АВТОМАТИЗАЦИЯ КОНТРОЛЯ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

И.В. Кузьмина, С.А. Минеев, О.В. Семенова

Нижегородский госуниверситет

Стремительный прогресс в сфере информационных технологий и радиотехнике позволяет создавать все более сложные многокомпонентные радиотехнические системы, в состав которых входят следующие элементы:

а) аналоговая аппаратура (усиление/ослабление, фильтрация радиочастотных (РЧ) сигналов и др.);
б) цифровые системы обработки сигналов (оцифровка, фильтрация, спектрально-временной анализ, корреляционная обработка);
в) вычислительные и управляющие узлы (управление аппаратурой, форматирование результатов работы, сохранение результатов в базах данных).

Быстрое увеличение сложности и размеров современных аппаратно-программных комплексов при одновременном росте ответственности выполняемых функций, резко повысило требования со стороны заказчиков и пользователей к их надежности, качеству и безопасности применения. Успешное функционирование таких систем зависит от многих факторов, и одним из основных является корректность работы программного обеспечения (ПО).

В настоящее время существует множество международных стандартов, регламентирующих процессы жизненного цикла программных средств. Их применение может служить основой для создания систем обеспечения качества программных продуктов. В стандартах отмечается, что оценка качества осуществляется на всех этапах – при планировании показателей качества, его контроле на отдельных стадиях разработки, в процессе производства, при проверке эффективности модификации на этапе сопровождения. Очевидно, что автоматизация процесса контроля качества ПО является актуальной задачей и позволит улучшить качество разрабатываемых продуктов и облегчить сам процесс контроля.

Основными составляющими в определении качества ПО является «соответствие требованиям» и «пригодность к использованию». Количественную оценку качества программных продуктов традиционно получают с помощью расчета метрик [1]. В соответствии со стандартом ISO 14598 рекомендуется следующая общая схема процессов оценки характеристик качества программ:

- установка исходных требований для оценки – идентификация типа метрик, выделение адекватных показателей и требуемых значений атрибутов качества;
- селекция метрик качества, установление уровней приоритета метрик;
- планирование и проектирование процессов оценки характеристик и атрибутов качества в жизненном цикле программного средства;

- выполнение измерений для оценки, сравнение результатов с критериями и требованиями, обобщение и оценка результатов.

Сложность программно-аппаратных комплексов, разрабатываемых в лаборатории «Моделирования физических процессов и цифровой обработки данных» (НИФТИ ННГУ) предъявляет все более высокие требования к качеству и надежности программного обеспечения этих систем. Чтобы удовлетворить этим требованиям, необходимо как организовать ряд мероприятий направленных на улучшение процесса разработки ПО, так и оценить эффективность их работы. К факторам, напрямую влияющим на качество процесса разработки ПО можно отнести:

- использование систем контроля версий при разработке ПО;
- организация процесса управления требованиями;
- наличие процесса управления изменениями;
- оптимальный тип интеграции программных компонентов в единую систему;
- использование сборочного сервера и ежедневной сборки ПО;
- модульное тестирование на этапе компиляции;
- автоматический расчет и анализ метрик на этапе сборки, характеризующих качество исходного кода.

Для каждого созданного комплекса проводился статистический анализ, включающий расчет количества строк кода (LOC – характеристик), хранящегося в системах контроля версий, таких как SourceSafe («Microsoft») и ClearCase («IBM Rational»), а так же количества ошибок, возникающих на протяжении всего жизненного цикла продукта, фиксируемых в системах управления изменениями, Gemini («CounterSoft» и ClearQuest («IBM Rational»).

Количество ошибок, содержащихся в ПО, как правило, зависит от объема исходного кода, и этот факт очень часто используется для расчета тех или иных метрик, характеризующих как качество программного продукта, так и процесс его производства. Накопленные в лаборатории статистические данные позволяют проводить сравнительный анализ эффективности мероприятий по улучшению качества ПО. Показателем эффективности является аддитивная вероятностная мера:

$$k = -\log \frac{N_{err}}{LOC}, \tag{1}$$

где N_{err} – интегральное число ошибок по проекту; LOC – количество строк кода по проекту.

В работе проведены исследования данного показателя для нескольких сложных программно-аппаратных комплексов, разработанных с использованием различных инструментов в разные периоды времени до внедрения мероприятий, направленных на улучшение качества ПО и системы качества в целом, и после.

[1] Орлов С.А. Технологии разработки программного обеспечения. СПб.: Питер, 2002. С.464.

О ПРИВЕДЕНИИ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ШИФРАТОРА К ФОРМЕ, ДОПУСКАЮЩЕЙ ОБРАЩЕНИЕ

А.А. Горбунов

Нижегородский госуниверситет

Представление математических моделей (ММ) основных блоков криптосистем (КС) (шифратора, дешифратора, канала связи) в форме дискретных конечных автоматов является полезным при решении ряда задач, тесно связанных со структурной и параметрической идентификацией ММ КС. Каждый из блоков КС как источник данных описывается в этом случае моделью типа «черный ящик» в форме синхронного автомата Хаффмана – Глушкова и описывается следующей системой уравнений [1]:

$$\begin{cases} \boldsymbol{x}(t+1) = \gamma\big(\boldsymbol{x}(t), \boldsymbol{u}(t); \boldsymbol{p}\big) \\ \boldsymbol{y}(t) = \lambda\big(\boldsymbol{x}(t), \boldsymbol{u}(t); \boldsymbol{p}\big) \\ \boldsymbol{x}(0) = \boldsymbol{x}_н \end{cases} \quad (1)$$

Здесь:
$t \in [0, 1, \ldots, M–1]$ – дискретное время;
$\boldsymbol{u}(t) = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{M-1}\}$ – входной векторный сигнал автомата;
$\boldsymbol{y}(t) = \{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{M-1}\}$ – выходной векторный сигнал автомата;
$\boldsymbol{x}(t) = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M-1}\}$ – внутреннее n-мерное состояние автомата;
$\boldsymbol{x}_н$ – вектор начального состояния автомата;
$\boldsymbol{p}$ – вектор свободных (рабочих) параметров.

В то же время, актуальной остается задача построения модели дешифратора как восстанавливающего автомата по отношению к ММ шифратора, представленной в том же самом формализме, что и имеющаяся модель шифратора. Подобный подход позволяет расширять математическое описание модели на всю криптосистему в целом, что дает возможность подходить к исследованию, например, стойкости всех её элементов с единых методических позиций.

Для случая, когда в соответствующей алгебраической структуре (АС) функции динамики γ и наблюдения λ являются линейными по отношению к переменным $\boldsymbol{x}(t)$ и $\boldsymbol{u}(t)$, синтез ММ дешифратора по модели шифратора может быть осуществлен при помощи формул, полученных в [2]. В ряде более общих случаев, при решении той же самой задачи возможен подход, основанный на выборе АС преобразований (1) таким образом, чтобы имелась возможность получать обратные им преобразования.

Если в некоторой АС имеется возможность представить ММ шифратора в виде:

$$\begin{cases} \boldsymbol{x}(t+1) = \gamma_1\big(\boldsymbol{x}(t); \boldsymbol{p}\big) \\ \boldsymbol{y}(t) = \lambda_1\big(\boldsymbol{x}(t); \boldsymbol{p}\big) \langle\circ\rangle\ \lambda_2\big(\boldsymbol{u}(t); \boldsymbol{p}\big) \\ \boldsymbol{x}(0) = \boldsymbol{x}_н \end{cases}, \quad (2)$$

где $\langle\circ\rangle$ – соответствующая данной АС операция «сложения» и функция λ_2 являются обратимыми, то соответствующая модель дешифратора – восстанавливающего автомата может быть записана как:

$$\begin{cases} \boldsymbol{x}(t+1) = \gamma_1\big(\boldsymbol{x}(t);\boldsymbol{p}\big) \\ \boldsymbol{u}(t) = \lambda_2{}^{-1}\Big(\boldsymbol{y}(t)\ \langle\circ\rangle^{-1}\ \lambda_1\big(\boldsymbol{x}(t);\boldsymbol{p}\big);\boldsymbol{p}\Big) \\ \boldsymbol{x}(0) = \boldsymbol{x}_н \end{cases}, \tag{3}$$

где $\langle\circ\rangle^{-1}$ – обратная введённой операции $\langle\circ\rangle$ операция «вычитания» в той же АС.

Пример использования подобного приема при синтезе обратимой модели шифратора на основе таблицы замены для q-значных алфавитов показан в [2] и представляет собой введение в качестве операции сложения нелинейной индексной операции над вектором и числом – операции *сложения по индексу:*

$$\boldsymbol{z} \oplus_i \boldsymbol{u} = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_q \end{bmatrix} \oplus_i \boldsymbol{u} = \begin{cases} z_1, & \text{если } u = 1 \\ z_2, & \text{если } u = 2 \\ & \dots \\ z_q, & \text{если } u = q \end{cases}. \tag{4}$$

Операция, обратная к ней, названная в [2] *извлечением индекса* введена как:

$$\boldsymbol{z} \oplus_i^{-1} \boldsymbol{u} = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_q \end{bmatrix} \oplus_i^{-1} \boldsymbol{u} = \begin{cases} 1, & \text{если } u = z_1 \\ 2, & \text{если } u = z_2 \\ & \dots \\ q, & \text{если } u = z_q \end{cases}. \tag{5}$$

Другой способ получения обратимой модели шифратора предложен в [3] и связан с построением эквивалентной ММ линейного цифрового автомата по имеющейся нелинейной модели автомата. Данный подход, в отличие от рассмотренного выше, не предполагает перехода к иным АС, в связи с чем, подобный вид линеаризации модели автомата не всегда представляется возможным.

[1] Кирьянов, К.Г., Горбунов, А.А., Туренко, Д.Л. // Труды VI Международной конференции «Идентификация систем и задачи управления SICPRO’07». 2007. С. 178.

[2] Горбунов А.А., Кирьянов К.Г. // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия «Радиофизика». Вып. 1(2). Н. Новгород: ННГУ. 2004. С. 24.

[3] Гилл А. Линейные последовательностные машины: Перев. с англ. М.: Наука. 1974.

ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА ПОСТРОЕНИЯ ГРАФА ПОТОКОВ ДАННЫХ КОДА X86

А.В. Корюкалов, А.Н. Малышев, Л.Ю. Ротков

Нижегородский госуниверситет

В настоящее время задача автоматизированного детектирования вредоносных программ решается с помощью подхода, основанного на поиске сигнатур в исполняемом коде. Данный подход имеет ряд недостатков [1]. Другие подходы к автоматизированному исследованию исполняемого кода основаны на его приведении к определенному представлению в виде графа потока управления или последовательности системных вызовов кода и на поиске в этом представлении шаблонов из библиотеки [2–4].

Для решения задачи автоматизированного детектирования вредоносных программ предлагается представление исполняемого кода в виде графа потоков данных.

В качестве вершин графа потоков данных выступают инструкции процессора. Наличие дуги между двумя инструкциями свидетельствует о возможном использовании значения объекта данных (регистра процессора, ячейки памяти), определяемого в первой инструкции, второй инструкцией. Каждая дуга также характеризуется объектом данных, который связывает инструкции. Граф потоков данных исполняемого кода является ориентированным.

Разработан алгоритм построения графа потоков управления и реализовано соответствующее программное средство. В качестве исходных данных средство принимает дизассемблированный код исследуемой программы. В качестве выходных результатов средство выдает набор вершин и набор дуг графа потока управления.

Для проверки корректности построения графа применялся набор специальных тестов, включающих в себя изменения в ассемблерном коде, подающегося на вход средства (изменение регистра-операнда, добавление/удаление инструкции) и анализ графа – результата работы средства.

После проведения данного тестирования выявлены некоторые недостатки алгоритма построения графа потоков данных.

Во-первых, алгоритм не распознаёт дуги между инструкциями, работающими с одним участком памяти, заданным с использованием косвенной адресации (с помощью комбинации регистров процессора), если при адресации используются разные регистры.

Во-вторых, алгоритм не учитывает передачу параметров через стек.

Таким образом, для увеличения точности вычисления графа потоков данных необходима доработка алгоритма и программного средства, включающая дополнительный анализ ассемблерного кода.

Дальнейшее исследование должно быть направлено на определение структуры шаблона, принципов составления библиотеки шаблонов и критериев поиска шаблонов в графе.

[1] Корюкалов А.В., Ротков Л.Ю. //Труды 10-й Научной конференции по радиофизике, ННГУ, 7 мая 2006 г. Н. Новгород.
[2] Bergeron J., Debbabi M., Desharnais J., Erhioui M.M., Lavoie Y., Tawbi N. // Proc. International Symposium on Requirements Engineering for Information Security, 2001.
[3] Christodorescu M., Jha S., Seshia S.A., Song D., Bryant R.E. // Proc. 2005 IEEE Symposium on Security and Privacy. 2005. P.32.
[4] Christodrescu M., Jha S. //Proc. 12th USENIX Security Symposium. 2003. P.169.
[5] Rozinov K. Efficient Static Analysis of Executables for Detecting Malicious Behaviors. 2005.

АНАЛИЗ ПОТОКОВ ДАННЫХ В КОДЕ АРХИТЕКТУРЫ X86

А.В. Корюкалов, Л.Ю. Ротков, В.А. Фролов

Нижегородский госуниверситет

В настоящее время существует ряд исследований и методов, направленных на автоматизацию процесса исследования программного обеспечения, представленного в виде исполняемых модулей, на предмет наличия в нем вредоносных функций [1,2]. Такие исследования основаны на анализе последовательностей вызовов функций операционной системы. При этом учитываются названия вызовов, но не учитываются их аргументы. Значительно повысить эффективность исследований может анализ аргументов вызовов.

В анализе исполняемого кода в качестве вспомогательной структуры для его промежуточного представления используется граф потока управления, изображающий возможные пути выполнения программы. Такой граф делает анализ более эффективным и наглядно изображает связи между различными частями программы. Достаточно часто для затруднения анализа авторы вредоносного программного обеспечения используют косвенную адресацию в инструкциях, нарушающих последовательное выполнение программы [3]. В этих случаях для получения точного графа потока управления необходимо вычислять аргументы таких инструкций.

Таким образом, существует актуальная задача вычисления значений объектов данных (регистров процессора и участков памяти) в определенных точках кода. В данной работе рассматривается метод таких вычислений, основанный на статическом анализе исполняемого кода архитектуры X86.

Предлагаемый метод основывается на построении графа потоков данных в коде. Вершинами графа являются инструкции ассемблерного кода, а дуга между двумя инструкциями свидетельствует об использовании во второй инструкции объекта данных, определенного в первой инструкции. Зависимость по данным между двумя инструкциями может существовать только тогда, когда существует путь выполнения программы, связывающий эти инструкции. Следовательно, для построения графа потоков данных предварительно необходимо строить граф потока управления программы. Вершинами данного графа являются базовые блоки (по-

следовательности инструкций, такие, что при получении управления первой инструкцией из последовательности, за этот же проход выполняются все инструкции блока), а дуги свидетельствуют о возможности передачи управления между базовыми блоками кода.

Поскольку в общем случае в определенную точку программы можно попасть разными способами, то в интересующей точке программы исследуемый объект данных будет характеризоваться набором значений.

В работе разработаны и реализованы алгоритм вычисления графа потока управления по исполняемому коду, алгоритм вычисления графа потоков данных, алгоритм вычисления набора значений объекта данных в определенной точке кода.

Тестирование разработанных программных средств проводилось на дизассемблированном исполняемом модуле, содержащем реализацию алгоритма сортировки массива целых чисел. Тесты показали, что в разработанных алгоритмах недостаточно анализируются возможные пути выполнения исследуемой программы. Так набор значений объекта данных, являющегося счетчиком цикла, был вычислен как {0, 1}, хотя на самом деле мог принимать значения {0, 31}. Такой результат обусловлен тем, что разработанный алгоритм не анализирует условия ветвлений в коде.

Исследования в данной области будут продолжены и направлены в первую очередь на анализ условий ветвлений в коде.

[1] Bergeron J., Debbabi M., Desharnais J., Erhioui M.M., Lavoie Y., Tawbi N. // Proc. International Symposium on Requirements Engineering for Information Security. 2001.
[2] Rozinov K. Efficient Static Analysis of Executables for Detecting Malicious Behaviors. 2005.
[3] Balakrishnan G., Reps T. // Proc. International Conference on Compiler Construction (CC'04). 2004. V. 2985. P. 5.

АНАЛИЗ ПОСЛЕДОВАТЕЛЬНОСТЕЙ API-ВЫЗОВОВ ВО ВРЕДОНОСНОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

А.В. Корюкалов, А.Н. Малышев, Л.Ю. Ротков

Нижегородский госуниверситет

В настоящее время актуальной является задача детектирования вредоносного программного обеспечения. Для борьбы с такими программами используются антивирусные средства. Принцип действия подавляющего большинства антивирусных средств основан на сигнатурном анализе, основные преимущества которого заключаются в достаточно высокой скорости работы и высокой точности обнаружения вредоносных программ. Основным недостатком такого анализа является то, что с помощью него принципиально невозможно защититься от ранее неизвестной вредоносной программы (если на программу никто не жаловался, соответственно,

отсутствует и ее сигнатура в антивирусной базе). Кроме того, авторы современных вредоносных программ используют различные метаморфные преобразования, что влечет за собой со стороны поставщиков антивирусных средств необходимость анализировать каждую версию вредоносной программы и синтезировать для нее свою сигнатуру. Также распространены методы анализа, основанные на эмуляции кода исследуемых программ. Недостатком таких методов является невозможность учесть все множество начальных условий выполнения программы.

В данной работе предлагается метод детектирования вредоносного исполняемого кода, основанный на анализе последовательностей вызовов операционной системы. Делается предположение о том, что в таких последовательностях достаточно часто сосредоточена семантика программы и эти последовательности будут достаточно похожими для программ со сходной функциональностью.

Метод анализа предполагает построение последовательностей вызовов операционной системы по исполняемому коду, которые могут быть исполнены при выполнении программы, составление и ведение базы данных потенциально вредоносных последовательностей и поиск последовательностей из базы в возможных последовательностях вызовов в исследуемой программе.

Для построения возможных последовательностей вызовов исследуемой программы необходима информация о потоке управления.

Пополнение базы данных потенциально вредоносных последовательностей может производиться двумя способами. Первый заключается в «ручном» анализе заведомо вредоносного кода и в поиске последовательностей, определяющих его ключевую функциональность. Данный способ является достаточно трудоемким и требует у эксперта профессиональных знаний высокого уровня. Второй способ заключается в автоматизированном поиске одинаковых (или похожих) последовательностей в двух и более заведомо вредоносных программах. Это могут быть либо разные версии одной вредоносной программы, либо разные программы одного типа (например, типа «троянский конь»).

Очевидно, что последовательности вызовов из библиотеки будут достаточно редко встречаться в исследуемом коде в чистом виде. Поэтому возникает задача поиска последовательностей, «похожих» на библиотечные, а также задача поиска количественных критериев «похожести».

В результате исследования разработано программное средство, позволяющее строить последовательности вызовов из исполняемого модуля, редактировать базу данных потенциально вредоносных последовательностей и осуществлять поиск последовательностей из базы в исследуемой программе. Проведена попытка пополнения базы данных путем автоматизированного поиска в чистом виде общих последовательностей в нескольких вредоносных программах. Поиск общих последовательностей не дал результатов, поэтому необходимо искать последовательности не в чистом виде, а похожие, и, следовательно, вырабатывать количественные критерии «похожести».

ПРИМЕНЕИЕ КЛАСТЕРНОГО АНАЛИЗА ПРИ ПРОГНОЗИРОВАНИИ Q-УРОВНЕВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

К.Г. Кирьянов, С.Н. Можайкин

Нижегородский госуниверситет

Прогнозирование временных рядов (выборок $y_0, y_1,\ldots,y_i,\ldots,y_{M-1}$ из аналоговых непрерывных и разрывных процессов) применимо во многих практически важных случаях (в задачах анализа данных, в криптографии, задачах восстановления пропущенных в результате сбоев наблюдений и т.д.). В данной работе частный случай кластерного анализа [1] применяется при прогнозировании предложенным ранее методом [2] на основе построения таблицы истинности (ТИ) для прогнозирующего оператора $f_n(y_{k-n+1}, y_{k-n+2}, \ldots, y_k)=y_{k+1}$, $k\le M-1$, q-уровневых последовательностей данных. Прогнозируемый q-уровневый временной ряд характеризуется базовыми параметрами q , n и M – числами: уровней квантования ряда, аргументов прогнозирующего оператора и элементов исходного временного ряда, соответственно.

В силу того, что ТИ для исходных q-уровневых последовательностей ограниченной длины $M<\infty$, как правило, не является полной (число строк $M-n<q^n$), при прогнозировании необходимо последовательно отображать отсутствующие в ТИ входные последовательности длины n, содержащие выборки y_s с номерами $s>M-1$, на уже имеющиеся по минимальной норме (минимальному расстоянию) $\|y_k - y_m\|$, где $\boldsymbol{y_s}=(y_{s-n+1}, y_{s-n+2}, \ldots, y_s)$, $s=k$ *u* $s=m \ge M-1$.

Существуют различные способы определения метрики в зависимости от характеристик исследуемых последовательностей. Так, если $q=2$, то под расстоянием будем понимать число различающихся элементов в “n-ках” (дискретная метрика), т.е. расстояние между $\boldsymbol{y_k}=(y_{k-n+1}, y_{k-n+2}, \ldots, y_k)$ *u* $\boldsymbol{y_m}=(y_{m-n+1}, y_{m-n+2}, \ldots, y_m)$, где компоненты y_k векторов принимают значения 0 или 1,

$$\| y_k - y_m \| = \sum_{i=0}^{n-1} \alpha_i (y_k^i \oplus y_m^i) \qquad (1)$$

Для $q>2$ вводится l_p-метрика [1]:

$$\| y_k - y_m \| = \left[\sum_{i=0}^{n-1} \alpha_i \, | y_k^i - y_k^i |^p \right]^{1/p} \qquad (2)$$

Весовые коэффициенты α_i выбираются для процессов с $q>>2$ с учетом большей связи между соседними значениями ряда данных, т.е. придания большего веса последним элементам “n-ки”.

Если элементами временной последовательности являются вектора (из S элементов), то l_p – метрика вводится следующим образом:

$$\| y_k - y_m \| = \left[\sum_{i=0}^{n-1} \alpha_i \rho^p (y_k^{ij}, y_m^{ij}) \right]^{1/p}, \quad (3)$$

$$\rho(y_k^{ij}, y_m^{ij}) = \left[\sum_{j=0}^{S-1} \beta_j \mid y_k^{ij} - y_m^{ij} \mid^p \right]^{1/p}. \quad (4)$$

Порядок прогнозирующего оператора определяется как минимальное число n, при котором по встретившимся одинаковым "n-кам" отсчетов (y_{k-n+1}, y_{k-n+2}, … , y_k) прогнозируются одинаковые значения $y_{k+1} = f_k$. В таблице истинности, соответствующей такому прогнозирующему оператору, имеется только M-n строк из $N=q^n$ возможных. Если все n-мерное фазовое пространство, состоящее из $N=q^n$ точек, разделить на $q < M-n \le q^n$ областей, то все "n-ки" распределяться между q *кластерами* – «сгустками» подмножеств, определяющими "классы эквивалентности" "n-ок".

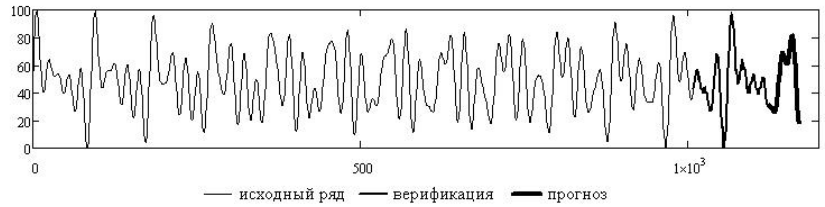

Рис.

На рисунке изображен пример прогноза для полигармонического временного ряда (5 гармоник, q=100) с $\alpha_i = exp(-n+i)$, $i \in [0,n-1]$. Жирной линией показан результат прогноза с метрикой l_2 (2), тонкой – исходные значения. Относительная ошибка прогноза, вычисленная по формуле $\delta = \frac{[\sum_i (x_i - y_i)^2]^{1/2}}{[\sum_i x_i^2]^{1/2}}$, где суммирование идет по области верификации (области проверки качества соответствия алгоритма прогнозируемому временному ряду), составляет 0,13%.

[1] Мандель И. Д. Кластерный анализ. М.: Финансы и статистика, 1988. С.31.
[2] Кирьянов К.Г., Кузнецов Е.С. // Труды XI научной конференции по радиофизике. ННГУ. 2007. С. 228.

МОДИФИКАЦИЯ МЕТОДА ПРОГНОЗИРОВАНИЯ АНАЛОГОВЫХ И ДИСКРЕТНЫХ ПРОЦЕССОВ В ПРОГРАММЕ FORECAST 2

К.Г. Кирьянов[1)], Е.С. Кузнецов[2)]

[1)]*Нижегородский госуниверситет*

[2)]*Нижегородский государственный технический университет*

Способ прогнозирования временных рядов (выборок из аналоговых непрерывных и разрывных процессов), рассмотренный ранее в работе [1] и реализованный в [2], применим во многих практически важных случаях. В настоящей работе предлагается модификация программы Forecast-2, выполненная на основе оптимальных базовых параметров (БП): q (числа уровней квантования выборок ряда данных (1)) и n (числа аргументов так называемого "прогнозирующего оператора" (ПО) для ряда (1)). Прогнозирование основано на предварительном преобразовании исходных временных рядов в q-уровневые ряды длиной M:

$$\{y_k\},\ k = 1,2,...,M < \infty \qquad (1)$$

путём определения их оптимальных БП [1].

После нахождения оптимальных базовых параметров по ряду (1) строится ПО для любого $k=n, n+1, \ldots, M-1$ в виде q-значной логической функции:

$$y_{k+1} = f(y_{k-n+1}, y_{k-n+2}, \ldots, y_k) \equiv f_k \qquad (2)$$

эквивалентной таблице истинности (ТИ). При этом ТИ удобно интерпретировать как эталонную и хранить в форме q-строчной таблице кодирования (табл.), где всем кодам μ-й строки (μ-му классу эквивалентности), состоящей *из* $s^{(\mu)}$ кодов таблицы $k_{\mu,\lambda}$, ($\mu=1,2,\ldots, q$; $\lambda=1,2,\ldots,s^{(q)}$), соответствует одинаковое прогнозируемое значение $y^{(\mu)}$ исходного ряда.

Таблица

Класс №	Классы эквивалентности "n-ок"	Прогноз ≡ номер класса	количество "n-ок" в классе
1	$k_{1,1}, \ldots, k_{1,s_1}$	$y^{(1)}$	S_1
2	$k_{2,1}, \ldots, k_{2,s_2}$	$y^{(2)}$	s_2
	…	…	**…**
μ	$k_{\mu,1}, \ldots, k_{\mu,s_\mu}$	$y^{(\mu)}$	s_μ
	…	…	**…**
q-1	$k_{q-1,1},\ldots,k_{q-1,s_{q-1}}-1$	$y^{(q\text{-}1)}$	$s_{q\text{-}1}$
q	$k_{q,1}, \ldots, k_{q,s_q}$	$y^{(q)}$	s_q

Прогнозирование при $k>M$ заключается в пошаговом построении продолжения ТИ с $M-n+1$-й по $M+sf$-ю строку, где $sf = 1,\ldots,Lf$, а Lf – номер максимального шага прогнозирования или т.н. «прогнозного горизонта» для пополнения выборок данных (1), имеющихся в исходной ТИ. В данной работе для определения

$y_{M+sf}=f_{M+sf-1}$ используется последовательное сравнение y_{M-n+sf} -ой “n-ки” (y_{M-n+sf}, $y_{M-n+sf+1}$, …, y_{M+sf-1}) со всеми “n-ми”, уже имеющимися в исходной таблице, рассматриваемыми как опорные («эталонные») по критерию «минимума расстояния» между “n-ми”:

$$y_{M+sf} = \arg \min_{y_k \in [y_{n+1}, y_{n+2}, \ldots, y_{M+sf-1}]} d_k^i(sf), \tag{3}$$

где

$$d_k^i(sf) = \sum_{j \in [1,n]} w_j^i \cdot | y_{k+1-j} - y_{M+sf-j} | / p_\mu, \tag{4}$$

$$p_\mu = S_\mu \Big/ \sum_{\lambda=1}^{q} S_\lambda \quad , \; k = n+1, n+2, \ldots, M+sf-1$$

Здесь p_μ – оценка вероятности появления «n-ки» μ-го класса эквивалентности, к которому относится k-ая «n-ка». В критерии близости (4) используются весовые функции индекса $j = 1,\ldots,n$ с типом веса $i=\{c,l,e,h\}$:

$$w_j^{(c)} = 1, \; w_j^{(l)} = 1 + (1-j)/n, \; w_j^{(e)} = e^{-j}, \; w_j^{(h)} = j^{-1} \tag{5}$$

Исследования, проведенные с помощью информационной системы прогнозирования Forecast-2, модифицированной указанным выше способом показали, что можно улучшить прогнозирование процессов различных классов (природных, полигармонических, хаотических, экономических). В качестве оценки точности прогноза использовалось отношение среднеквадратического отклонения прогнозируемых данных от исходных к дисперсии исходных данных (D_{Lf}) на участке прогнозирования:

$$Er_{sf}(Lf) = \frac{1}{\sqrt{D_{Lf}}} \cdot \sqrt{\frac{1}{Lf-1} \cdot \sum_{sf=1}^{Lf} \left| y_{M+sf-1} - y'_{sf} \right|^2}, \tag{6}$$

где y_{sf} – исходные данные, y'_{sf} – прогнозируемые значение сигнала, а Lf – длина прогноза. Например, прогнозирование ряда значений ежегодного прироста колец деревьев при $q_{opt}=25$, $n_{opt}=4$ [1] с различными весовыми функциями индекса j показало, что модификация формулы (4) позволила улучшать точность прогноза для весов типа $i=e$ и $i=h$ примерно в 1.5 раза ($Er_{sf}(2) \cong 0.05$).

[1] Кирьянов К.Г., Кузнецов Е.С. // Труды XI научной конференции по радиофизике. Н.Новгород: Изд-во ННГУ. 2008. С.226.

[2] Кирьянов К.Г., Кузнецов Е.С. / Свидетельство о государственной регистрации программы для ЭВМ №2008611799 «Прогнозирование временных рядов Forecast-2». 2008.

ПРЕОБРАЗОВАТЕЛЬ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДАННЫХ С РЕГУЛИРУЕМОЙ АЛГЕБРАИЧЕСКОЙ СТРУКТУРОЙ

К.Г.Кирьянов, С.В. Лобанов

Нижегородский госуниверситет

Преобразователь предназначен для обработки дискретных последовательностей данных, при этом сами элементы последовательности могут принадлежать к различным алгебраическим структурам [1]. Данный преобразователь может применяться как для частотной фильтрации дискретизированных сигналов, так и для преобразования последовательностей, таких как генетические тексты, изначально имеющие дискретное представление.

В основу преобразователя положено Z-преобразование [2], являющееся аналогом преобразования Лапласа для дискретных сигналов. Рассматривается передаточная характеристика, представленная в виде отношения двух полиномов от z^{-1}:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ...}{a_0 + a_1 z^{-1} + a_2 z^{-2} + ...}, \quad (1)$$

где a_0, a_1, a_2, … , b_0, b_1, b_2,…– параметры преобразователя, принадлежат выбранной алгебраической структуре и задаются пользователем. На языке Z-преобразования преобразователь описывается формулой $Y(z)=H(z)X(z)$, где X и Y – образы входной и выходной последовательностей. Представив $H(z)$ в виде (1), получим:

$$(a_0 + a_1 z^{-1} + a_2 z^{-2} + ...)Y(z) = (b_0 + b_1 z^{-1} + b_2 z^{-2} + ...)X(z). \quad (2)$$

Далее, воспользуемся свойством Z-преобразования:

$$Z^{-1}\{z^{-n}X(z)\} = x(t-n),$$

свойством линейности Z-преобразования:

$$Z^{-1}\{X_1(z) + X_2(z)\} = Z^{-1}\{X_1(z)\} + Z^{-1}\{X_2(z)\},$$

положив a_0=1 и сделав обратное Z-преобразование от выражения (2), получим алгоритм преобразователя:

$$y(t) = (b_0 x(t) + b_1 x(t-1) + b_2 x(t-2) + ...) - (a_1 y(t-1) + a_2 y(t-2) + ...). \quad (3)$$

Как видно из формулы (3), для данного алгоритма достаточно иметь три арифметических операции в алгебраической структуре. В рассматриваемом преобразователе можно выбирать самые различные структуры: действительные числа (16-битное представление), Z_q [3], а также произвольное кольцо с конечным количеством элементов, задаваемое таблицами сложе-

Табл. 1

+	*A*	*B*	*C*	*	*A*	*B*	*C*
A	A	B	C	A	A	A	A
B	B	C	A	B	A	B	C
C	C	A	B	C	A	C	B

ния и умножения, например такими, как в приведенной Таблице 1. Таблица вычитания строится автоматически по таблице сложения.

В алгебраической структуре Z_q задаётся только параметр q, который определяет количество элементов. Все операцию производятся по модулю q.

Программа, осуществляющая преобразование может записывать результаты в файл, а также входные и выходные последовательности, отображать графически, например, в форме, изображенной на рисунке, что позволяет удобно анализировать результаты работы преобразователя.

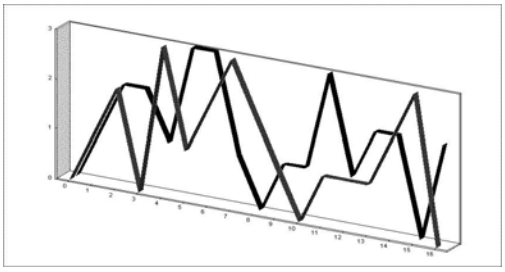

Рис.

Элементами алгебраической структуры могут быть не только числа, но также и буквы, при этом алгебраическая структура входной последовательности должна быть такой же, как у коэффициентов $H(z)$. Это может быть удобным при работе с генетическими текстами, когда аминокислоты обозначены первыми буквами своего названия.

Пример такого преобразования:

Алгебраическая структура: Табл. 1

Коэффициенты $H(z)$: b_0=B, b_1=C, b_2=A, b_3=B, a_1=B, a_2=C, a_3=C

Входная последовательность: A B C B B B A C A A B

Выходная последовательность: A B A A C A C A B B A

[1] Макклеллан Дж. Х.,Рейдер Ч. М. Применение теории чисел в цифровой обработке сигналов. М.: Радио и связь, 1983. С.338.

[2] Баскаков С.И. Радиотехнические цепи и сигналы. М.: Высш. школа, 1988. С.338.

[3] Курош А.Г. Курс высшей алгебры. М.: Наука, 1968. С. 278.

ОПРЕДЕЛЕНИЕ АНОМАЛЬНОГО ПОВЕДЕНИЯ ПРОЦЕССОВ НА ОСНОВЕ ИХ ПРОЦЕССОРНЫХ ВЫЗОВОВ В ОПЕРАЦИОННЫХ СИСТЕМАХ СЕМЕЙСТВА UNIX

В.А. Сангалов, С.В. Корелов, Л.Ю. Ротков

Нижегородский госуниверситет

Сегодня для любой операционной системы (ОС) актуальным является вопрос ее безопасности. Один из эффективных путей его решения – обнаружение аномального поведения процессов на основе последовательности их системных вызовов.

Последовательности системных вызовов процессов в ОС семейства Unix хорошо подходят для формирования признаков их штатного поведения, уникальных для

каждого объекта. Используя эти признаки можно не только обнаружить факт аномального поведения процесса, но и отследить его вариации для последующей идентификации видов атак. В ОС семейства Unix наиболее актуальным является мониторинг привилегированных процессов, т.к. они наиболее опасны по сравнению с пользовательскими.

Для формирования штатного поведения процессов ОС семейства Unix можно использовать один из следующих методов перехвата системных вызовов:

1. Замена таблицы sys_call_table. При этом необходимо создавать свою таблицу и перед совершением вызовов анализировать необходимые параметры. Реализация данного метода затруднена необходимостью вновь переписывать всю таблицу системных вызовов.

2. Замена обработчика прерывания. В данном случае необходимо подменять функцию, вызывающую системный вызов.

3. Использование системного вызова “ptrace”.

4. Использования специальных утилит.

Последние два способа наиболее удобны для реализации мониторинга поведения процесса, т.к. они написаны разработчиками ОС и гарантированно работают в разных версиях ОС, что делает систему мониторинга универсальной, в то время как первые два зависят от конкретной архитектуры ОС и могут некорректно работать на различных версиях. В том случае, если интересует только факт наступления вызова, а не параметры, которые в него передаются, то использование системного вызова “ptrace” является наиболее простым.

Предлагаемый алгоритм обнаружения аномального поведения процессов состоит из двух этапов. На первом происходит трассировка исследуемого процесса при его штатном функционировании и составление базы данных его нормального поведения (БДНП). Сформировать БДНП можно, либо смоделировав различные варианты работы процесса, либо при его функционировании в режиме реального времени. БДНП должна отражать все возможные варианты штатного поведения процессов и иметь небольшой объем. На втором этапе процесс трассируется во время штатной работы. Решение об аномальности его поведения принимается путем сравнения полученных последовательностей с имеющимися в БДНП. На данном этапе также возможна корректировка БДНП.

Для построения БДНП и дальнейшего мониторинга используется метод «скользящего окна». При этом возможны ошибки «первого» и «второго» рода. В связи с этим необходимо определить значение порогового уровня допустимого совпадения с последовательностями в БДНП. В качестве меры предлагается использовать расстояние Хемминга, в общем случае выражаемое формулой:

$$d_H\left(X_i, X_j\right) = \sum_{s=1}^{p} \left| x_i^{(s)} - x_j^{(s)} \right|,$$

где d_H – расстояние Хемминга, X_i, X_j – объекты, p – размерность.

По расстоянию Хемминга можно судить о степени совпадения текущего поведения процесса с хранящимися в БДНП и на основе этого принимать решение об

аномальности его поведения. В связи со сложностью теоретического обоснования меры нормального поведения допустимое значение d_H предлагается выбирать эмпирическими методами в ходе составления БДНП.

Представленный способ является эффективным методом аудита нормальной работы ОС и обнаружения атак за счёт адаптации его под конкретные особенности операционной системы и отслеживание изменений в программном обеспечении или административных политиках.

[1] Hofmeyr S., Forrest S., Somayaji A. // Journal of Computer Security. 1998. V.6. P.151.
[2] Warrender C., Forrest S., Pearlmutter B. // Proc. 1999 IEEE Symposium on Security and Privacy. 1999. P. 133.
[3] http://seclab.cs.sunysb.edu/seclab/pubs/papers/ndss00.pdf.
[4] http://tldp.org/LDP/lkmpg/2.6/html/.
[5] Hamming R. // Bell System Technical Journal. 1950. V.29, No.2. P.147.

ОБНАРУЖЕНИЕ СПАМА ПРИ ПОМОЩИ АППАРАТА WAVELET-ПРЕОБРАЗОВАНИЙ

В.В. Агаджанов, С.В. Корелов, Л.Ю. Ротков

Нижегородский госуниверситет

В данной статье рассмотрен алгоритм обнаружения спама на основе wavelet-преобразования текста письма, а также представлены некоторые практические результаты работы данного алгоритма.

Любая система характеризуется набором параметров, признаков, определяющих как саму систему, так и ее поведение. Для текста таковыми являются длина, частота появления отдельных букв, частота появления слов и т.д. Некоторые из этих характеристик успешно используются алгоритмами обнаружения спама, например такими, как фильтрация по Байесу. Однако при анализе писем ни один метод не дает стопроцентного результата.

Поэтому можно сделать вывод, что существующий анализируемый различными алгоритмами набор характеристик текста не является полным.

Аппарат wavelet-преобразований дает возможность пополнить этот набор.

Алгоритм обнаружения можно условно разбить на три этапа:

1) Формирование банка данных, в котором хранятся значения характеристик как для спам так и не-спам текстов.
2) Получение характеристик для конкретного, анализируемого текста.
3) Сравнение характеристик и формирование результата.

Обобщенная схема анализа текста представлена на рис.

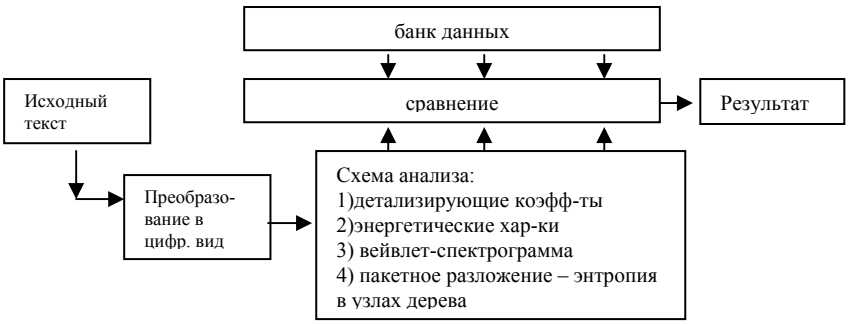

Рис.

При анализе текстов использовались различные wavelet-преобразования, однако результаты по сравнению спам и не-спам писем, а также частота появления «ложной тревоги» позволили судить о наибольшей эффективности вейвлетов Добеши, «мексиканская шляпа» и гауссовых вейвлетов различных порядков. Кроме того, часть информации хранится не в самих наборах детализирующих и аппроксимирующих коэффициентов преобразования, а в характеристиках, полученных из них: это энергетический Фурье-спектр восстановленных из детализирующих коэффициентов текста, а также энтропия в различных узлах (при пакетном разложении).

Как видно из схемы, отнесение текста к спаму носит вероятностный характер, поэтому для формирования эффективного банка данных необходимо провести достаточно большое количество испытаний над уже существующими текстами. В частности, при анализе пятидесяти писем алгоритм отбрасывает меньше половины спам-текстов, однако при увеличении их числа до пятисот результативность значительно возрастает.

ПРИМЕНЕНИЕ МЕТОДА ПОСТРОЕНИЯ ГЕНЕТИЧЕСКОЙ КАРТЫ ТЕКСТА ДЛЯ ИДЕНТИФИКАЦИИ СПАМА

С.В. Корелов, А.К. Крюков, Л.Ю. Ротков

Нижегородский госуниверситет

Исследуется возможность применения метода построения генетической карты текстового сообщения для фильтрации спама.

Были взяты выборки легальных рассылок и спама, прошедшего через существующие анти-спам фильтры. Каждая выборка была разделена на две группы: первая – для формирования цифрового образца рассылок и спама, вторая – для тестирования. Для каждого письма строилась генетическая карта. Цифровой образец выборки строился усреднением генетических карт писем первой группы. Генетическая карта каждого письма из второй группы сравнивалась с цифровыми образцами

выборок. При сравнении использовался метод наименьших квадратов. Таким образом, при поступлении нового письма в почтовый ящик можно сделать вывод о его принадлежности к спаму.

Основной результат, на основании которого можно делать вывод о применимости данного метода для фильтрации спама – вероятности ошибок пропуска спама ($p_{ПЦ}$) и вероятности идентификации письма легальной рассылки как спама ($p_{ЛТ}$).

Метод генетических карт был протестирован на ряде выборок различного размера.

Перед построением генетической карты текст должен быть проквантован. В работе рассматривались только русскоязычные тексты, поэтому максимальное количество уровней квантования – 78 (буквы, цифры, основные знаки препинания). Семейство кривых – зависимости вероятностей ошибок от количества уровней квантования текста для различного размера выборок – представлено на рис.1.

На рис.2 представлены зависимости вероятностей ошибок от количества писем в выборке.

Из результатов, показанных на рис.1, видно, что вероятности ошибок имеют тенденцию к возрастанию при уменьшении количества уровней квантования.

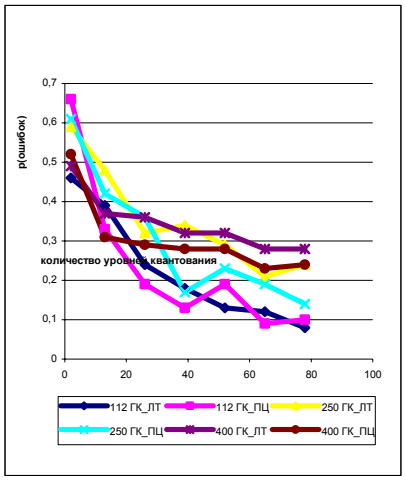

Рис. 1

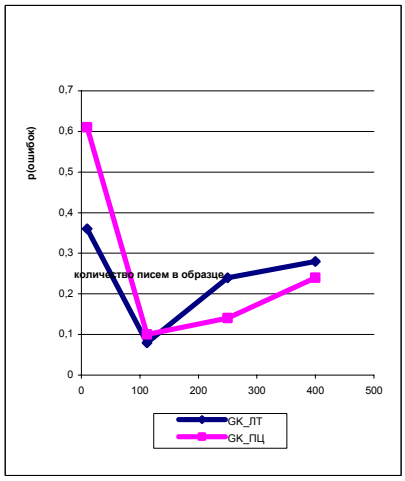

Рис. 2

Из полученных результатов, которые отображены на рис.2, были сделаны следующие выводы: оптимальное количество писем для формирования одного образца находится в пределах от 80 до 200. Таким образом, большие выборки спама и легальных рассылок имеет смысл группировать в несколько образцов по семантическому или иному признаку. Каждый образец должен рассматриваться независимо.

В результате было взято 8 групп легальных рассылок (всего 1180 писем) и 6 групп спама (всего 1180 писем). Легальные рассылки сгруппированы по названиям

рассылок. Спам сгруппирован двумя способами: по семантическому признаку и по объему письма.

Пропуском спама считалось отнесение спама к какой-либо группе легальных рассылок, легального письма к какой-либо из групп спама.

Результаты показали сильную зависимость вероятности ошибок от семантики писем. В зависимости от группы вероятности ошибок составляли от 13 до 41% в случае разбиения спама по семантическому признаку, и от 9 до 36% при разбиении спама по объему письма. Средние значения вероятностей ошибок при разбиении спама по группам: $p_{ПЦ}$=29%, $p_{ЛТ}$=19%. Средние значения вероятностей ошибок при разбиении спама по объему письма: $p_{ПЦ}$=26%, $p_{ЛТ}$=13%.

Таким образом, применение метода генетических карт позволяет фильтровать до 74% спама при ошибочном принятии решения для 13% легальных рассылок. При квантовании рекомендуется установить минимальный разумный шаг квантования (т.е., не пропускать никакие символы). Для эффективной работы фильтра нужно группировать письма легальных рассылок по названиям тем, а спам – по объему письма либо использовать более эффективное разделение по семантике. При очень большой выборке спама (N>>1000 писем), вероятно, имеет смысл комплексного разделения спама: по объему, а внутри объема – по семантике.

[1] Кирьянов К.Г. Выбор оптимальных базовых параметров источников экспериментальных данных при их идентификации, 2004. 22 с.

СРЕДСТВО ИССЛЕДОВАНИЯ АЛГОРИТМОВ УПРАВЛЕНИЯ ПОТОКОМ В УСЛОВИЯХ РЕАЛЬНОЙ СЕТИ

Г.А. Салагацкий, Д.В. Яковлев

Нижегородский госуниверситет

Тема исследования алгоритмов управления потоком, которые использует протокол TCP, активно разрабатывается последние десять лет многими авторами [1-5]. Предлагаются как принципиально новые алгоритмы [1,2], так и модификации алгоритмов широко использующихся в текущее время [3]. Большинство исследований проводится с помощью сетевого симулятора NS2 [6], который фактически стал стандартным инструментом большинства исследователей. Гораздо реже предложенные идеи и алгоритмы доходят до этапа проверки в условиях реальной сети. Причем использование симулятора у многих авторов часто ограничивается только простыми сетевыми моделями [1-4]. К примеру, чрезвычайно популярны варианты «гантели» [1,2]. Более сложные модели, имитирующие многосоставной канал с участками разной пропускной способности и источниками фонового трафика, рассматриваются в [5].

Если исследователя интересует наблюдение поведения его алгоритма управления потоком в условиях реальной сети, то для этого необходимо модифицировать часть кода операционной системы, отвечающую за стек TCP/IP. Для операционных

систем с открытым исходным кодом, таких как Linux, это технически возможно, но требует достаточной квалификации программиста.

В результате проведенной работы создано программное средство, позволяющее меньшими усилиями решить эту и некоторые вспомогательные задачи специалистом без опыта программирования модулей ядра ОС Linux. Разработанное программное средство в частности позволяет: открыть отдельное сетевое соединение TCP/IP с нестандартным алгоритмом управления потоком; проводить мониторинг и модификацию параметров TCP/IP соединения недоступных в стандартной ОС; собрать информацию о состоянии канала связи текущего соединения доступную на стороне отправителя для дальнейшего воссоздания подобных условий в симуляторе; смоделировать поведение соединения во встроенном простом симуляторе.

Наличие встроенного симулятора дает гарантии того, что при симуляции и при работе с реальной сетью используются одни и те же реализации алгоритмов управления потоком.

Использование разработанного программного средства при исследовании алгоритмов управления потоком TCP-соединения помогает ответить на многие вопросы, интересующие исследователя, как то: оценить возможности определения состояния канала связи и физических параметров сети со стороны отправителя и использовать полученные данные при симуляционном моделировании; оценить возможности адекватной симуляции реальной сети моделями различной сложности и др.

Программное средство реализовано на языке C# и имеет многоуровневую модульную структуру. Нижний уровень представляют системно зависимые библиотеки перехвата и фильтрации пакетов и взаимодействия с сетевым драйвером канального уровня. Средний уровень содержит функции, отвечающие за реализацию стека TCP/IP. Верхний уровень образуют приложения использующие стек для передачи своих данных, простой симулятор и интерпретатор управляющих приложениями скриптов. Для удобства автоматизации исследований предлагается писать управляющие приложениями скрипты на языках C# и Visual Basic. Описанная иерархическая структура позволяет гибко расширять функциональность и независимо модифицировать компоненты различных уровней в зависимости от текущих потребностей исследователя.

Эксперименты с использованием описанного программного средства позволяют судить о низкой эффективности многих современных алгоритмов управления потоком в условиях работы соединения в сильно загруженном разделяемом канале связи. Современные алгоритмы управления потоком пытаются сформировать оценку доступной полосы пропускания канала связи, его емкости с учетом задержек соединения и оптимизируют размер блока передаваемых данных в соответствии с этими оценками. Однако в «агрессивной среде» сильно загруженного разделяемого канала связи, в котором пульсирующий фон создают короткоживущие соединения, сформировать и эффективно использовать указанные оценки практически очень сложно. Широкое внедрение средств управления трафиком на сервисах предоставляющих услуги хранения данных, на proxy-серверах, контролирующих исходящий

трафик локальных сетей и в самой основе современных технологий мультисервисных сетей (MPLS) является дополнительным источником непредсказуемых для TCP-клиента пульсаций доступной полосы пропускания. Эффективно подстраиваться в таких условиях под доступные ресурсы – чрезвычайно непростая задача и исследование поведения нового алгоритма в реальной сети может дать дополнительные сведения о его конкурентоспособности.

[1] Mascolo S., Sanadidi M.Y., Casetti C., Gerla M., Wang R. // Wireless Networks J. 2002. V.8. P.467.
[2] Capone A., Fratta L., Martignon F. // IEEE Transactions on Mobile Computing. 2004. V.3, No.2.
[3] Bhandarkar S., Jain S., Narasimha Reddy A.L. // ACM SIGCOMM Computer Communication Review. 2006. V.36. P.41.
[4] Mo J., La R.J., Anantharam V., Walrand J. // Proc. IEEE Computer and Communications Societies. 1999. V.3. P.1556.
[5] Dovrolis C., Ramanthan P., Moore D. //Proc. Proc. IEEE Computer and Communications Societies. 2001. V.2. P.905.
[6] NS-2 network simulator (ver. 2): http://www.isi.edu/nsnam, 2008.