УДК 004.054

ИССЛЕДОВАНИЕ РАБОТЫ СЕТЕВОГО СТЕКА СОВРЕМЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМ ПРИ ОБРАБОТКЕ ИНФОРМАЦИИ, ПОСТУПАЮЩЕЙ ИЗ СЕТИ

Бородин А.А.

ФГБОУ ВПО «Московский государственный университет леса», Мытищи, Россия (141005, Московская обл., г.Мытищи-5, ул. 1-я Институтская, д. 1), e-mail: AntonioBorodin@gmail.com

Современная цивилизация не может эффективно решать свои задачи без информационных систем. Обеспечение надежности функционирования этих систем является важной и актуальной проблемой. Одним из центральных способов ее решения является тестирование. В настоящий момент существуют различные виды тестирования. Для информационных систем глобальной сети наиболее значимым является нагрузочное тестирование. С его помощью удается измерить характеристики информационной системы и исследовать ее поведения при эксплуатационных условиях. На улучшение данного процесса направлены усилия многих исследователей. Однако исследования вопросов стадии запуска тестов и сбора результатов привлекает значительно меньше внимания. В ходе этой стадии осуществляется генерация нагрузки на информационную систему и собираются метрики производительности. Сбор метрик производительности проводится на основе анализа ответов, поступающих от исследуемой системы. В ходе этого каждый ответ должен пройти через сетевой стек операционной системы. В данной статье представлены результаты экспериментов по измерению времени, необходимого операционной системе для обработки каждого ответа. Полученные результаты демонстрируют, что среднее время обработки снижается по мере роста интенсивности поступления ответов. Эксперименты также показали, что время обработки входящих данных превышает время исходящих. Это обстоятельство очень важно учитывать при проведении тестирования на практике.

Ключевые слова: время обработки пакета операционной системой, нагрузочное тестирование, сетевой стек.

THE RESEARCH OF OPERATION SYSTEM NETWORK STACK DURING PROCESSING OF DATA ARRIVING FROM NETWORK

Borodin A.A.

Moscow State Forest University, Mytischi, Russia (1, 1st Institutskaya street, Mytischi, Moscow region, Russia, 141005), e-mail: AntonioBorodin@gmail.com

Modern civilization can't solve problems effectively without information systems. Providing reliability of such systems is very important and urgent task. One of the central ways to solve it is a testing. Nowadays there are different types of testing. For information system of global network the most important testing type is a load testing. It helps to estimate characteristics of information system and to research behavior of that system under the operation conditions. The improvement of load testingis focused the efforts of many scientists. However the research of testing launch and collecting the results are attracting much less attention. Onthis stage, the loading to information system is being generated and performance metrics are collected. Collecting of performance metrics is based on response analysis that comes from system are being tested. Every response must pass through the network stack of an operation system. This article is representing the results of experiments on measurements of network stack processing time. The obtained results demonstrate the average processing time is decreasing with increasing of responses intensity. Additionally our results show that processing time of incoming data is exceed time of out coming data. This circumstance must be considered during testing on a practice.

Keywords: packet processing time, load testing, network stack.

Постоянное развитие мыслительной деятельности является важнейшим условием адаптации человека к меняющимся условиям существования. Благодаря этой способности он не только смог выжить, но и стал оказывать решающее влияние на окружающий мир. Человек создал инструменты, которые облегчали не только его физические, но и интеллектуальные усилия. Настоящую революцию пережила человеческая цивилизация

после создания компьютера – инструмента значительно увеличивающего мыслительные возможности. Появилась возможность обрабатывать и хранить колоссальный объем информации, облегчая решение многих задач. Цивилизация в ее нынешнем виде не могла бы существовать без постоянного функционирования компьютеров и их надежность является важнейшим фактором. Значение этих мыслительных инструментов трудно переоценить, так как от их успешного функционирования зависит само существование нашей цивилизации.

В любом инструменте могут возникать различные поломки, недочеты, и чем он сложнее, тем труднее их обнаружить. Поэтому поиску ошибок и проблем в компьютерных системах отводится исключительное место. Важнейшим средством диагностики информационных систем является тестирование. В настоящий момент существуют различные виды тестирования. Каждое из них направлено на проверку различных аспектов функционирования исследуемой системы. В данной статье рассматриваются вопросы нагрузочного тестирования. Оно используется для измерения характеристик и проверки надежности информационных систем, которые взаимодействуют с большим количеством пользователей. Нагрузочное тестирование является сложным и многоплановым процессом, состоящим из множества шагов. Оно является предметом внимания многих исследователей. Условно их усилия можно разбить на 2 группы:

- улучшение процесса создания тестов и подготовки к тестированию [2-5,7-9,11];
- улучшение процесса анализа результатов [6,12].

Однако изучение литературы показало, что процессу запуска тестов и сбора результатов уделяется недостаточное внимание. На восполнение этого недостатка и направлена наша работа.

В ходе стадии запуска тестов исследуемая система подвергается нагрузке, то есть на нее отправляются тестовые запросы с большой интенсивностью. Она отвечает на каждый запрос и эти ответы анализируются. В предыдущей работе [1] нами были описаны результаты экспериментов, которые мы проводили с целью измерения времени, необходимого операционной системе для подготовки и отправки одного запроса, сформированного прикладным приложением. Проведение нагрузочного тестирования требует не только генерации запроса, но и приема ответов от информационной системы для их анализа. В настоящее время принято считать, что время обработки исходящих и входящих данных примерно одинаково. С целью проверки этого предположения мы провели ряд экспериментов для измерения времени обработки данных, поступающих из сети, операционной системой. В настоящей статье представлены результаты этих экспериментов.

Для достижения указанной цели необходимо узнать время перед поступлением сообщения в сетевой стек и время после него. Исходя из разницы, можно вычислить время обработки. Решение поставленных задач было осуществлено с помощью разработки 3 экспериментальных программ, а также приложения tcpdump.

Программа tcpdump является перехватчиком пакетов, функционирующим на канальном уровне сетевого стека [10]. Учитывая параметры перехвата, данная программа осуществляет сбор исходящих и входящих пакетов. В рамках нашей задачи tcpdump определяла момент времени перед поступлением сообщения в сетевой стек. Разработанные нами экспериментальные программы выполняли другие функции:

- программа генератор сообщений отправляла множество пронумерованных сообщений на исследуемую систему;
- программа сервер формировала файл, содержащий номер сообщения и время его получения (программа регистрировала момент времени после обработки сообщения сетевым стеком);
- программа анализатор на основе результатов работы tcpdump и программы сервера определяла время обработки каждого сообщения.

Эксперименты проводились на базе компьютеров с адекватным программным и аппаратным обеспечением. В рамках нашей работы эти компьютеры назывались экспериментальными платформами. Программа генератор сообщений, запущенная на отдельном узле, отправляла данные на исследуемую платформу через сеть. В ходе этого она формировала сообщения с плавным повышением интенсивности генерации. Приложение генератор было спроектировано таким образом, чтобы достичь предельной интенсивности используемого узла для однопоточного приложения. Каждое сгенерированное сообщение представляет собой UDP пакет с порядковым номером.

В ходе экспериментов применялись узлы со следующими характеристиками:

- экспериментальная платформа № 1, компьютер IntelCorei7 920 2.67 ГГц, DDR3-1066 9ГБ, CentOS 6.4;
- экспериментальная платформа № 2, ноутбук SamsungR580-JS03 IntelCorei5 430M 2.26 ГГц, DDR3-1066 3 ГБ, Ubuntu 12.10;
- экспериментальная платформа № 3, ноутбук MacBookAirIntelCorei7 3667U 2 ГГЦ, DDR3L-1600МГц 8 ГБ, OS X 10.8.5.

В ходе экспериментов было собрано около миллиона измерений на каждой из платформ. Для большей достоверности результатов эксперимент повторялся 10 раз. Обобщенные результаты представлены в таблице 1. Детализированные результаты отражены в таблице 2. Расчет погрешности осуществлялся с помощью следующей формулы:

$$S_x = \frac{S}{\sqrt{n}} = \sqrt{\sum_{i=0}^{n} \frac{(t_i - \bar{t})^2}{n(n-1)}}$$

Для расчета погрешности среднего между испытаниями применялся коэффициент Стьюдента равный 2,262 (число измерений 10, надежность 0,95).

Таблица 1

Обобщенные результаты измерения времени обработки входящих данных сетевым стеком

Платформа	Кол-во испытаний	$t_{\text{среднее}}$, мкс	t_{min}, мкс	t_{max}, мкс	Погрешность среднего между испытаниями, мкс
ЭП № 1	10	24,13	2	35 431	± 2,55
ЭП № 2		52,43	1	466 721	± 26,841
ЭП № 3		69,96	7	20 079	± 1,3499

Таблица 2

Детализированные результаты измерения времени обработки входящих данных сетевым стеком

Платформа	Номер испытания	Обработано пакетов	$t_{\text{среднее}}$, мкс	t_{min}, мкс	t_{max}, мкс	Погрешность измерений среднего, мкс
ЭП № 1	1	1 557 825	25,72	2	3637	± 0,0127
	2	1 569 322	20,39	2	35431	± 0,0863
	3	1 575 246	25,5	2	608	± 0,0079
	4	1 568 375	19,8	2	1391	± 0,0073
	5	1 579 085	24,45	2	1981	± 0,0078
	6	1 551 849	23,39	2	1348	± 0,0077
	7	1 582 120	24,38	2	929	± 0,0075
	8	1 577 589	19,81	2	1651	± 0,0064
	9	1 566 528	26,52	2	1585	± 0,0096
	10	1 572 961	31,37	2	32431	± 0,3388
ЭП № 2	1	785 535	38,75	2	2794	± 0,0391
	2	796 895	40,62	2	3820	± 0,058
	3	800 569	40,70	2	3707	± 0,044
	4	800 825	159,28	1	466721	± 10,1942
	5	811 398	40,1	2	4435	± 0,0458
	6	804 588	41,11	2	4290	± 0,0593
	7	802 063	40,55	1	4576	± 0,0573
	8	799 033	42,47	1	49059	± 0,3792
	9	811 631	40,35	2	4790	± 0,0578
	10	801 517	40,33	1	3409	± 0,0441
ЭП № 3	1	815 149	65,14	8	19935	± 0,3027
	2	813 021	68,81	7	14929	± 0,1608
	3	815 307	70,43	8	18452	± 0,2188
	4	809 208	71,03	8	20079	± 0,2868
	5	807 930	70,32	7	17076	± 0,2297
	6	810 694	72,03	7	19816	± 0,3086
	7	810 209	69,91	7	19554	± 0,2431
	8	814 732	70,11	8	16860	± 0,193
	9	811 285	71,05	8	15180	± 0,2206
	10	811 599	70,73	8	18993	± 0,2337

Графики 1–3 демонстрируют изменение интенсивности и времени обработки сообщения в ходе экспериментов для каждой платформы.

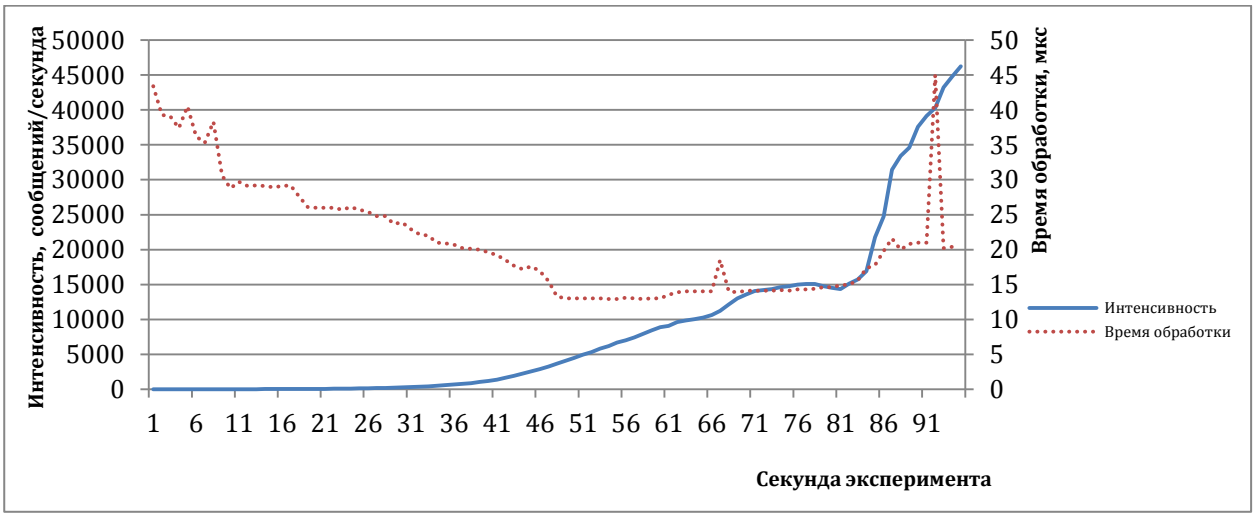

График 1. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 1

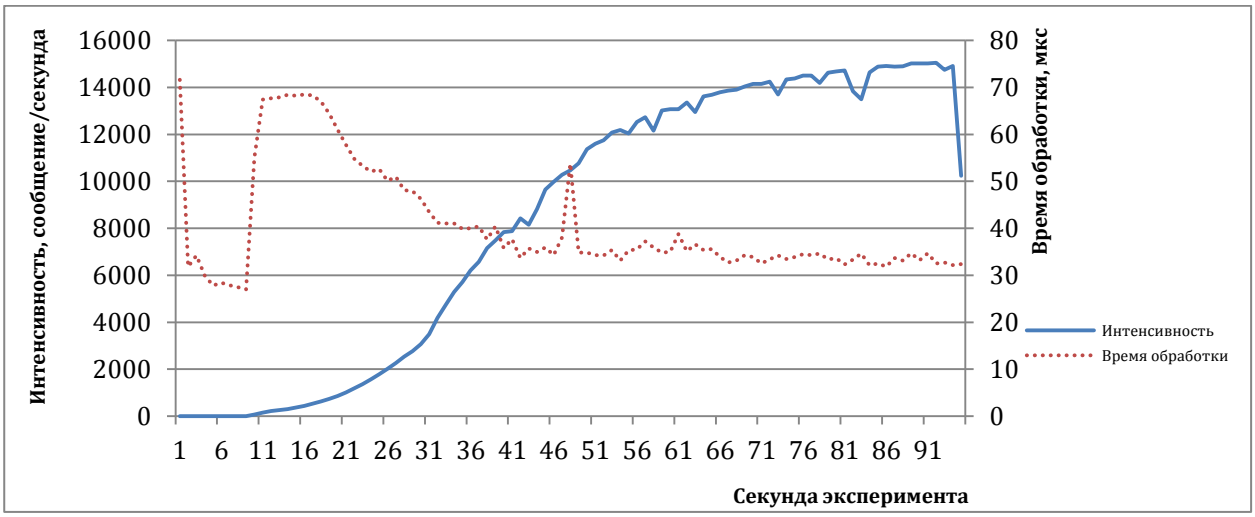

График 2. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 2

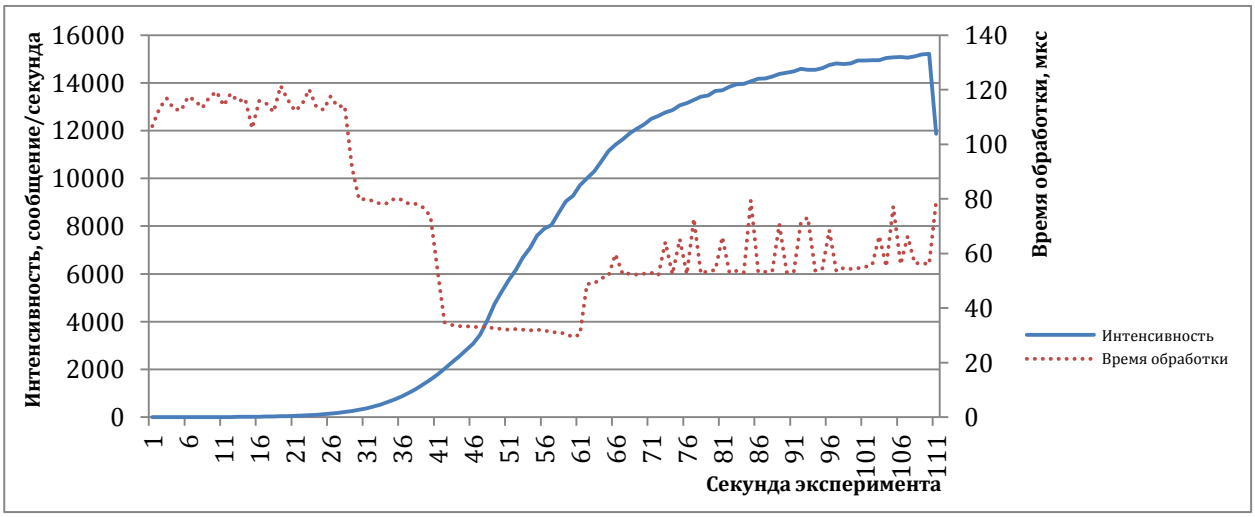

График 3. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 3

Из представленных графиков следует, что с ростом интенсивности время обработки сообщения уменьшается. Точно такая же тенденция наблюдается при обработке исходящих данных сетевым стеком [1]. Для большей уверенности в полученных результатах экспериментальная программа генератор сообщений была модифицирована. Модификация заключалась в том, что интенсивность, достигнув пика, начинала свое плавное уменьшение. Для сокращения объема обрабатываемых данных предельная интенсивность была ограничена 9000 сообщений в секунду. Результаты проведенного эксперимента с модифицированной программой представлены на графиках 4–6.

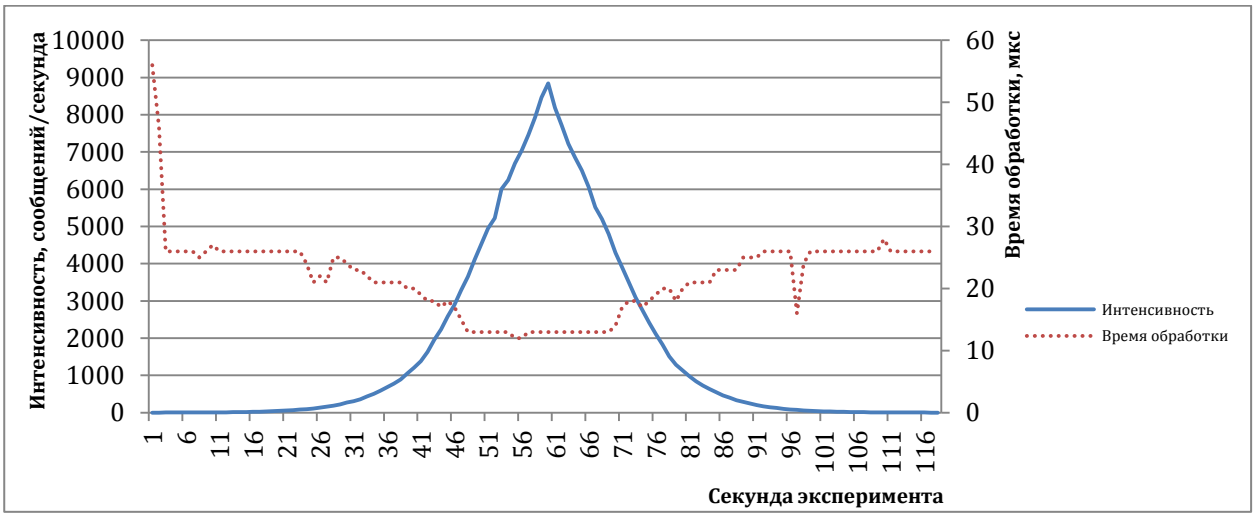

График 4. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 1 после модификации экспериментальной программы

График 5. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 2 после модификации экспериментальной программы

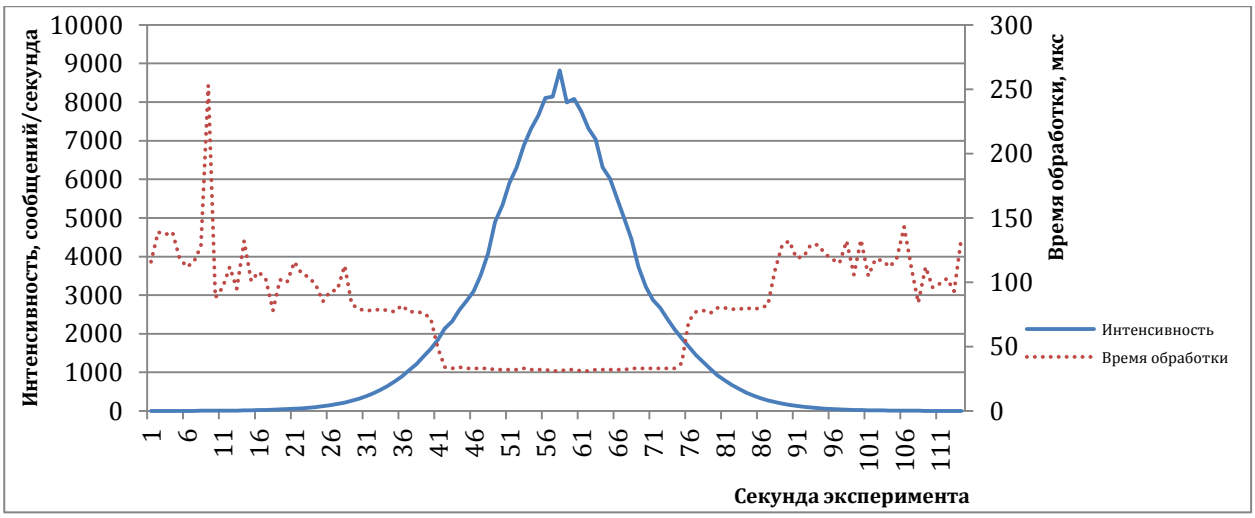

График 6. Изменение интенсивности и времени обработки входящих данных сетевым стеком ЭП 3 после модификации экспериментальной программы

Результаты, представленные на графиках 4-6, подтверждают предположение о том, что с ростом интенсивности уменьшается время обработки. График изменения среднего времени обработки симметричен относительно пиковой интенсивности. С ростом интенсивности среднее время обработки снижается до некоторого предела. После достижения пикового значения интенсивности и начала ее снижения среднее время обработки начинало расти.

Полученные результаты отражают количество времени, необходимого сетевому стеку операционной системы, для обработки поступающего из сети сообщения. Проведенные эксперименты также показали, что современные операционные системы оптимизированы для работы с данными, поступающими с большой интенсивностью. Вместе с тем время обработки одиночных сообщений может достигать, в некоторых случаях, высоких значений. Кроме того экспериментальные результаты демонстрируют, что время обработки входящих сообщений превышает время исходящих. Так, например, рост времени обработки входящих сообщений по сравнению с исходящими [1] для ЭП1 составил 166 %, для ЭП2 – 406 %, а для ЭП3 – 16 %. Очевидно данная величина напрямую зависит от особенностей операционной системы. Среди представленных платформ лишь на ЭП3 время обработки исходящих и входящих сообщений различается незначительно. Таким образом, проведенная работа позволяет говорить об ошибочности существующего предположения о примерном равенстве времени обработки входящих и исходящих данных. Полученные результаты демонстрируют время, необходимое сетевому стеку операционной системы для обработки данных, поступающих из сети. В нашем исследовании мы разрабатываем новый подход проведения нагрузочного тестирования, который основан на применении аппаратного блока для создания нагрузки и сбора результатов. При разработке этого устройства были учтены

недостатки применяемых в настоящее время систем, при этом скорость обработки входящих и исходящих данных оказалась значительно выше.

Список литературы

1. Бородин А.А. Исследование нагрузочных способностей компьютерных систем // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2014. № 3 (198). С. 19-27.
2. Ермыкин А.А. Разработка метода построения комплекса нагрузочного тестирования распределенной информационной системы: дис. … канд. тех. наук: 05.13.13. СПб., 2005. 147 л.
3. Силаков Д.В. Автоматизация тестирования web-приложений, основанных на скриптовых языках // Труды ИСП РАН. 2008. № 2. С.159-178.
4. Сортов А., Хорошилов А. Функциональное тестирование Web-приложений на основе технологии UniTesK // Труды Института системного программирования РАН. № 8. 2004. С. 77-97.
5. Яковенко П. Н., Сапожников А. В. Инфраструктура тестирования веб-сервисов на базе технологии ttcn-3 и платформы.net // Труды ИСП РАН. 2009. Т. 17. С.63-74.
6. Haroon Malik, A Methodology to Support Load Test Analysis // ICSE '10 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. Vol. 2, pp. 421-424.
7. Kevin Morrison, Hisham M. Haddad Converting users to testers: an alternative approach to load test script creation, parameterization and data correlation // Journal of Computing Sciences in Colleges. Vol. 28. Issue 2, December 2012, pp. 188-196.
8. Pingyu Zhang, Sebastian Elbaum, Matthew B. Dwyer Automatic Generation of Load Tests // ASE '11 Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, pp. 43-52.
9. Pingyu Zhang, Sebastian Elbaum, Matthew B. Dwyer Compositional load test generation for software pipelines // ISSTA 2012 Proceedings of the 2012 International Symposium on Software Testing and Analysis, pp. 89-99.
10. TCPdump [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Tcpdump (дата обращения 29.08.2014).
11. YuhongCai, John Grundy, John Hosking Synthesizing client load models for performance engineering via web crawling // ASE '07 Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, pp. 353-362.

12. Zhen Ming Jiang, Automated analysis of load testing results // ISSTA '10 Proceedings of the 19th international symposium on Software testing and analysis, pp. 143-146.

Рецензенты:

Корольков А.В., д.ф.-м.н., профессор, декан ФЭСТ, ФГБОУ ВПО «Московский государственный университет леса», г. Мытищи;

Финогеев А.Г., д.т.н., профессор кафедры «Системы автоматизации проектирования», ФГБОУ ВПО «Пензенский государственный университет», г. Пенза.