УДК 004.773.5
doi:10.15217/issn1684-8853.2016.3.15

АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ В КОНТРОЛИРУЕМЫХ АККАУНТАХ СИСТЕМЫ ВИДЕОКОНФЕРЕНЦСВЯЗИ

А. И. Савельев[a, 1], *научный сотрудник*
[a]*Санкт-Петербургский институт информатики и автоматизации РАН, Санкт-Петербург, РФ*

Цель: *анализ алгоритмов обработки данных в контролируемых аккаунтах системы видеоконференцсвязи и их сложности. Оценка влияния внедрения нового модуля в приложение на его производительность.* ***Результаты:*** *анализ сложности алгоритмов и подсчет количества новых функций и графических интерфейсов, необходимых для внедрения дополнительного модуля в пиринговое приложение видеоконференцсвязи, показал, что новый модуль незначительно влияет на производительность приложения в целом. Высокая производительность модуля достигается за счет линейной сложности его алгоритмов обработки данных, работа которых потребляет малое количество ресурсов процессора и оперативной памяти компьютера по сравнению с работой остальных частей пирингового приложения видеоконференцсвязи в клиентской части приложения.* ***Практическая значимость:*** *модуль контролируемых аккаунтов и его алгоритмы обработки данных позволяют расширить возможности приложения видеоконференцсвязи. Линейная сложность алгоритмов обработки данных обеспечивает высокую производительность данного модуля, что положительно сказывается на работе приложения в целом. Алгоритмы обработки данных построены на базе существующих в приложении видеоконференцсвязи алгоритмов или копируют их, обеспечивая простую интеграцию модуля в архитектуру приложения, а также облегчая задачу разработки программного обеспечения.*

Ключевые слова *— пиринговое взаимодействие, клиент-серверное взаимодействие, видеоконференцсвязь, модульная архитектура программного обеспечения.*

Введение

Современные приложения видеоконференцсвязи позволяют обмениваться не только аудио- и видеопотоками, но и совместно редактировать документы, рисовать схемы, графики, изображения [1]. Личный кабинет пользователя такого приложения содержит настройки и графические компоненты интерфейса, обеспечивающие взаимодействие с другими пользователями. Развитие инфотелекоммуникационных технологий повлияло на развитие пиринговых сетей, которые осуществляют передачу данных напрямую между пользователями, тем самым снижая нагрузку на серверную часть приложения и распределяя ее между клиентами [2]. Данный подход позволяет создавать небольшие видеочаты до десяти человек, где хорошее качество связи обеспечивается максимум при пяти участниках. Пиринговые приложения видеоконференцсвязи направлены на обеспечение обмена данными в узком кругу участников, но они также востребованы, как и их клиент-серверные аналоги, а значит, их разработка и развитие являются перспективными и целесообразными [3, 4].

Управление числом активных участников в одном сеансе видеоконференцсвязи осуществляется функционалом клиентских приложений, настраиваемых и взаимодействующих с центральным сервером. Рассмотрим подходы к управлению пользовательскими аккаунтами и средства многомодальных интерфейсов, применяемых в архитектурах видеоконференцсвязи.

Одним из востребованных направлений в развитии модулей для приложений видеоконференцсвязи является разработка корпоративных аккаунтов. В статье [5] рассматривается управление аккаунтами пользователей корпоративной сети. Для решения задач управления аккаунтами используются несколько технологий, разработанных в Carnegie Mellon University, предназначенных для облегчения работы с пользователями и их правами в корпоративной среде. Информация обычно обрабатывается в дереве LDAP (Lightweight Directory Access Protocol) серверов, поддерживаемых различными группами на различных административных уровнях — от корпорации до отдела. Для оптимизации информации, проходящей через уровни LDAP-серверов, проводится процедура минимизации необходимых связей между поддерживающими группами и используются механизмы кэша для повторно передаваемой информации с высших административных уровней на низшие. Эти технологии могут быть применены как на различных уровнях LDAP-серверов, так и на компьютере конечного пользователя.

Управление аккаунтами в среде гибридного облака рассматривается в работе [6]. Для решения задач контроля аккаунтов используется система,

[1] Научный руководитель — профессор, доктор технических наук, заместитель директора по научной работе Санкт-Петербургского института информатики и автоматизации РАН *А. Л. Ронжин*.

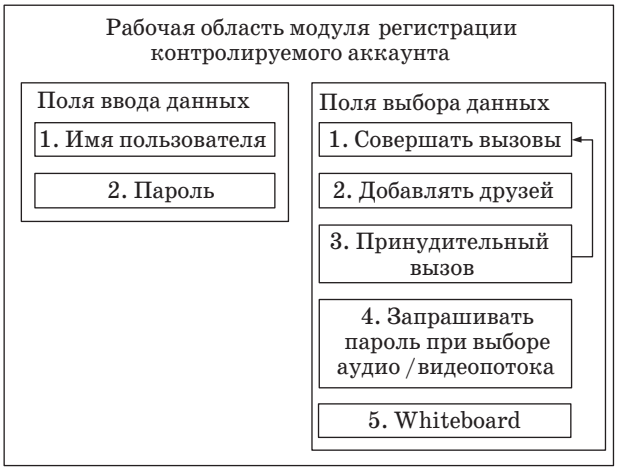
управляющая облаком Monsoon. Данная система предоставляет корпоративным пользователям интерфейс и портал для управления инфраструктурой облака во множественных публичных и приватных облачных провайдерах. Она обеспечивает управление пользовательскими аккаунтами, контроль доступа, инструменты анализа и отчетов, корпоративные аккаунты ролей и систему, осуществляющую контроль, поддерживающую пользовательские подписки и управление многоуровневыми аккаунтами в гибридном облаке, которое может содержать множество публичных и приватных облачных сервисов.

Немаловажным аспектом при разработке модулей является сама архитектура приложения. Правильно спроектированная архитектура позволяет легко разрабатывать и внедрять новые модули. В работе [7] описывается системная архитектура видеоконференции на базе рекомендации ITU-T стандарта H.323, рассматривается как серверная, так и клиентская часть. Данная архитектура проводит обработку видео для создания эффекта личного присутствия посредством управления объектами видео и динамического распределения скоростей. В работе описан подход, основанный на цветности ключа, разрешающий извлечения объектов сервером PPMCU так, что видеообъектами можно управлять в соответствии с потребностями пользователей. После извлечения объектов сервер перекодирует их, используя интеллектуальный метод распределения битов, и вставляет обратно фон цветового ключа. Клиент извлекает видео из принятого потока и манипулирует 2D-объектами и виртуальным фоном в 3D-пространстве.

Функции и алгоритмы модуля контролируемого аккаунта

В данной статье рассматриваются алгоритмы обработки данных модуля, внедряемого в уже существующее приложение видеоконференцсвязи, на примере контролируемых личных кабинетов (далее — контролируемых аккаунтов). Работа контролируемых аккаунтов взаимосвязана с алгоритмами работы приложения видеоконференцсвязи, и аккаунты являются частью приложения, архитектура которого представлена в работе [8].

Графический интерфейс для регистрации контролируемого аккаунта доступен только из зарегистрированного обычного аккаунта приложения. Схема графического интерфейса (рис. 1) позволяет пользователю осуществить регистрацию нового имени пользователя (логина), пароля, а также задать права доступа для управления контролируемым аккаунтом. С целью выявить необходимые данные для внедрения проведем сравнение обычного и контролируемого аккаунтов относительно возможностей контролируемого аккаунта.

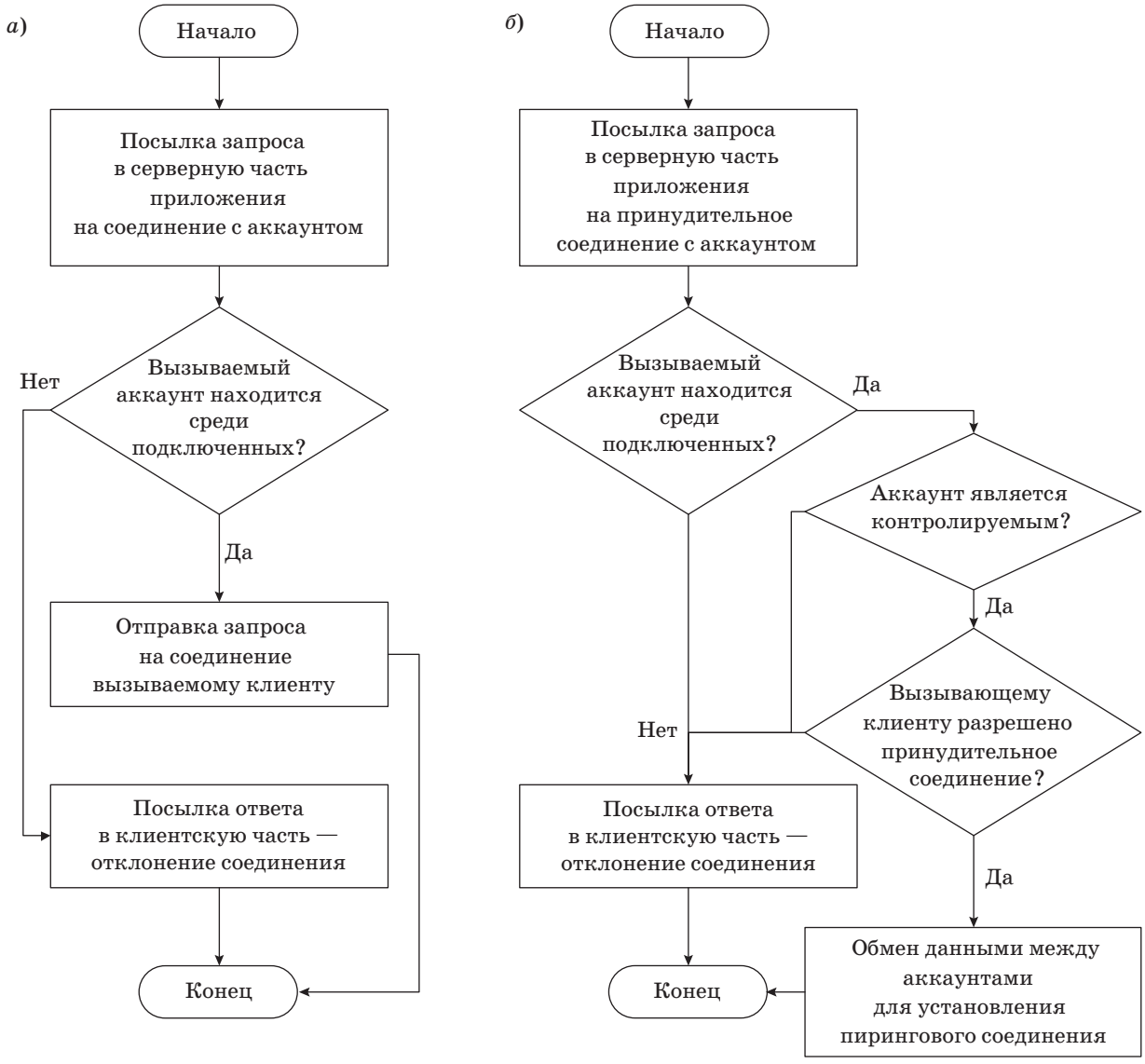


■ ***Рис. 1***. Схема графического интерфейса регистрации контролируемого аккаунта

В области полей выбора (см. рис. 1) существует несколько различных параметров для управления контролируемым аккаунтом. Блокировка возможности совершать вызовы запрещает контролируемому аккаунту самому устанавливать соединение с другими пользователями приложения. При выборе данного пункта приложение автоматически будет разрешать ряду пользователей, выбранных администратором аккаунта, принудительно соединяться с данным аккаунтом, что соответствует третьему пункту полей выбора данных на схеме графического интерфейса. На рис. 1 первый и третий пункты полей выбора соединены стрелкой, что показывает их зависимость друг от друга; при выборе третьего пункта первый пункт будет автоматически заблокирован. Взаимодействие между клиентской и серверной частями всех рассматриваемых далее алгоритмов осуществляется по протоколу WebSocket. Алгоритмы обычного соединения аккаунтов и принудительного соединения с контролируемым аккаунтом представлены на рис. 2, *а* и *б* соответственно.

Данные алгоритмы очень схожи, за исключением нескольких дополнительных проверок, присутствующих в алгоритме принудительного соединения, и последнего этапа работы алгоритмов при удачном выполнении всех проверок. Сначала более подробно рассмотрим алгоритм принудительного соединения и его сложность, так как он содержит в себе большую часть алгоритма обычного соединения аккаунтов.

Данный алгоритм направляет текстовый идентификатор запроса и имени аккаунта из клиентской части приложения в серверную. Сложность этой операции равна $C1$ и является постоянной. Серверная часть приложения опознает идентификатор запроса и имени аккаунта из клиентской части приложения в серверную. Сложность этой операции равна $C1$ и является постоянной. Серверная часть приложения опознает иденти-



■ ***Рис. 2.*** Алгоритм обычного соединения аккаунтов (*а*) и принудительного соединения с аккаунтом (*б*)

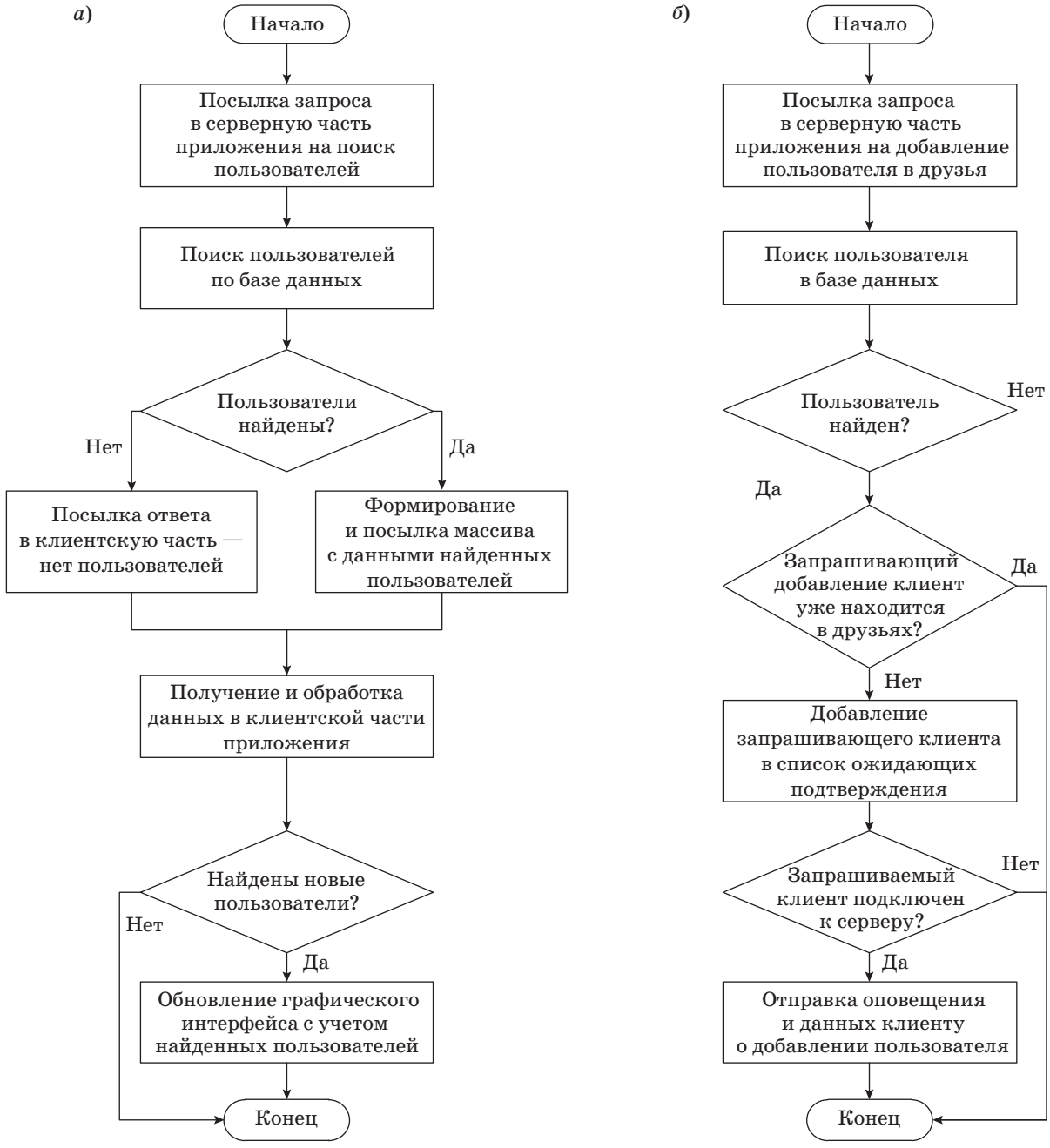
фикатор запроса и запускает соответствующий ему процесс обработки данных, который попытается найти интересующего клиента по идентификатору имени среди списка подключенных к серверу аккаунтов, в худшем случае содержащему все n пользователей базы данных. Если клиент найден, производится проверка, находится ли пользователь в друзьях. В худшем случае сложность этой проверки равна n. Если пользователя в друзьях нет, он добавляется в список ожидающих подтверждения, после чего производится проверка, подключен ли пользователь к серверу. При положительном результате трех вышерассмотренных проверок между клиентами будет произведен обмен данными для осуществления пирингового соединения. В случае отрицательного исхода одной из проверок сервер отправит клиенту сообщение об отклонении запроса на соединение. Эта операция имеет постоянную сложность и равна $C2$. В итоге выражение, описывающее сложность алгоритма, выглядит как $2n+C1+C2=\theta(n)$, поэтому можно сказать, что сложность алгоритма линейна.

В алгоритме обычного соединения аккаунтов выполняется всего одна проверка, при положительном результате которой будет выполнен не обмен данными для соединения аккаунтов, как в случае с контролируемым аккаунтом, а отправление запроса вызываемому клиенту. Это отличие является очень важным, так как при обычном соединении аккаунтов оба аккаунта являются равноправными, и вызываемый аккаунт при получении запроса сможет как принять, так и отклонить его. В случае с контролируемым

аккаунтом соединение происходит принудительно и никакого подтверждения о соединении с вызываемым аккаунтом не требуется, если для этого выполнены все рассмотренные выше предпосылки. Если рассматривать вопрос сложности, то можно сказать, что изменения в алгоритме не сильно влияют на его сложность. Она описывается линейным выражением $2n+C1+C2=\theta(n)$, так же, как и в алгоритме принудительного соединения.

При выборе пункта графического меню «Добавлять друзей» (см. рис. 1) контролируемый аккаунт сможет осуществлять поиск пользователей в базе данных и отправлять заявки на добавление их в список друзей. Алгоритмы поиска пользователей и добавления пользователей в друзья представлены на рис. 3, *а* и *б* соответственно.

Два эти алгоритма работают одинаково для всех типов аккаунтов, ограничения возникают только при отключении работы данных алгоритмов в контролируемых аккаунтах. Далее рассмотрим подробнее этапы их выполнения и вычислительную сложность.



■ ***Рис. 3.*** Алгоритм поиска пользователей (*а*) и добавления пользователя в друзья (*б*)

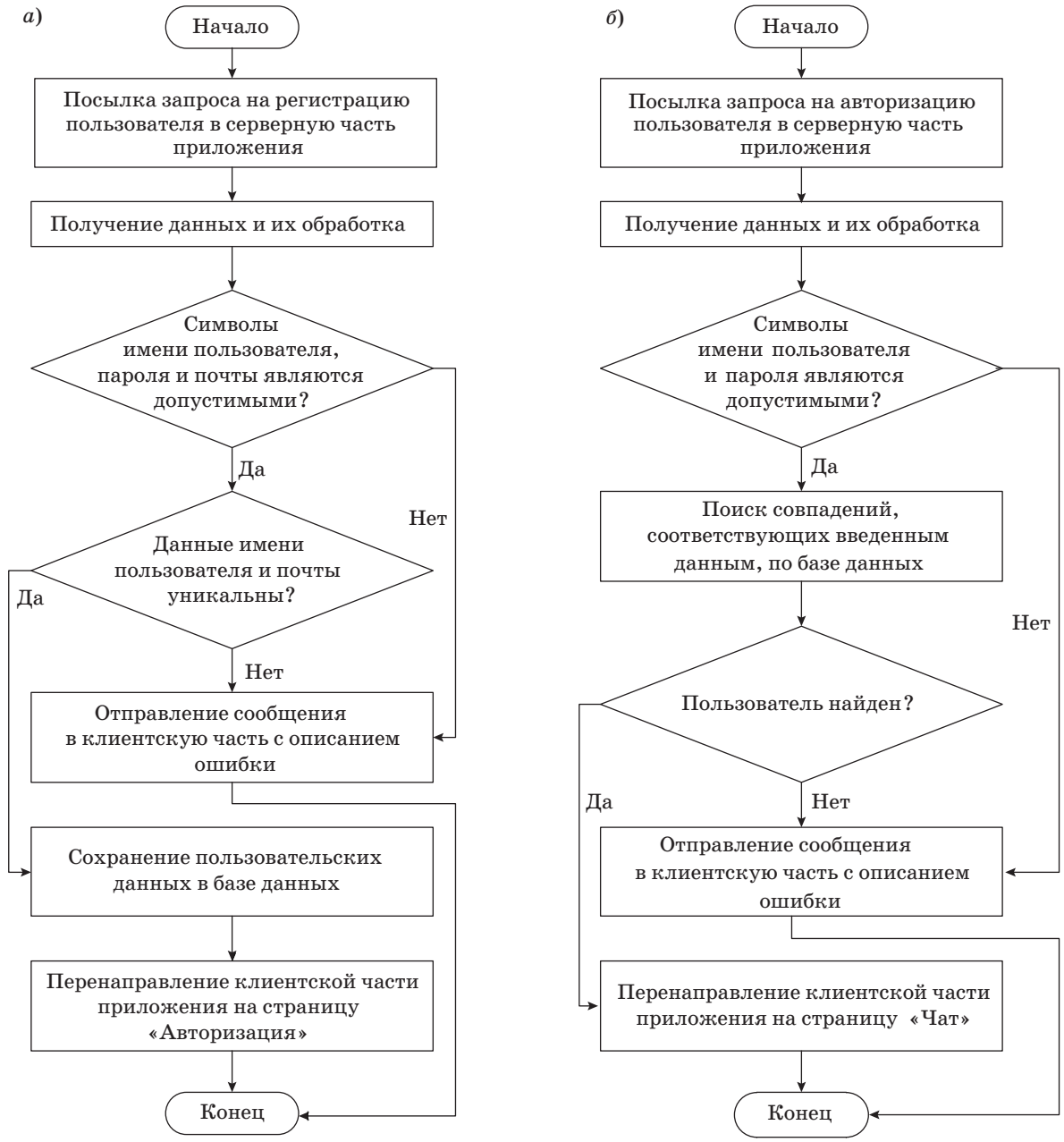
Алгоритм поиска пользователей начинает свою работу с отправления запроса из клиентской части приложения в серверную. Сложность этой операции равна $C1$ и является постоянной. Запрос представляет собой строку, содержащую целое или часть имени пользователя. Далее сервер создаст запрос в базу данных, содержащую n пользователей, с данными из переданной строки и получит результат. В случае возврата пустого результата сервер отправит клиенту информационное сообщение, в противном случае сервер отправит найденные данные в виде массива найденных пользователей. Независимо от варианта развития событий сложность этой конструкции можно оценить как постоянную, равную $C2$. Клиентская часть приложения обработает полученные данные; в случае если новые данные отличаются от тех, что уже есть в хранилище клиентской части приложения, выполнится поиск новых пользователей со сложностью, в худшем случае равный n. Далее произойдет перерисовка графического интерфейса на основе полученных данных, сложность которой постоянна и равна $C3$. В результате сложность всего алгоритма описывается выражением $2n+C1+C2+C3=\theta(n)$. Таким образом, сложность алгоритма линейна.

Алгоритм добавления пользователя в друзья начинает свою работу с посылки строки в серверную часть приложения с именем пользователя, которого необходимо добавить в друзья. Сложность этой операции, как и операции из предыдущего алгоритма, постоянна и равна $C1$. Далее серверная часть приложения выполнит поиск пользователя по базе данных, содержащей n пользователей, и в случае если пользователь найден (сложность проверки постоянна и равна $C2$), проверит наличие его в друзьях найденного аккаунта. В худшем случае сложность этой проверки равна n. Если запрашиваемый аккаунт уже находится в друзьях, то алгоритм завершит свою работу, если нет — добавит найденного пользователя в список ожидающих подтверждения запроса на добавление в друзья. Последним шагом алгоритма будет проверка, подключен ли аккаунт найденного пользователя к серверу или нет. В случае если аккаунт подключен, сервер отправит ему данные о новых заявках в друзья. Данные операции имеют сложность n. В итоге выражение, описывающее сложность алгоритма, выглядит как $3n+C1+C2=\theta(n)$, поэтому можно сказать, что сложность алгоритма линейна.

Примерно 30 % операций во всех вышеприведенных алгоритмах занимают различные проверки, которые заменяются обычным блокированием возможностей графического интерфейса в клиентской части приложения. Но ограничение графического интерфейса клиентской части приложения не гарантирует, что пользователь умышленно с помощью вызова определенных команд не попытается воспользоваться возможностями, которые заблокированы в графическом интерфейсе и на программном уровне. Также невозможно быть полностью уверенным, что при обфускации клиентского кода его не смогут расшифровать и использовать в собственных целях для совершения XSS-атак на сервер. Поэтому для безопасности серверной части приложения, обеспечения конфиденциальности информации пользователей и разграничения доступа к функционалу приложения используются различные проверки в серверной части приложения.

В любых типах аккаунтов возможность включения и выключения захвата аудио- и видеопотоков с соответствующих устройств доступна по умолчанию. В контролируемых аккаунтах пункт настроек «Запрашивать пароль при выборе аудио/видеопотока» (см. рис. 1) позволяет ограничить доступ к управлению устройствами посредством логина и пароля. Логин и пароль хранятся в серверной части приложения, и при каждой попытке доступа к устройствам клиентская часть приложения будет запрашивать ввод логина и пароля для отправки его в серверную часть приложения и дальнейшей проверки. Последний пункт графического интерфейса, так же как и предыдущий, оперирует логином и паролем, в данном случае они используются для ограничения доступа к выходу из профиля пользователя. Логин и пароль в приложении впервые используются при регистрации пользователя, затем следует авторизация; алгоритмы данных процессов представлены на рис. 4, *а* и *б*. Все последующие операции для подтверждения логина и пароля аккаунта используют тот же алгоритм авторизации, что и в начале работы приложения, но данные в серверную часть пересылаются через протокол WebSocket.

Алгоритм регистрации пользователя начинает свою работу с посылки запроса на регистрацию в серверную часть приложения. Запрос содержит в себе данные имени пользователя, пароля и почты клиента. Сервер проверит поступившие данные на соответствие допустимому набору символов. Сложность этих операций является постоянной и равна $C1$. Затем следует проверка на уникальность почты и имени пользователя со сложностью n, где n — число пользователей. В случае если данные не прошли одну из проверок, сервер сформирует сообщение с описанием некорректных данных и отправит его в клиентскую часть. В случае если все введенные пользователем данные корректны, сервер сохранит их в базу данных и сформирует ответ на запрос клиента с перенаправлением на страницу авторизации. Сложность этих операций оценивается как постоянная и равна $C2$. В результате сложность алгоритма описывается выражением $n+C1+C2=\theta(n)$, из чего следует, что алгоритм имеет линейную сложность.



■ ***Рис. 4.*** Алгоритм регистрации (*а*) и авторизации (*б*) пользователя

Алгоритм авторизации посылает запрос из клиентской части приложения с именем пользователя и паролем в серверную. Сервер проверит поступившие данные на соответствие допустимому набору символов. Сложность этих операций является постоянной и равна $C1$. Затем сервер попытается найти пользователя в базе данных с n пользователями. Если данные, отправленные клиентом, некорректны, сервер сформирует сообщение с описанием некорректных данных и отправит его в клиентскую часть. В случае успешного завершения проверок сервер перенаправит клиента на страницу-чат. Сложность этих операций оценивается как постоянная, равная $C2$. После завершения авторизации сервер отправит клиенту cookie-данные. Cookie-данные будут идентифицировать клиента всякий раз, когда он будет пытаться отправить новые запросы на сервер, не требуя от него повторного ввода логина и пароля, кроме ситуаций с контролируемыми аккаунтами, описанными выше. В результате сложность алгоритма описывается выражением $n+C1+C2=\theta(n)$, из чего следует, что алгоритм имеет линейную сложность.

Регистрация контролируемых аккаунтов проходит по тому же алгоритму, что и регистрация обычных аккаунтов, но существуют некоторые

различия. Регистрация контролируемого аккаунта может быть осуществлена только со страницы-чата зарегистрированным и авторизированным пользователем. Второе различие заключается в протоколе передачи данных: при обычной регистрации аккаунта используется протокол HTTP, при регистрации контролируемого аккаунта используется протокол WebSocket, который способствует ускорению идентификации пользователя на сервере, упрощению процесса приема и передачи данных между сервером и клиентом. Последнее различие заключается в количестве данных, отправляемых при регистрации аккаунта; при регистрации контролируемого аккаунта используются только логин и пароль, в то время как при регистрации обычного аккаунта требуется еще и адрес электронной почты. Данное различие возникает по причине того, что контролируемый аккаунт позволяет управлять им с помощью обычного аккаунта, на который был зарегистрирован управляемый аккаунт, поэтому почта у таких аккаунтов является общей.

Контролируемые аккаунты в архитектурном плане отличаются от обычных только взаимодействием с сервером. В случае с пиринговым соединением и передачей мультимедийных данных они подобны обычным аккаунтам. Далее рассмотрим их графический интерфейс.

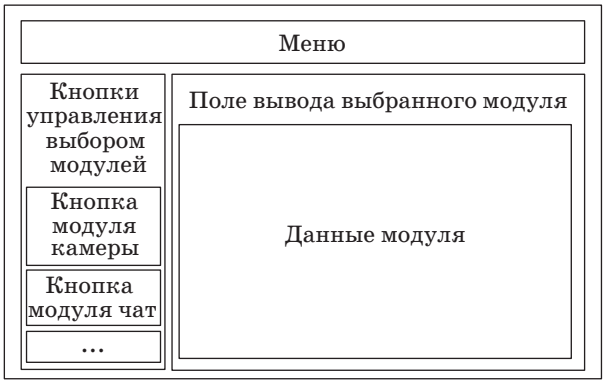
Оценка новых графических интерфейсов для модуля контролируемого аккаунта

Несмотря на линейную сложность алгоритмов и схожесть либо идентичность некоторых из них, необходимо учитывать сложность проектирования графических интерфейсов приложения. Графические интерфейсы позволяют осуществлять взаимодействие между пользователями и клиентской частью приложения, запуская на выполнение различные алгоритмы.

Для внедрения нового графического интерфейса в приложение требуется разработать ряд функций, которые будут связывать его с хранилищем данных, и ряд функций, которые связывают хранилище и конечные модули приложения. Количество функций, связывающих хранилище данных и модули, не всегда линейно зависит от количества данных, полученных из графического интерфейса, поэтому рассмотрим конкретные случаи внедряемых модулей.

Схема графического интерфейса модулей обычного и контролируемого аккаунтов представлена на рис. 5.

Данный графический интерфейс полностью идентичен для обоих аккаунтов, за исключением ситуаций, когда контролируемый аккаунт имеет ограничения в настройках и не содержит некоторых элементов интерфейса. При добавлении модуля контролируемого аккаунта в приложение



■ ***Рис. 5.*** Схема графического интерфейса аккаунта

в обычном аккаунте появляется новый графический элемент, позволяющий управлять настройками контролируемого аккаунта. Новый элемент содержит в себе восемь различных настроек, представляющих шесть полей выбора и два поля ввода данных. Все перечисленные поля тесно взаимодействуют с логикой приложения, формируя данные для сервера. Взаимодействие с каждым графическим элементом настроек и полями ввода данных описывается функцией, позволяющей передавать данные в хранилища, находящиеся в клиентской части приложения. В клиентской части приложения существует несколько типов хранилищ, которые отвечают за различные модули и типы данных. При изменении данных в хранилищах все модули, которые взаимодействуют с ними, получают оповещения, что данные изменились. Модуль при получении оповещения об изменении данных запустит алгоритмы, необходимые для их обработки. В общей сумме имеется 16 новых функций, из которых восемь взаимодействуют с хранилищем и восемь служат для оповещения модулей о дальнейшей обработке. Такой подход оправдан, так как перед отправлением на сервер данные проходят предварительную обработку в клиентской части приложения. Обработка подразумевает различные проверки на соответствие определенному набору и количеству символов и взаимодействие связанных друг с другом пунктов. Контролируемый аккаунт в клиентской части приложения полностью заимствует элементы графического интерфейса обычных аккаунтов, таким образом, сложность приложения не возрастает.

Оценка влияния модуля контролируемого аккаунта на производительность приложения

Несмотря на различные задачи, которые должны выполнять модули и каналы связи, передающие необходимые данные, сложность внедрения модулей в данное пиринговое приложение видео-

конференцсвязи примерно одинакова. Это достигается за счет проектирования такой архитектуры приложения, которая позволяет добиться однозначности при интеграции модулей, использовать имеющиеся алгоритмы, незначительно модернизировать их или разрабатывать новые с линейной сложностью. Благодаря подобной архитектуре упрощается поддержка и развитие приложения. Все алгоритмы, необходимые для внедрения модулей в приложение, имеют линейную сложность, что обеспечивает высокую скорость обмена данными между модулями и самим приложением. Наконец, очевидное достоинство разбиения приложения на отдельные модули заключается в том, что каждый элемент приложения является целостным, не зависимым напрямую от остальных и легко поддается изменению, не затрагивая другие части системы.

Для внедрения модуля контролируемого аккаунта необходимо разработать шестнадцать новых функций, модернизировать два алгоритма и привязать модуль к четырем уже существующим алгоритмам; также необходимо разработать новый графический интерфейс, состоящий из восьми элементов. Все алгоритмы являются линейными и частично или полностью заимствованными, таким образом, их работа не влияет на производительность приложения. При проведении тестирования в клиентской части были получены данные о работе приложения в различных режимах. При запуске системы клиентская часть приложения потребляет примерно 3500 КБ оперативной памяти (ОП) и 1–2 % мощности центрального процессора (ЦП). Во время пирингового соединения с одним клиентом потребление составило примерно 10 МБ ОП и 5–6 % мощности ЦП. Последний этап тестирования показал, что при четырех пиринговых соединениях потребление ресурсов составило 300 МБ ОП и 25–30 % мощности ЦП. Модуль контролируемого аккаунта при всех режимах работы приложения потребляет от 2 до 10 КБ ОП и 0,01–0,02 % мощности ЦП, что вносит незначительный вклад в потребление ресурсов клиентской частью приложения видеоконференцсвязи. Таким образом, можно судить о высокой эффективности и быстродействии данного модуля.

Заключение

Все вышеприведенные данные обеспечивают полную интеграцию модуля контролируемого аккаунта в приложение видеоконференцсвязи. Конечная сложность внедрения модуля является достаточно низкой и обеспечивает простоту и надежность интеграции. При такой невысокой сложности можно утверждать, что алгоритмы и функции, связывающие ядро приложения и модуль, потребляют очень мало ресурсов оперативной памяти и процессорного времени, благодаря чему работа такой системы не влияет на производительность работы всего приложения.

В целом внедряемые модули хорошо адаптируемы в сложных системах с минимальным клиент-серверным взаимодействием [8], где основная часть нагрузки распределяется между клиентскими частями приложения. Также работа модулей не нарушает целостность данных, обрабатываемых другими частями приложения, поэтому их можно использовать с алгоритмами и средствами передачи мультимедийных потоков данных [9, 10].

Работа выполнена при частичной поддержке гранта РФФИ 15-07-06774.

Литература

1. **Hsiao K. L., Chiang H. S., Huang C. T.** Continuous Usage Intention of Videoconferencing Software: A Case Study of One-to-Some Online Courses // Advanced Information Networking and Applications Workshops (WAINA). 2013. P. 990–995. doi:10.1109/WAINA.2013.24
2. **Civanlar M. R., Ozkasap O., Celebi T.** Peer-to-peer Multipoint Videoconferencing on the Internet // Signal Processing: Image Communication. 2005. Iss. 20. N 8. P. 743–754. doi:10.1109/ICIP.2004.1421484
3. **Ramdan A. A., Munir R.** Selective Encryption Algorithm Implementation for Video Call on Skype Client // 7th Intern. Conf. on Telecommunication Systems, Services, and Applications (TSSA). 2012. P. 120–124. doi:10.1109/TSSA.2012.6366035
4. **Xu Y., Yu C., Li J., Liu Y.** Video Telephony for End-Consumers: Measurement Study of Google+, iChat, and Skype // IEEE/ACM Transactions on Networking. 2014. Vol. 22. P. 826–839. doi:10.1109/TNET.2013.2260354
5. **Manolache F. B., McDowell W., Rusu O.** Directory Cache Techniques for Efficient User Management // 10th Roedunet Intern. Conf. (RoEduNet). 2011. P. 1–5. doi:10.1109/RoEduNet.2011.5993696
6. **Yan S.**, et al. Infrastructure Management of Hybrid Cloud for Enterprise User // 5th Intern. DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM). 2011. P. 1–6. doi:10.1109/SVM.2011.6096463
7. **Lin C. W.**, et al. Implementation of H.323 Multipoint Video Conference Systems with Personal Presence Control // Intern. Conf. on Consumer Electronics (ICCE). 2000. P. 108–109. doi:10.1109/ICCE.2000.854518
8. **Saveliev A. I., Vatamaniuk I. V., Ronzhin An. L.** Architecture of Data Exchange with Minimal Client-Server Interaction at Multipoint Video Conferenc-

ing / Ed. by S. Balandin, et al. // LNCS. 2014. Vol. 8638. P. 164–176. doi:10.1007/978-3-319-10353-2
9. **Saveliev A. I., Ronzhin A. L.** Algorithms and Software Tools for Distribution of Multimedia Data Streams in Client Server Videoconferencing Applications // Pattern Recognition and Image Analysis. 2015. Iss. 25. N 3. P. 517–525.
10. **Савельев А. И.** Управление передачей аудиовизуальных потоков в системе видеоконференцсвязи // Тр. Междунар. студ. конф. по автоматизации и управлению (ISCAC-2013). 2013. С. 107–111.

UDC 004.773.5
doi:10.15217/issn1684-8853.2016.3.15

Algorithms of Data Processing in Supervised Accounts of a Videoconferencing System

Saveliev A. I.[a], Researcher, saveliev@iias.spb.su
[a]Saint-Petersburg Institute for Informatics and Automation of RAS, 39, 14 Line V. O., 199178, Saint-Petersburg, Russian Federation

Purpose: Analyzing algorithms of data processing in supervised accounts of a videoconferencing system; estimating their complexity; evaluation of how introducing a new module into the application affects its performance. **Results:** The analysis of algorithm complexity and the estimation of the number of new features and graphic interfaces required for introducing an additional module into a peering videoconferencing application have shown that the new module only negligibly affects the overall application performance. Such an effect was achieved due to the linear complexity of the implemented algorithms which consume a small amount of CPU and RAM resources as compared to the functioning of other parts of the application. **Practical relevance:** The module of supervised accounts and its data processing algorithms make it possible to extend the possibilities of the videoconferencing application. The linear complexity of the data processing algorithms provides high performance of the module, which has a positive effect on the application overall. The data processing algorithms are based on the existing videoconferencing application algorithms, or just copy them. This provides simple integration of the module into the application architecture, and facilitates the task of software development.

Keywords — Peering Interaction, Client-Server Interaction, Videoconferencing, Modular Architecture of Software.

References

1. Hsiao K. L., Chiang H. S., Huang C. T. Continuous Usage Intention of Videoconferencing Software: A Case Study of One-to-Some Online Courses. *27th Intern. Conf. on Advanced Information Networking and Applications Workshops (WAINA)*, 2013, pp. 990–995. doi:10.1109/WAINA.2013.24
2. Civanlar M. R., Ozkasap O., Celebi T. Peer-to-Peer Multipoint Videoconferencing on the Internet. *Signal Processing: Image Communication*, 2005, vol. 20, no. 8, pp. 743–754. doi:10.1109/ICIP.2004.1421484
3. Ramdan A. A., Munir R. Selective Encryption Algorithm Implementation for Video Call on Skype Client. *7th Intern. Conf. on Telecommunication Systems, Services, and Applications (TSSA)*, 2012, pp. 120–124. doi:10.1109/TSSA.2012.6366035
4. Xu Y., Yu C., Li J., Liu Y. Video Telephony For End-Consumers: Measurement Study of Google+, iChat, and Skype. *IEEE/ACM Transactions on Networking*, 2014, vol. 22, pp. 826–839. doi:10.1109/TNET.2013.2260354
5. Manolache F. B., McDowell W., Rusu O. Directory Cache Techniques for Efficient User Management. *10th Roedunet Intern. Conf. (RoEduNet)*, 2011, pp. 1–5. doi:10.1109/RoEduNet.2011.5993696
6. Yan S., et al. Infrastructure Management of Hybrid Cloud for Enterprise User. *5th Intern. DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)*, 2011, pp. 1–6. doi:10.1109/SVM.2011.6096463
7. Lin C. W., et al. Implementation of H.323 Multipoint Video Conference Systems with Personal Presence Control. *Intern. Conf. on Consumer Electronics (ICCE)*, 2000, pp. 108–109. doi:10.1109/ICCE.2000.854518
8. Saveliev A. I., Vatamaniuk I. V., Ronzhin An. L. Architecture of Data Exchange with minimal Client-Server Interaction at Multipoint Video Conferencing. Ed. by S. Balandin, et al. *LNCS*, 2014, vol. 8638, pp. 164–176. doi:10.1007/978-3-319-10353-2
9. Saveliev A. I., Ronzhin A. L. Algorithms and Software Tools for Distribution of Multimedia Data Streams in Client Server Videoconferencing Applications. *Pattern Recognition and Image Analysis*, 2015, vol. 25, no. 3, pp. 517–525.
10. Saveliev A. I. Audiovisual Stream Transfer Control in Videoconferencing. *Trudy Mezhdunarodnoi studencheskoi konferentsii po avtomatizatsii i upravleniiu* [Proceedings of the International Student Conference on Automation and Management], 2013, pp. 107–111 (In Russian).