developers are essentially downloading and executing code from a third party, be it in the form of a system under test or in the form of an automated test suite. The question arises as to how peers can be protected from such a mechanism being used maliciously. Furthermore, in the case of mobile devices, an application could potentially carry out actions which cost the device owner money. For example, an application could make calls, send messages and carry out actions which cost the device owner money. For example, an application could make calls, send messages and use data connections. This may be acceptable if the owner is reimbursed but protection mechanisms need to be in place. The developer also needs to place trust in peers who execute tests on his system. Firstly, one needs to ensure that the tests were actually executed in the required scenarios as opposed to the peer just returning faked results. Secondly, if the SUT is being transferred to the peer’s device, the developer should rest assured that the code cannot be stolen.

3.3 Combinatorial explosion of profiles

Given a large P2P network with little or no central control, the question arises as to how one can locate peers with particular profiles in a feasible amount of time. The traditional use of P2P networks for file sharing involves searching a network for a file with a particular name. In our case however, the search is more complicated. For example, if one were to search for peers whose profiles indicate that they have the Firefox web browser installed, a multitude of different profiles will be valid candidates. For example, two profiles could be used to search for peers whose profiles indicate that they have the Firefox web browser installed, but these profiles could be different. In our case however, the search is more complicated. For example, if one were to search for peers whose profiles indicate that they have the Firefox web browser installed, a multitude of different profiles will be valid candidates. For example, two profiles could both indicate that Firefox is installed but then have different operating systems. Since profiles will likely have dozens of key-value pairs, efficient algorithms for storage and lookup of profiles need to be designed.

3.4 Automated Negotiations

Consider a situation whereby a developer might request permission to execute test cases which send SMS messages. Such actions are likely to incur costs for the device owner and as such would require some form of negotiation whereby a peer would be willing to accept the job subject to certain...
conditions. For example, the peer might require that tests be executed between 2am and 4am because messages are not charged during that time. In other cases, the peer would require that a fee be paid for every message that is sent during testing. If a fee is to be charged then the fee needs to be negotiated. Following initial configuration by developers and device owners, negotiations would ideally completely automated. Logical frameworks for automated contract negotiation exist. Deontic logic [9] for example, provides syntax and semantics for reasoning about what every party in a contract is obliged to do, prohibited from doing or permitted to do. Its automation has been successfully applied in a number of case studies [2, 5] and applying it in our context would be interesting.

3.5 Peer Incentives

Participants in such a network would be expected to contribute their resources for the good of others. In contrast to file sharing networks whereby everyone can both contribute and benefit from the community, the network proposed here would only benefit developers. This makes it difficult to adapt existing work on incentives in P2P systems to this context [1, 6]. Consequently, an incentive system would need to be devised such that individualise would be compelled to participate. Our initial ideas in this regard revolve around a monetary scheme whereby peers could receive a small token payment for each executed test. Whilst a token payment of $0.01 may not sound like much, it could easily add up to an attractive sum considering hundreds of tests could execute in a short amount of time and the device owner does not need to do any work. This would also be attractive to developers who would not only have access to an inexpensive testing platform, but also have fine-grained control over the amount of money they spend on a particular test run.

Since global financial regulations can be tedious, non-monetary incentives should also be considered. Incentives like space on an online cloud storage service, reputation-based incentives, access to the developer’s products, and so on, could still be effective at attracting participants.

4. CONCLUSIONS AND FUTURE WORK

In order to address the test scenario explosion problem, we have proposed the construction of a P2P network that allows on-the-fly creation of large-scale virtual test labs with representative characteristics of a company’s customer base. We are currently in the advanced stages of creating a prototype system which can be used as a basis for experiments and improvements with the intent of addressing the research challenges identified in this paper.

We believe that the successful deployment of such a network would have a considerable impact on the software development community and if successful, could further strengthen the position of individual developers and small companies who have been empowered with app store based delivery channels that appeared in recent years.

5. REFERENCES