DOI: 10.22184/1992-4178.2023.231.10.80.86

Надежность и безопасность операционных систем различной архитектуры

Часть 3

С. Назаров, д. т. н.[1], А. Барсуков, к. т. н.[2]

УДК 621.3.012 | ВАК 2.2.11

В заключительной части статьи представлена модель операционной системы Касперского. На сегодняшний день KasperskyOS – это микроядерная частично POSIX-совместимая операционная система, не являющаяся системой общего назначения. Ее основное предназначение – разработка ИТ-продуктов для отраслей с повышенными требованиями к кибербезопасности, надежности и предсказуемости работы. Однако реальна возможность ее развития до операционной системы универсального применения. Цель статьи – показать, насколько кибериммунный принцип построения системы повышает надежность и безопасность создания операционных систем по сравнению с системами другой архитектуры.

ОСОБЕННОСТИ АРХИТЕКТУРЫ KasperskyOS

Прежде всего, надо отметить, что система полностью создана с чистого листа специалистами «Лаборатории Касперского», является оригинальной разработкой компании и зарегистрирована в реестре российского ПО как рекомендованная для приобретения отечественными организациями и государственными структурами [18]. Основой ОС служит надежное микроядро, которое допускает только строго определенный способ взаимодействия системных процессов и обмена данными, тем самым обеспечивая полноту контроля доступа. Будучи компактным, оно может портироваться на различные аппаратные платформы. KasperskyOS построена на базе концепции MILS (Multiple Independent Levels of Security – «множественные независимые уровни защиты/безопасности»), определяющей строгие принципы изоляции системных процессов и политик управления информационными потоками. Одним из наиболее важных компонентов ОС является модуль контроля доступа Kaspersky Security System (KSS), отслеживающий все межпроцессорные взаимодействия и исключающий доступ приложений к защищенным областям памяти с критически важными процессами и данными. KSS поддерживает широкий набор правил и политик доступа и может быть настроен в соответствии с конкретными требованиями заказчика. Любое действие, не предусмотренное политиками безопасности, запрещено по умолчанию. И это, пожалуй, самое важное отличие KasperskyOS от популярных сегодня операционных систем, в которых все, что не запрещено, по умолчанию разрешено.

Микроядро KasperskyOS – собственная разработка «Лаборатории Касперского». Как дизайн, так и реализация всех частей микроядра отвечают главной задаче – созданию безопасной операционной системы [19]. В KasperskyOS всего три системных вызова и только один интерфейс межпроцессного взаимодействия. Благодаря этому поверхность атаки минимальна. Микроядро KasperskyOS спроектировано так, чтобы по максимуму использовать возможности Kaspersky Security System. И ее микроядро – действительно «микро» (не более тысячи строк кода). В них втиснуты диспетчер процессов, механизм межпроцессного взаимодействия (IPC – interprocess communication), модуль контроля доступа KSS, который следит за механизмом IPC.

Таким образом, базовый принцип KasperskyOS аналогичен общему подходу любых микроядерных систем. Процессы взаимодействуют между собой и с функциями ядра системы, отправляя и получая IPC-сообщения. Микроядро предоставляет им эту возможность с помощью

[1] Московский научно-исследовательский телевизионный институт, главный научный сотрудник, профессор.

[2] Московский научно-исследовательский телевизионный институт, заместитель начальника научно-технического отдела, с. н. с.

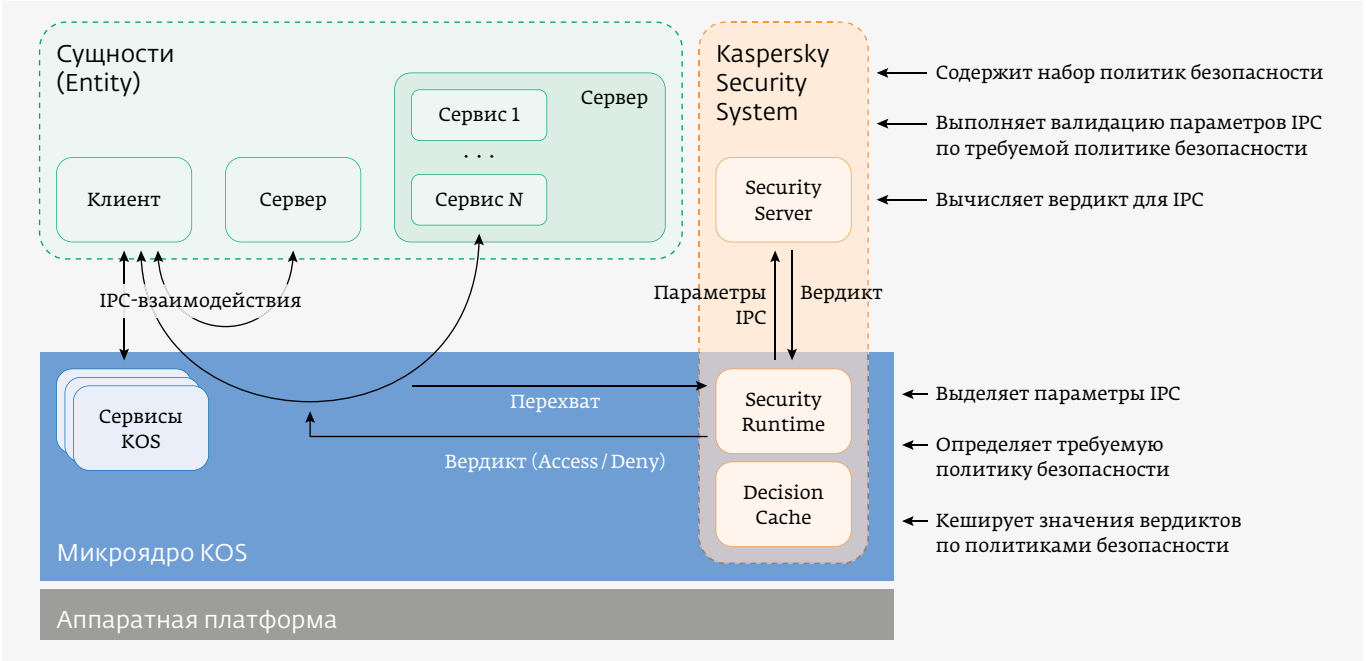

Рис. 4. Схема межпроцессного взаимодействия (из статьи Е. Лебеденко. Операционная система от «Лаборатории Касперского». Как устроена KasperskyOS. 21.12.2015)

механизма IPC. В случае с KasperskyOS за этим механизмом следит KSS, которая для каждого IPC-сообщения выносит вердикт «можно» (allow) или «нельзя» (deny). При этом по умолчанию KSS реализует принцип default deny. То есть если программа по какой-то причине не реализует такую модель взаимодействия, то ни отправить, ни получить IPC-сообщения она не сможет. И останется в полной изоляции.

Вся остальная функциональность операционной системы, включая драйверы, файловые системы и сетевые стеки, вынесена в режим пользователя. Ядро KasperskyOS гарантирует полную изоляцию компонентов IT-системы. Единственный вид межпроцессных взаимодействий, предоставляемый ядром, – синхронный обмен сообщениями («запросом» и «ответом»). При этом каждое сообщение передается подсистеме Kaspersky Security System для проверки на соответствие заданной политике безопасности. Ядро доставит сообщение, только если это разрешено политикой. Таким образом, базовый принцип KasperskyOS идентичен общему подходу любых микроядерных систем. Процессы взаимодействуют между собой и с функциями ядра системы, отправляя и получая IPC-сообщения. Микроядро предоставляет им эту возможность с помощью механизма IPC (рис. 4). В случае KasperskyOS за этим механизмом следит KSS, которая для каждого IPC-сообщения выносит вердикт «можно» (allow) или «нельзя» (deny). При этом по умолчанию KSS реализует принцип default deny. Если программа по какой-то причине не реализует такую модель взаимодействия, то ни отправить, ни получить IPC-сообщения она не сможет и останется в полной изоляции. Следовательно, ни для вирусов, ни для вредоносного ПО, ни для «криворуко» написанного софта нет возможности нарушить вычислительный процесс.

Учитывая изложенные особенности построения микроядерной кибериммунной операционной системы KasperskyOS, ее модель графа состояний и переходов можно представить следующим образом (рис. 5).

Будем считать, что пользователь в работе может использовать свою программу, а также пользоваться услугами сервисов и сервера операционной системы KasperskyOS. Перечислим состояния и соответственно вероятности нахождения системы в этих состояниях.

P_1 – вероятность того, что пользователь системы занят работой, в этом состоянии система исправна и работоспособна.

P_2 – пользователь простаивает в связи с отказом со стороны ОС в использовании ее сервисов или сервера (причины: небезопасный код программы пользователя, неправильный запрос на услуги ОС и др.). В этом состоянии возможна перезагрузка программы и попытка дальнейшей работы. Операционная система исправна, поскольку она практически в силу небольшого объема кода (несколько десятков строк кода) практически безотказна. Для определенности примем наработку на отказ, равную 1000 000 ч.

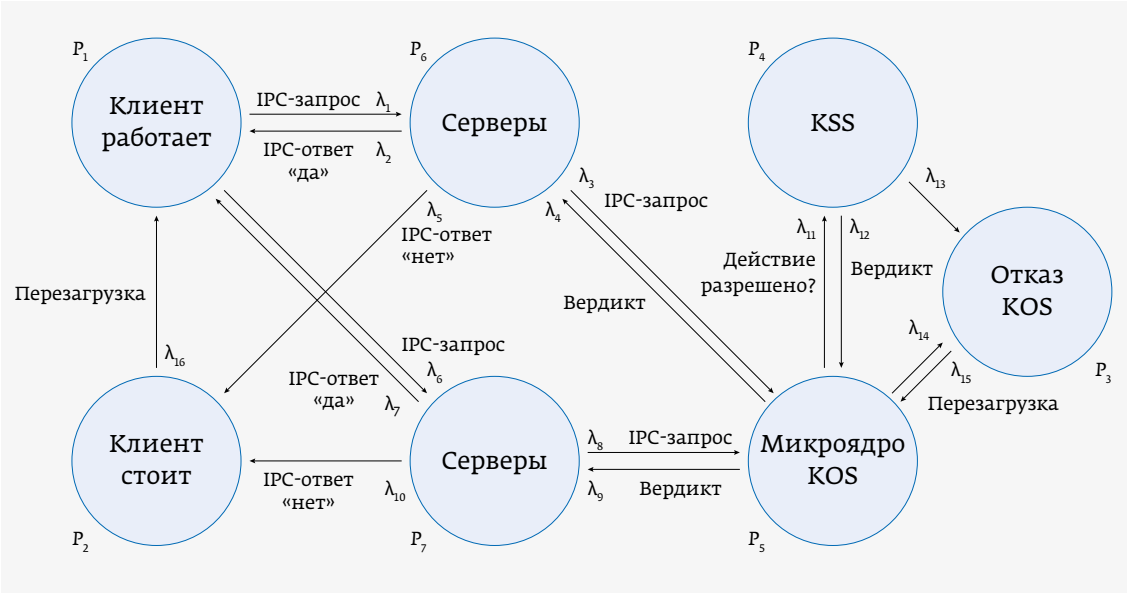

Рис. 5. Граф состояний и переходов KOS

P_3 – состояние отказа KOS по причине проявления ошибки в KSS или микроядре KOS. В этом случае возможна перезагрузка системы и дальнейшая работа пользователя.

Дальнейшие состояния и, соответственно, вероятности отображают работу компонентов KOS.

P_4 – работа подсистемы Kaspersky Security System (KSS).

P_5 – работа микроядра KOS.

P_6 и P_7 – работа сервисов и сервера системы, соответственно.

Перейдем к определению интенсивностей перехода между состояниями системы. Примем на основе данных статьи [2] интенсивность обращения клиента к услугам сервисов $\lambda_1 = 100\,000$ 1/с. В любом случае получение требуемой услуги происходит по схеме, приведенной на рис. 6. Как видно из этого рисунка, происходит некоторая задержка в обработке запроса клиента (дважды работает ядро и KSS), однако в нашей модели это можно не учитывать. Поэтому $\lambda_3 = \lambda_4 = 100\,000$ 1/с. В потоке λ_4 передается вердикт, определяющий возможность правильного выполнения запроса. Ответ «да» определяет интенсивность потока λ_2, и, соответственно, ответ «нет» определяет интенсивность потока λ_5. Будем считать, что с вероятностью, близкой к 1, например $0{,}9999999$, будет получен ответ «да», в этом случае вероятность ответа «нет» равна $0{,}00000001$. В этих условиях интенсивности потока $\lambda_2 = \lambda_1 \cdot 0{,}9999999 = 100\,000 \cdot 0{,}9999999 = 9999999{,}9$ 1/с, соответственно $\lambda_5 = 0{,}001$ 1/с. Рассмотрим обращение клиента к услугам сервера. Пусть время выполнения сервиса KOS = $0{,}05$ мс, а время выполнения драйвера сервера (печать на мониторе) – 5 мс [2]. Предположим, что на

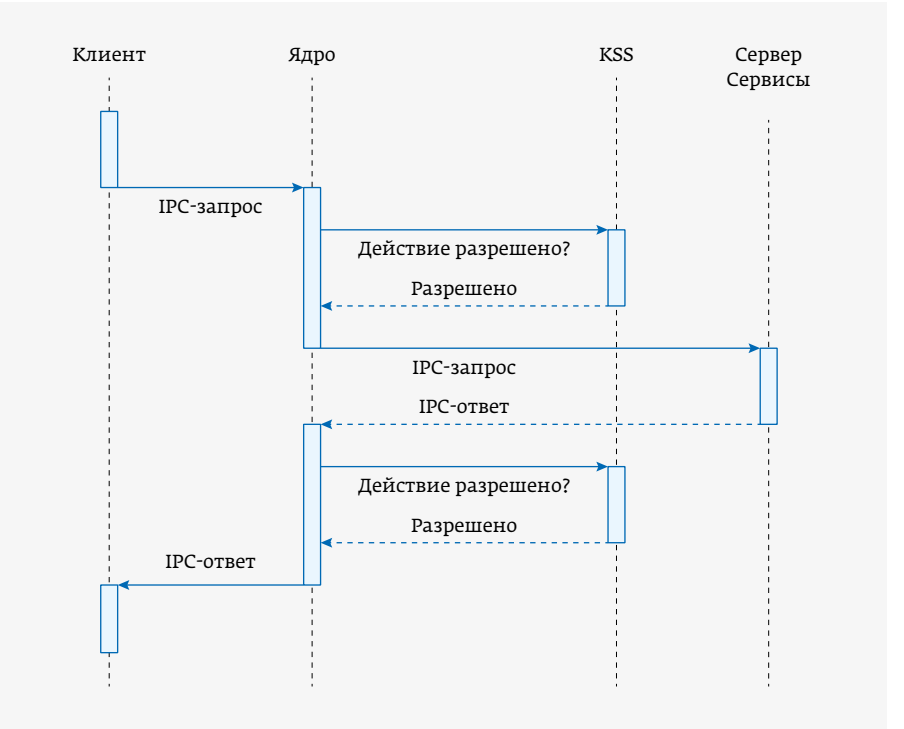

Рис. 6. Обмен сообщениями Клиент-Сервер

10 000 запросов клиента приходится одно обращение к драйверу печати. Тогда среднее время обслуживания составит значение

$$t_{ср} = \frac{0,05 \cdot 10\,000 + 5}{10\,000} = 0,0505 \text{ мс} = 0,0000505 \text{ с}.$$

Таким образом, получаем, что интенсивность потока $\lambda_6 = 1/0,0000505 \approx 19\,802$ 1/с. Интенсивности потоков λ_8 и λ_9 имеют такое же значение. Последний поток содержит значение вердикта. Ответ «да» определяет в этом случае интенсивность потока λ_7, соответственно ответ «нет» определяет интенсивность потока λ_{10}. Примем условие, что с вероятностью 0,999999 будет получен ответ «да», в этом случае вероятность ответа «нет» равна 0,000001. В этих условиях интенсивности потока $\lambda_7 = \lambda_9 \cdot 0,999999 = 19\,802 \cdot 0,999999 = 19\,801,98$ 1/с, соответственно $\lambda_{10} = 0,02$ 1/с.

Интенсивность потока λ_{11} определяется суммой интенсивностей λ_3 и λ_8 и, следовательно, равна 119 802 1/с. Такую же интенсивность имеет поток вердиктов λ_{12}. Интенсивность потоков λ_{13} и λ_{14} определяется только возможными программными ошибками в KOS. Пересчитывать количество таких ошибок, как это делалось выше, по статистическим данным в зависимости от числа строк программного кода не имеет смысла. Поверим заявлениям разработчиков KasperskyOS об отсутствии программных ошибок и для определенности будем считать, что наработка на отказ KOS равна 1 000 000 ч. На основании этого можно считать, что интенсивности отказов λ_{13} и λ_{14} составляют значение, равное 1/1000000 ч. Перевод в секунды дает значение $\lambda_{13} = \lambda_{14} = 0,00000000028$ 1/с. Что касается интенсивности потоков перезагрузки, примем ее равной значению в предыдущих моделях, то есть $\lambda_{15} = \lambda_{16} = 0,0055$ 1/с.

По полученному и размеченному графу состояний и переходов в соответствии с правилом, изложенным в разделе «Метод и ограничения, принятые при разработке моделей архитектур операционных систем», составляем систему линейных алгебраических уравнений, описывающих стационарный режим работы KOS:

$$\begin{cases} (\lambda_1 + \lambda_6) \cdot P_1 = \lambda_2 \cdot P_6 + \lambda_7 \cdot P_7 + \lambda_{16} \cdot P_2; \\ \lambda_{16} \cdot P_2 = \lambda_5 \cdot P_6 + \lambda_{10} \cdot P_7; \\ \lambda_{15} \cdot P_3 = \lambda_{13} \cdot P_4 + \lambda_{14} \cdot P_5; \\ (\lambda_{12} + \lambda_{13}) \cdot P_4 = \lambda_{11} \cdot P_5; \\ (\lambda_9 + \lambda_4 + \lambda_{11} + \lambda_{14}) \cdot P_5 = \lambda_8 \cdot P_7 + \lambda_3 \cdot P_6 + \lambda_{12} \cdot P_4 + \lambda_{15} \cdot P_3; \\ (\lambda_2 + \lambda_5 + \lambda_3) \cdot P_6 = \lambda_1 \cdot P_1 + \lambda_4 \cdot P_5; \\ (\lambda_7 + \lambda_8 + \lambda_{10}) \cdot P_7 = \lambda_6 \cdot P_1 + \lambda_9 \cdot P_5; \\ P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 = 1. \end{cases} \quad (6)$$

Аналогично, как и в предыдущих системах уравнений, исключаем из системы (6) пятое уравнение. Учитывая установленные выше значения коэффициентов при переменных и записав систему уравнений в матричной форме, удобной для решения средствами Excel, получим систему (6) в следующем виде:

$$\begin{cases} 119802 \cdot P_1 - 99999,9 \cdot P_6 - 19801,98 \cdot P_7 - 0,0055 \cdot P_2 = 0; \\ 0,0055 \cdot P_2 - 0,1 \cdot P_6 - 0,022 \cdot P_7 = 0; \\ 0,0055 \cdot P_3 - 0,00000000028 \cdot P_4 - 0,00000000028 \cdot P_5 = 0; \\ 100\,000 \cdot P_4 - 119802 \cdot P_5 = 0; \\ 200\,000 \cdot P_6 - 100\,000 \cdot P_1 - 100\,000 \cdot P_5 = 0; \\ 39604 \cdot P_7 - 19802 \cdot P_1 - 19802 \cdot P_5 = 0; \\ P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 = 1. \end{cases} \quad (7)$$

Самым распространенным способом решения системы линейных уравнений в Excel является матричный метод. Он заключается в построении матрицы из коэффициентов уравнений и создании обратной матрицы. Матрицу заполняем числами, которые являются коэффициентами уравнения. Числа должны располагаться последовательно по порядку с учетом расположения корня, которому они соответствуют. Если в выражении отсутствует корень, то коэффициент считается равным нулю. Если коэффициент не обозначен, но соответствующий корень имеется, то считается, что коэффициент равен единице. Полученную таблицу считаем вектором $\vec{A}$. Далее необходимо получить матрицу, обратную существующей. Для этого можно воспользоваться функцией Excel МОБР. Окно аргументов этой функции и только одно поле – «Массив», в котором нужно указать адрес таблицы. При работе с массивами после завершения ввода формулы следует произвести набор сочетания клавиш Ctrl + Shift + Enter. После этого программа производит вычисления и на выходе в предварительно выделенной области мы имеем матрицу, обратную данной. Далее нужно умножить обратную матрицу на матрицу $\vec{B}$, которая состоит из одного столбца значений, расположенных после знака «равно» в выражениях. Для умножения таблиц в Excel есть функция МУМНОЖ. Активируется окно аргументов функции МУМНОЖ. В поле «Массив1» указываются координаты обратной матрицы. Аналогичное действие проводим для внесения координат в поле «Массив2», выделяя значения колонки $\vec{B}$. После набираем комбинацию клавиш Ctrl + Shift + Enter. В результате в предварительно выделенной ячейке отобразятся корни уравнения. Решение трех систем уравнений, составленных выше для рассмотренных моделей, показано на рис. 7.

ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИРОВАНИЯ

Решение системы уравнений (3) дает следующие значения переменных: $P_1 = 0,090815456$; $P_2 = 0,0000227039$; $P_3 = 0,908154557$; $P_4 = 0,001007284$. Таким образом, вероятность того, что компьютерная система в установившемся состоянии исправно функционирует, равна сумме $P_2 + P_3 + P_4 = 0,909185$. Такая низкая надежность ОС

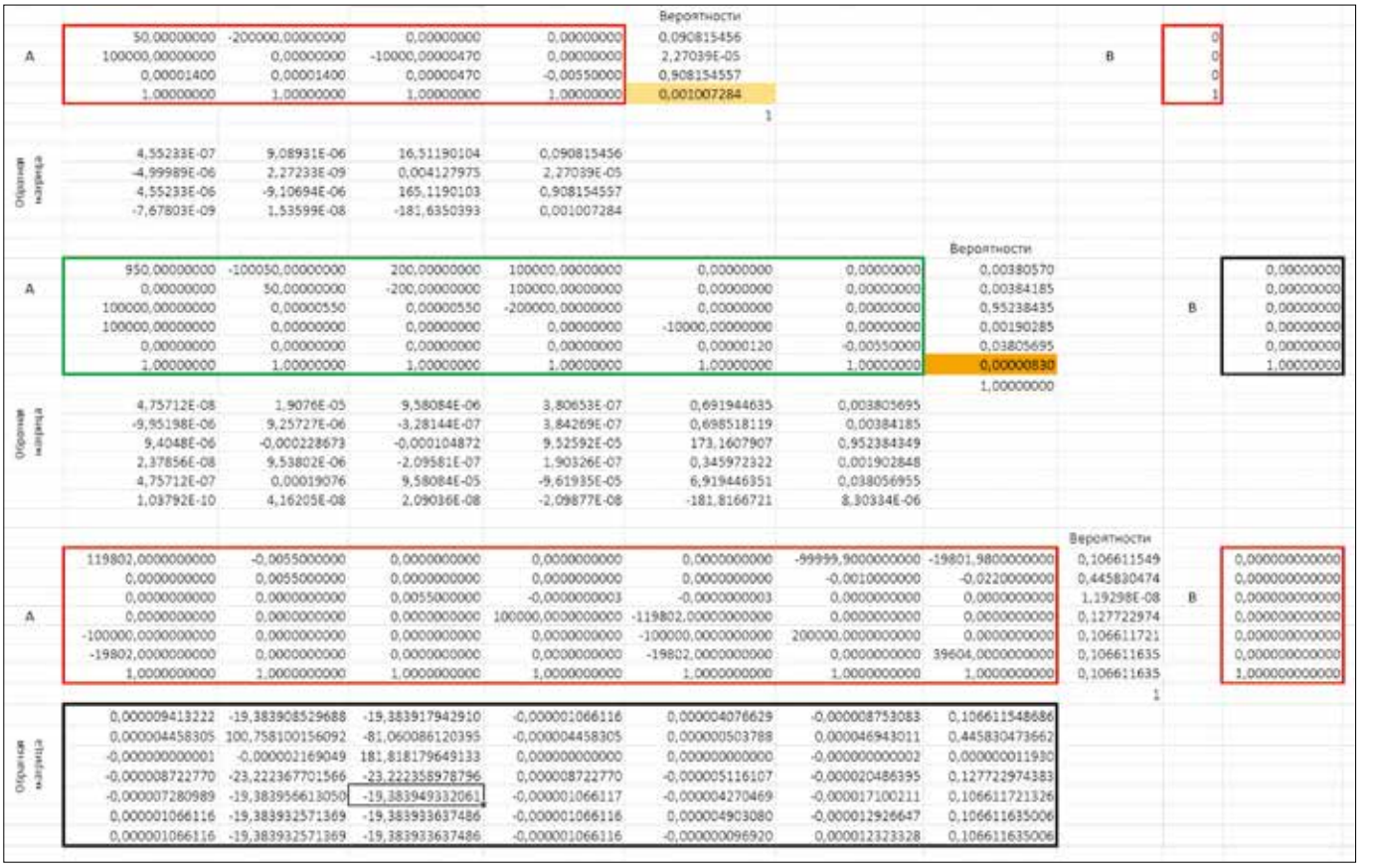

Рис. 7. Решение систем уравнений

с многослойным ядром в нашем случае объясняется заданием в исходных данных большого количества программных ошибок в ядре и той части системы, которая работает в пользовательском режиме (300 000 ошибок). Вероятность нахождения системы в состоянии перезагрузки равна 0,090815456. Ситуация весьма похожа на начальные версии Linux и других операционных систем. В процессе устранения ошибок надежность системы, конечно, повысится, но все-таки не будет достаточно высокой. Радикальный путь – уменьшение ядра ОС.

Решение системы уравнений (6) дает следующие значения переменных: $P_1 = 0,003880570$; $P_2 = 0,00384185$; $P_3 = 0,952238435$; $P_4 = 0,00190285$; $P_5 = 0,3805695$; $P_6 = 0,00000830$. Таким образом, вероятность того, что компьютерная система в установившемся состоянии исправно функционирует, равна сумме $P_1 + P_2 + P_3 + P_4 + P_5 = 0,999534035$. Вероятность того, что система в установившемся состоянии будет находиться на этапе перезагрузки, равна 0,00000830. Таким образом, результаты проведенных вычислений позволяют сделать вывод о значительно более высокой надежности работы микроядерной операционной системы по сравнению с операционной системой с большим многоуровневым модульным ядром. И в заключение рассмотрим решение системы уравнений (7). Оно дает следующие значения переменных: $P_1 = 0,106611549$; $P_2 = 0,445830474$; $P_3 = 0,0000000119298$; $P_4 = 0,127722974$; $P_5 = 0,106611721$; $P_6 = 0,106611635$; $P_7 = 0,106611635$. Заметим, что в случае KasperskyOS, клиент может простаивать, получив вердикт deny (нет на рис. 5) на запрошенную услугу. В случае наших исходных данных вероятность такого простоя $P_2 = 0,445830474$. Изменив интенсивности потоков λ_2, λ_7, которые определяют частоту появления вердикта «да» и λ_2, λ_7, которые определяют частоту появления вердикта «нет», можно получить требуемое значение вероятностей, определяющих работу и простой клиента. При этом операционная система работоспособна, что подтверждается низкой вероятностью ее отказа $P_3 = 0,0000000119298$ в установившемся режиме. Конечно, в реальности (не в модели) вердикт определяется правильностью работы клиента при формировании запросов обращения к услугам сервисов и сервера.

* * *

При решении задач разработки, производства, выбора и эксплуатации программных и операционных систем,

в частности, для исследования критических параметров, таких как надежность, безопасность, защищенность, часто требуется обратиться к методам моделирования, среди которых широкое применение получили методы имитационного, полунатурного и натурного моделирования. Традиционно под моделированием на ЭВМ понималось лишь имитационное моделирование (ИМ). Можно, однако, увидеть, что и при других видах моделирования компьютер может быть весьма полезен, за исключением разве физического моделирования, где компьютер тоже может использоваться, но, скорее, для целей управления процессом моделирования. При математическом моделировании выполнение одного из основных этапов – построение математических моделей по экспериментальным данным просто немыслимо без компьютера.

Имитационное моделирование представляет собой метод конструирования модели реальной системы и постановки экспериментов на этой модели с целью исследования ее поведения, либо оценивания различных стратегий, обеспечивающих функционирование данной системы. Этот метод может быть успешно использован для определения критических параметров архитектур операционных систем, их оценки, сравнения и выбора наиболее подходящей для конкретного применения. При этом необходимо создать структуру исследуемой системы и описать ее поведение с помощью состояний и моментов переходов между этими состояниями. Состояние системы в каждый момент времени можно определить как множество значений ее параметров в этот момент времени. Изменение значений параметров можно считать переходом в другое состояние. Внешняя среда задается посредством входных данных. При необходимости моделирования вероятностных систем и процессов в ИМ включается и статистическое моделирование.

Имитационное моделирование достаточно распространено при исследовании сложных систем благодаря ряду преимуществ. Благодаря ИМ на ранних стадиях предварительного проектирования систем можно быстро получить нужную информацию, пусть и с некоторыми допущениями, о возможном функционировании проектируемой системы. Можно исследовать особенности функционирования системы при любых условиях, в частности тех, которые не могут быть реализованы в натурных экспериментах. При этом параметры системы и окружающей среды можно варьировать в широких границах, воссоздавая произвольные, как реальные, так и гипотетические, ситуации. Главными недостатками метода машинной имитации являются довольно большие затраты времени и средств на построение адекватной модели, а также трудность и даже невозможность учета в модели некоторых важных особенностей реальной системы. По этим причинам машинную имитацию как численный машинный метод решения сложных задач целесообразно применять при следующих условиях:

- непригодность или отсутствие аналитических методов решения задач;
- полная уверенность в успешном создании имитационной модели, которая адекватно описывает исследуемую систему (процесс);
- возможность использовать сам процесс построения имитационной модели для предварительного исследования моделируемой системы.

Натурный эксперимент для определения надежности и безопасности функционирования операционных систем различных архитектур практически невозможен в силу неприемлемых временных и финансовых затрат. Поэтому единственный путь решения этой проблемы – разработка простых и достаточно адекватных моделей.

Предложен подход к построению моделей архитектур операционных систем, основанный на теории марковских процессов, описываемых системами дифференциальных уравнений Колмогорова, в условиях принятия ограничений об абсолютной надежности аппаратуры, случайном характере и независимости отказов в программах без учета их восстановления и отсутствии параллельных процессов в работе модулей ОС. Этот подход может быть использован для начального исследования надежности функционирования конкретных архитектур операционных систем. Нельзя сказать, что такой подход не имеет недостатков. Один из них – ограничение числа возможных состояний в модели системы. При числе состояний более 8–10 усложняется решение систем линейных алгебраических уравнений. Повышение точности результатов моделирования в подобных моделях связано с уточнением исходных данных, которые определяют значения наработки на отказ модулей исследуемой операционной системы и интенсивности переходов исследуемой системы из одного состояния в другое. Для уточнения характеристик надежности модулей системы необходимо построение моделей изменения надежности при выявлении и устранении программных ошибок и соответствующая статистика разработчика операционных систем. Тем не менее, во многих случаях предложенный подход достаточно результативен.

ЛИТЕРАТУРА

1. **Назаров С. В.** Эффективность и оптимизация компьютерных систем: монография / 2-е изд. М.: РУСАЙНС, 2021. 294 с.
2. **Назаров С. В.** Эффективность современных операционных систем // Современные информационные технологии и ИТ-образование. 2017. Т. 13 № 2. С. 9–24.
3. **Назаров С. В., Вилкова Н. Н.** Эффективность систем отображения информации коллективного пользования // Электросвязь. 2015. № 9. С. 29–33.

4. **Назаров С. В., Вилкова Н. Н.** Выбор оптимального варианта системы отображения информации коллективного пользования // Электросвязь. 2017. № 1. С. 60–65.
5. **Таненбаум Э., Вудхалл А.** Операционные системы. Разработка и реализация / 3-е изд. СПб: Питер, 2007. 704 с.
6. **Назаров С. В.** Архитектура и проектирование программных систем: монография / 2-е изд., перераб. и доп. М.: ИНФРА-М, 2016. 376 с.
7. **Таненбаум Э., Хердер Дж., Бос Х.** Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств. [Электронный ресурс]. URL: http://citforum.ru/operating_systems/microkernel_tanenbaum/
8. **Назаров С. В., Широков А. И.** Технологии многопользовательских операционных систем. М.: Изд. дом МИСиС, 2012. 296 с.
9. **Назаров С. В., Вилкова Н. Н.** Структурный рефакторинг многослойных программных систем // Информационные технологии и вычислительные системы. 2016. № 4. С. 13–23.
10. **Tanenbaum Andrew S., Herder Jorrit N., Bos Herbert.** Vrije Universiteit, Amsterdam. Can We Make Operating Systems Reliable and Secure? Computer (IEEE Computer Society, V. 39, No 5, May 2006).
11. **Хердер Й., Бос Х., Таненбаум Э.** Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств, 2006. [Электронный ресурс]. URL: http://citforum.ru/operating_systems/reliable_os/
12. **Tanenbaum A.** Introduction to MINIX 3 // OS News is Exploring the Future of Computing. 2006. [Электронный ресурс]. URL: http://www.osnews.com/story/15960/
13. **Игнатов Р.** MINIX 3 – реинкарнация? [Электронный ресурс]. URL: http://www.minix3.ru/articles/Ignatov-_minix_reincarnation.pdf
14. **Таненбаум Э., Бос Э.** Современные операционные системы // 4-е изд. СПб: Питер, 2015. 1120 с.
15. **Кельберт М. Я., Сухов Ю. М.** Вероятность и статистика в примерах и задачах. Т. 2. Марковские цепи как отправная точка теории случайных процессов и их приложения. М.: МЦНМО, 2009. 476 с.
16. **Кемени Дж., Снелл Дж.** Конечные цепи Маркова. М.: Наука, 1970. 273 с.
17. **Блинов А.** Лаборатория Касперского – об основах кибериммунитета и концепции KasperskyOS. [Электронный ресурс]. URL: https://spbit.ru/news/Laboratoriya-Kasperskogo-ob-osnovakh-kiberimmuniteta
18. Архитектура KasperskyOS [Электронный ресурс]. URL: https://support.kaspersky.ru/help/KCE/1.1/ru-RU/overview_architecture.htm
19. Микроядро KasperskyOS. [Электронный ресурс]. URL: https://os.kaspersky.ru/technologies/microkernel/?ysclid=ll3ego7wh4870216945

Векторные анализаторы цепей

АКИП-6604 и АКИП-6605

Экономичное решение большинства задач

Модель	Порты	Частотный диапазон	Динамический диапазон	Выходная мощность
АКИП-6604/1	2	9 кГц…4,5 ГГц	125 дБ	- 55 дБм … 10 дБм
АКИП-6604/2	4	9 кГц…4,5 ГГц	125 дБ	- 55 дБм … 10 дБм
АКИП-6604/3	2	9 кГц…8,5 ГГц	125 дБ	- 55 дБм … 10 дБм
АКИП-6604/4	4	9 кГц…8,5 ГГц	125 дБ	- 55 дБм … 10 дБм
АКИП-6605/1	2	100 кГц…13,5 ГГц	125 дБ	- 55 дБм … 10 дБм
АКИП-6605/2	2	100 кГц…26,5 ГГц	125 дБ	- 55 дБм … 10 дБм

Опционально доступны возможности:

- Анализатора спектра в полном диапазоне частот
- Рефлектометра и дифференциальные пробники до 26,5 ГГц
- Анализа во временной области
- Режим импульсных измерений, сегментированный режим
- Свипирование по частоте и уровню
- Модули электронной калибровки и механические калибровочные комплекты
- Кабели с NMD разъёмами

У ТОЧНОСТИ ЕСТЬ ИМЯ! ПРИСТ

111141, **г. Москва**, ул. Плеханова 15а; тел.: +7 (495) 777-5591; факс: +7 (495) 640-3023
196006, **г. Санкт-Петербург**, ул. Цветочная, д. 18, лит. В, офис 202; тел./факс: +7 (812) 677-7508
620089, **г. Екатеринбург**, ул. Цвиллинга, д. 58, офис 1; тел./факс: +7 (343) 317-3999; ek@prist.ru

prist.ru