УДК 004.773.5
doi:10.15217/issn1684-8853.2018.1.2

АРХИТЕКТУРНЫЕ РЕШЕНИЯ ИНТЕГРАЦИИ МОДУЛЯ ВИДЕО-КОНФЕРЕНЦ-СВЯЗИ В КИБЕРФИЗИЧЕСКОЕ ИНТЕЛЛЕКТУАЛЬНОЕ ПРОСТРАНСТВО

Е. Ю. Карасёв[a], *программист, gnomskg@gmail.com*
И. В. Ватаманюк[a], *аспирант, vatamaniuk@iias.spb.su*
А. И. Савельев[a], *канд. техн. наук, старший научный сотрудник, saveliev@iias.spb.su*
А. Л. Ронжин[a], *доктор техн. наук, профессор, ronzhin@iias.spb.su*
[a]*Санкт-Петербургский институт информатики и автоматизации РАН, 14-я линия В.О., 39, Санкт-Петербург, 199178, РФ*

Введение: *киберфизические системы — широкий класс систем, основной характеристикой которых является высокая степень интеграции вычислительных ресурсов в каждый физический компонент системы. Одна из задач киберфизического интеллектуального пространства — обеспечение удобного взаимодействия пользователей друг с другом посредством различных узлов данной системы. Для этого современные приложения видео-конференц-связи должны не только использовать различные модальности: текст, аудио- и видеоданные, графические объекты, — но и соответствовать критериям модульности и кроссплатформенности.* ***Цель:*** *разработка архитектуры приложения видео-конференц-связи, предоставляющего сервис видео-конференц-связи пользователям киберфизического интеллектуального пространства.* ***Результаты:*** *предложены архитектурные решения модуля видео-конференц-связи, обеспечивающие возможность его интеграции в киберфизическую многомодальную информационно-навигационную облачную систему, а также обобщенная структура модуля видео-конференц-связи с реализацией приватных и контролируемых публичных аккаунтов. Сетевое взаимодействие представлено в виде трех отдельных слоев, отвечающих за управление сетевым соединением с сервером, соединением пользователей, а также соединением для передачи мультимедийных данных участников видео-конференц-связи. Произведены замеры времени отклика соединения при организации видео-конференц-связи. В среднем отклик для двух пользователей составил 5,81 мс.*

Ключевые слова *— пиринговое взаимодействие, клиент-серверное взаимодействие, теле-конференц-связь, видео-конференц-связь, модульная архитектура программного обеспечения, киберфизические системы, интеллектуальное пространство.*

Цитирование: Карасёв Е. Ю., Ватаманюк И. В., Савельев А. И., Ронжин А. Л. Архитектурные решения интеграции модуля видео-конференц-связи в киберфизическое интеллектуальное пространство // Информационно-управляющие системы. 2018. № 1. С. 2–10. doi:10.15217/issn1684-8853.2018.1.2
Citation: Karasev E. Yu., Vatamaniuk I. V., Saveliev A. I., Ronzhin A. L. Architectural Solutions for Integrating a Video Conferencing Module into Cyberphysical Intelligent Space. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2018, no. 1, pp. 2–10 (In Russian). doi:10.15217/issn1684-8853.2018.1.2

Введение

Видео-конференц-связь (ВКС) сегодня востребована как государственными министерствами и ведомствами, к примеру, арбитражным и гражданским судом РФ [1], медицинскими организациями [2], образовательными учреждениями [3], так и крупными корпорациями, общественно-политическими организациями — теми институтами, которым необходим простой и удобный инструмент, позволяющий оперативно проводить совещания с удаленными коллегами [4]. Внедрение модуля ВКС в распределенную киберфизическую систему (КФС) предприятия либо организации существенно расширяет возможности пользователей, а также предоставляет дополнительные инструменты для контроля безопасности и управления рабочим процессом.

Кроссплатформенные веб-технологии позволяют адаптировать работу приложения ВКС под различные программно-аппаратные платформы и обеспечивать контроль над обычными и управляемыми аккаунтами в единообразной адаптируемой среде в режиме реального времени. Специализированные системы совместной работы, помимо передачи аудиовизуальных потоков данных, захватываемых микрофонами и видеокамерами, обеспечивают обмен дополнительной мультимедийной информацией, включая [5]:

— совместный доступ к экрану или отдельным приложениям (screen sharing);
— интерактивную доску (whiteboard);
— демонстрацию презентаций;
— синхронный просмотр веб-страниц (co-browsing);
— аннотацию экрана;
— мониторинг присутствия участников;
— текстовый чат;
— интегрированную VoIP-связь;
— видео-конференц-связь;

— возможность менять ведущего;

— возможность отдавать контроль над мышью и клавиатурой;

— модерацию онлайн-встреч;

— обратную связь (например, опросы или оценки);

— планирование встреч и приглашение участников;

— запись хода веб-конференции.

Эффективная передача мультимедийных данных в реальном времени без потери пакетов — одна из немаловажных задач при организации ВКС. В статье [6] исследуется проблема пакетных передач от одного источника нескольким при организации ВКС в реальном времени. Схема передачи TRAding DElay for Reliability (TRADER), предложенная авторами, распределяет отправление пакетов с учетом ограничений по задержке, что существенно улучшает качество передаваемого видеопотока с точки зрения отношения сигнал/шум.

В последние годы мультимедийные приложения на мобильных устройствах пользуются большой популярностью. Благодаря быстрому развитию технологий беспроводной связи пропускная способность мобильных соединений становится достаточной для поддержания потокового видео в реальном времени, что позволяет интегрировать пользовательские мобильные устройства в КФС. Однако из-за их ограниченных возможностей потоковая передача видео высокой четкости (HD) по-прежнему представляет собой непростую задачу. Зачастую мобильные клиенты имеют ограниченный размер экрана, относительно медленное сетевое соединение, а также ограниченное время работы от аккумулятора, поэтому при разработке приложений ВКС для мобильных устройств необходимо учитывать следующие аспекты [7].

1. Изменчивость сетевой среды: в отличие от проводной сети, мобильное соединение зачастую неустойчиво, а пропускная способность может значительно изменяться. Чтобы гарантировать бесперебойную и высококачественную доставку видео для мобильного клиента, система должна динамически регулировать скорость передачи видеопотока, отправленного клиенту.

2. Разнообразие возможностей мобильных устройств: мобильные устройства очень разнородны с точки зрения аппаратных и программных решений для обработки видео. У высокопроизводительных смартфонов часто есть аппаратный ускоритель для декодирования видео, тогда как большинство мобильных телефонов не могут поддерживать аппаратное декодирование. Для повышения производительности видеообработки система должна отправлять видео в подходящем разрешении в соответствии с возможностями клиентов.

При разработке модуля ВКС КФС следует учитывать эти особенности функционирования мобильных клиентских устройств, чтобы оптимизировать обработку мультимедийных данных соответствующим образом. Как правило, средства ВКС ориентированы на захват клиентским устройством аудиовидеоданных только одного пользователя, и чаще всего сеанс связи организован между двумя устройствами. Однако при проведении сеансов связи между несколькими пользователями такой подход зачастую недостаточен, и требуется многоканальная обработка аудиовизуальных сигналов на стороне каждого клиента [8]. В следующем разделе рассмотрены особенности разрабатываемого модуля ВКС, которые позволяют решить проблему обработки и хранения разнородной информации, поступающей в различные моменты времени, обеспечить модульность и простое взаимодействие логических частей КФС [9, 10].

Архитектурные решения модуля ВКС, разрабатываемого для КФС МИНОС

Рассмотрим структуру модуля ВКС, разрабатываемого в рамках многомодальной информационно-навигационной облачной системы (МИНОС) (рис. 1) [9, 10]. С точки зрения КФС обобщенный профиль пользователя ВКС состоит из следующих компонентов:

— профиля пользователя в системе (если пользователь зарегистрирован);

— журналов действий пользователя, включая историю его запросов, историю перемещений, историю проложенных маршрутов, историю вызовов ВКС;

— данных о его клиентском устройстве.

■ ***Рис. 1.*** Обобщенная структура модуля ВКС

■ ***Fig. 1.*** The general structure of the videoconferencing module

Киберфизическая система располагает о пользователе следующей информацией. Если пользователь зарегистрирован в системе, то в его профиле хранятся данные о его ФИО, его статусе (сотрудник, посетитель, прочее), его полномочиях в системе, а также, если это сотрудник, то о подразделении организации, в котором он работает, и занимаемой должности. Кроме того, в профиль может быть добавлена и другая информация, например, о дне рождения сотрудника — для показа поздравительных слайдов в модуле корпоративного телевидения, уведомлений сотрудников по ВКС и т. д.

Отталкиваясь от предложенной схемы, рассмотрим подробнее возможную реализацию модуля ВКС в рамках КФС. Модуль ВКС должен быть доступен на устройствах, интегрированных в КФС, включая терминалы, мобильные робототехнические платформы, рабочие компьютеры сотрудников, а также клиентские устройства пользователей. Объединение перечисленных устройств в единую локальную сеть позволит модулю ВКС функционировать в рамках организации независимо от сети Интернет, что может быть полезно при потере связи во внештатных ситуациях или для обеспечения безопасности предприятия.

Архитектура модуля ВКС пиринговая, что позволяет пользователям подключаться друг к другу напрямую, не используя сервер в качестве промежуточного звена связи. Однако в разрабатываемом решении для КФС серверная часть используется для хранения промежуточных данных, данных пользователей и соединения с другими облачными сервисами, в частности, для авторизации пользователей и обмена «сигнальными» данными. Подобная реализация передачи и хранения «сигнальных» данных на клиенте и сервере обеспечивает буферизацию и последующую обработку «сигнальных» данных, исключая их потерю и поддерживая взаимодействие между группами клиентов [11].

Разрабатываемый модуль ВКС предполагает наличие так называемых публичных аккаунтов (рис. 2), обладающих двумя важными особенностями. Во-первых, публичный аккаунт предоставляет доступ к ВКС незарегистрированным пользователям (с некоторыми ограничениями); во-вторых, публичные аккаунты являются контролируемыми, т. е. зарегистрированные пользователи могут принудительно к ним подключиться. 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 [12].

Аккаунты пользователей подразделяются на публичные и персональные, последние в свою

■ ***Рис. 2.*** Типы и способы взаимодействия аккаунтов ВКС

■ ***Fig. 2.*** Types and ways of interactions of videoconferencing accounts

очередь могут быть открытыми или закрытыми. Так, публичные аккаунты по умолчанию установлены на терминалах КФС и мобильных платформах, предусматривающих возможность использования ВКС. Они не требуют регистрации в системе ВКС и предназначены для оперативной связи с сотрудниками, имеющими открытый персональный аккаунт. Персональные аккаунты зарегистрированных в системе пользователей могут соединяться друг с другом без ограничений (при наличии вызываемого пользователя онлайн и принятии им вызова), кроме того, они обладают возможностью установить принудительное соединение с любым публичным аккаунтом системы. В отличие от открытых персональных аккаунтов, закрытые не доступны для связи со стороны публичных аккаунтов. При желании зарегистрированный пользователь может войти в свою учетную запись через терминал или мобильную платформу, чтобы воспользоваться своим персональным аккаунтом.

Архитектура модуля ВКС

Реализация предложенной гибридной схемы организации сетевого соединения пользователей приложения ВКС представлена рядом программных компонентов, объединенных в одну архитектуру. Чтобы разрабатываемое программное обеспечение (ПО) соответствовало критерию пластичности, сетевое взаимодействие представляется в виде трех отдельных слоев:

1) слой, отвечающий за управление сетевым соединением с сервером приложения;

2) слой, отвечающий за управление соединением пользователей, вступающих в ВКС;

3) слой, отвечающий за управление соединением для передачи мультимедийных данных участников ВКС.

Каждый слой выражен совокупностью объектов, функций, формирующих выраженную абстракцию над логикой обработки запросов к графическому интерфейсу и обращений к соответствующему программному интерфейсу (WebRTC, WebSocket, GraphQL).

Каждый из программных компонентов, представляющих описываемые слои, в своей основе имеет схожую логику управления соединением — операции по открытию, закрытию транспортных каналов, передаче и обработке поступающих сообщений между соединившимися пользователями. В разработанном ПО данная логика отражена в классах *Transport*, *Channel* и его подтипов.

Transport инкапсулирует логику работы с создаваемым соединением, предоставляя клиентам обобщенный программный интерфейс управления (рис. 3).

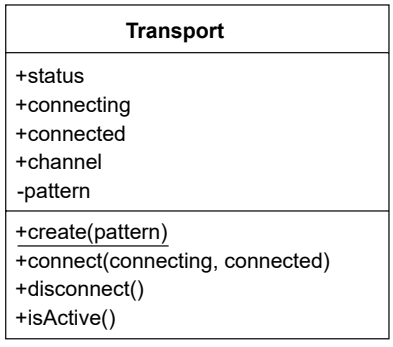
Статический метод *create*(*pattern*) класса *Transport* используется для инстанцирования новых объектов класса. Метод принимает стратегию создания канала для передачи данных между пользователями [13]. Стратегия представляет собой функцию, возвращающую объект типа *Channel* и принимающую в качестве аргументов соединяемых пользователей. При создании объект находится в состоянии ожидания подключения, что выражено значением "*inactive*" свойства *status*. Метод *connect*(*connecting, connected*) позволяет клиенту выполнить соединение пользователей, информация о которых была передана в аргументах, с использованием переданной ранее стратегии. Информация о пользователях должна быть передана посредством объектов типа *PeerData*. При удачном соединении объект класса переходит в состояние "*active*", записывает информацию о соединенных пользователях в свойствах *connecting* и *connected* и сохраняет созданный канал в свойстве *channel*. В случае возникновения ошибок метод *connect* генерирует исключение с соответствующим проблеме сообщением. Для отключения соединения пользователей класс предоставляет метод *disconnect*(), при использовании которого объект класса удаляет сохраненные данные о созданном соединении и переводит *status* в состояние "*inactive*". Таким образом, объект класса *Transport* может быть использован в различных сеансах связи.

Класс *Channel* является базовым для объектов, скрывающих в себе логику создания канала связи для обмена данными между двумя пользователями. *Channel* сводит различия в управлении каналов разных типов к общему для всех интерфейсу, обеспечивая единообразную работу с каналами на всех сетевых уровнях приложения (рис. 4).

Инстанцирование объекта класса *Channel* выполняется посредством статического метода класса *create*(*connection*). В качестве аргумента метод ожидает простой объект *connection* с методами *send*(*event, data*), *onmessage*(*event, callback*) и *close*(), оперирующими активным соединением пользователей. *Channel* выступает в роли класса-адаптера, предоставляя потомкам возможность адаптации интерфейсов создаваемых каналов к общему для всех виду. Подтипы *Channel* инкапсулируют логику создания каналов, расширяя конструктор базового класса. В конструкторе подтипов происходит выполнение соединения пользователей, переданных в аргументе, и последующий вызов конструктора базового класса с передачей созданного *connection* в аргументе.

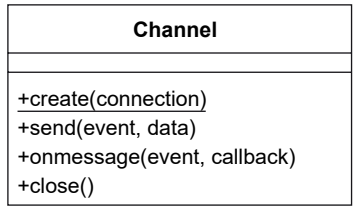
В разработанном ПО для реализации слоев сети использованы следующие подтипы, образованные от *Channel*: *IOChannel*, *P2PChannel*, *MP2PChannel* (рис. 5).

IOChannel принимает в аргументах конструктора данные о соединяемых пользователях, представленных объектами с полями *name* и *address*. При создании объекта выполняется соединение клиента, представленного в *connecting*, с *connected* по WebSocket, с использованием записанного в *connected* адреса.



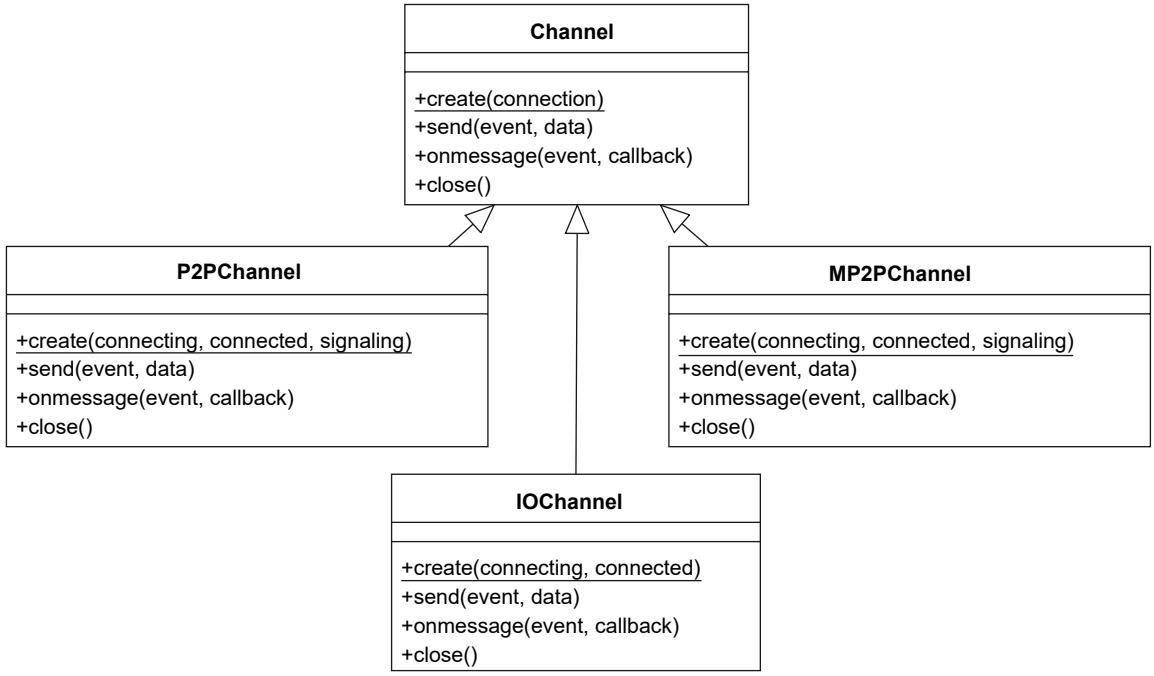
■ ***Рис. 3.*** Диаграмма класса *Transport*

■ ***Fig. 3.*** Diagram of class *Transport*

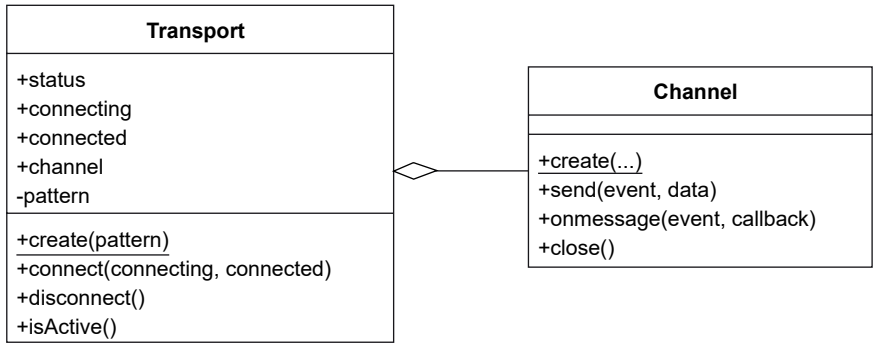


■ ***Рис. 4.*** Диаграмма класса *Channel*

■ ***Fig. 4.*** Diagram of class *Channel*



■ ***Рис. 5.*** Диаграмма иерархии классов, образованных от *Channel*
■ ***Fig. 5.*** Hierarchy diagram of classes inherited from *Channel*



■ ***Рис. 6.*** Диаграмма связи *Transport* и *Channel*
■ ***Fig. 6.*** Diagram of relationship between *Transport* and *Channel*

P2PChannel использует в качестве механизма соединения API *RTCPeerConnection* и *RTCDataChannel* из WebRTC. *P2PChannel* при создании объекта принимает в аргументах конструктора данные о соединяемых пользователях и канал передачи сигнализирующих сообщений. Канал передачи представляет собой объект, образованный от *Channel*. Канал передачи используется на этапе открытия соединения для обмена между клиентами информацией, необходимой для установления сеанса связи. После завершения соединения канал может быть закрыт.

Класс *MP2PChannel* имеет схожую с *P2PChannel* логику работы. Для соединения пользователей он использует API *RTCPeerConnection* из WebRTC и *MediaStream* API, соединяя пользователей для обмена мультимедийной информацией.

При вызове *connect*(*connecting, connected*) объект *Transport* создает канал связи между пользователями с использованием переданной в аргументе конструктора стратегии. Таким образом, *Transport* агрегирует *Channel* или его подтипы (рис. 6).

Объекты класса *PeerData* в разработанном ПО используются как контейнеры для хранения информации о пользователях, участвующих в сетевом взаимодействии. *PeerData* позволяет компонентам использовать в качестве исходных данных единообразные персистентные структу-

ры данных, обеспечивая упрощение взаимодействия внутри ПО (рис. 7).

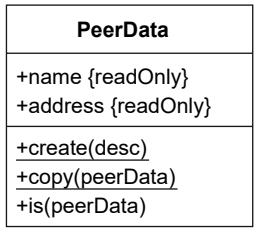
PeerData обладает свойствами *name* и *address*: *name* является уникальным идентификатором пользователя, участвующего в межсетевом взаимодействии; *address* предоставляет адрес, по которому пользователь может быть локализирован. При создании объекта класса конструктор *PeerData* принимает объект с информацией о пользователе, ожидая при этом, что *desc* имеет свойства *name* и *address*. В случае получения некорректного объекта-дескриптора конструктор генерирует исключение с описанием обнаруженной проблемы.

PeerData предоставляет ряд методов, упрощающих работу с созданным контейнером. Статический метод *copy*(*peerData*) позволяет сделать чистую копию существующего объекта *PeerData*. Метод *is*(*peerData*) позволяет сравнивать объекты *PeerData*. Данные методы позволяют обеспечивать неизменяемость данных, сохраненных в *PeerData*, в процессе их использования компонентами ПО.

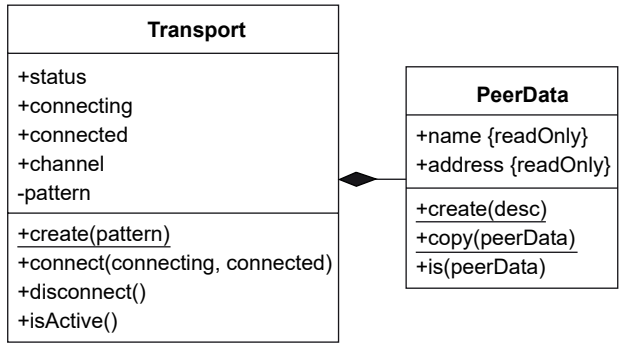
При успешном соединении пользователей объект *Transport* сохраняет копии переданных *connecting* и *connected* в своих свойствах. Таким образом, поля *connecting* и *connected* объекта класса *Transport* имеют тип *PeerData*. Описанные отношения между классами *Transport* и *PeerData* выражены в виде композиции, представленной на рис. 8.

Объекты *Transport* отражают связь между парами пользователей, участвующих в сетевом взаимодействии. Каждый слой ПО сетевого взаимодействия выражен в виде отдельного модуля и сформирован определенным набором объектов *Transport*. Первый слой ПО сети использует объект *Transport* с переданной ему при создании фабричной функцией *createIOChannel*(*connecting, connected*). Функция является оберткой для вызова метода *create*(*connecting, connected*) класса *IOChannel*. Клиентское приложение вызывает метод *connect*(*connecting, connected*) объекта *Transport*, используя в качестве аргументов объекты типа *PeerData* с информацией о себе и о сервере приложения, тем самым выполняя их соединение. Модуль слоя сети устанавливает ряд обработчиков сообщений на объект C*hannel*, записанный в объекте *Transport*, необходимых для синхронизации состояния клиентского приложения с состоянием других пользователей. Кроме того, модуль устанавливает ряд обработчиков событий на действия пользователя, выполняемые с интерфейсом приложения, для отправления соответствующих сообщений на сервер. Второй слой при инициации нового сеанса ВКС попарно соединяет каждого из участников связи, создавая для этого объекты типа *Transport* с передачей им фабричной функции *createP2PChannel*(*connecting, connected, signaling*). В качестве сигнального сервера будет использован созданный на прошлом слое WebSocket-канал. Третий слой выполняет соединение участников ВКС согласно сформированным в процессе инициации ВКС отношениям между участниками, используя для соединения объект *Transport* с переданной ему функцией *createMP2PChannel*(*connecting, connected, signaling*). В качестве сигнального сервера будет использован объект *P2PChannel*, созданный на втором слое сети.

Представленная сетевая архитектура ПО приложения ВКС обладает иерархией отношения между отдельными компонентами. Решения, принимаемые на первом слое сетевой архитектуры, например, об отключении соединения с сервером приложения, могут повлиять на работу второго и третьего слоев. В свою очередь второй слой может повлиять на работу третьего. Иерархические отношения между слоями сетевой архитектуры в ПО выражены посредством композиции. На стороне клиента слои представлены объектами класса *NetworkNode*. При создании объект *NetworkNode* принимает ссылку на слой выше по иерархии. Класс предоставляет метод *createChild()*, создающий дочерний объект *NetworkNode*. Объект класса содержит информацию о всех соединениях на своем уровне (вклю-



■ ***Рис. 7.*** Диаграмма класса *PeerData*
■ ***Fig. 7.*** Diagram of class *PeerData*



■ ***Рис. 8.*** Диаграмма связи *Transport* и *PeerData*
■ ***Fig. 8.*** Diagram of relationship between *Transport* and *PeerData*

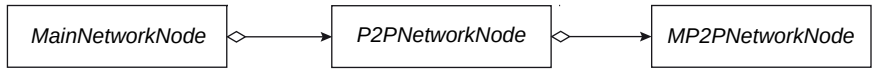
чая информацию о самих пользователях) и может оперировать дочерним *NetworkNode* на свое усмотрение.

Подтипы класса *NetworkNode* инкапсулируют логику работы для каждого слоя сетевой архитектуры. Отношения между уровнями сети, сформированные в приложении ВКФ, отражены на рис. 9.

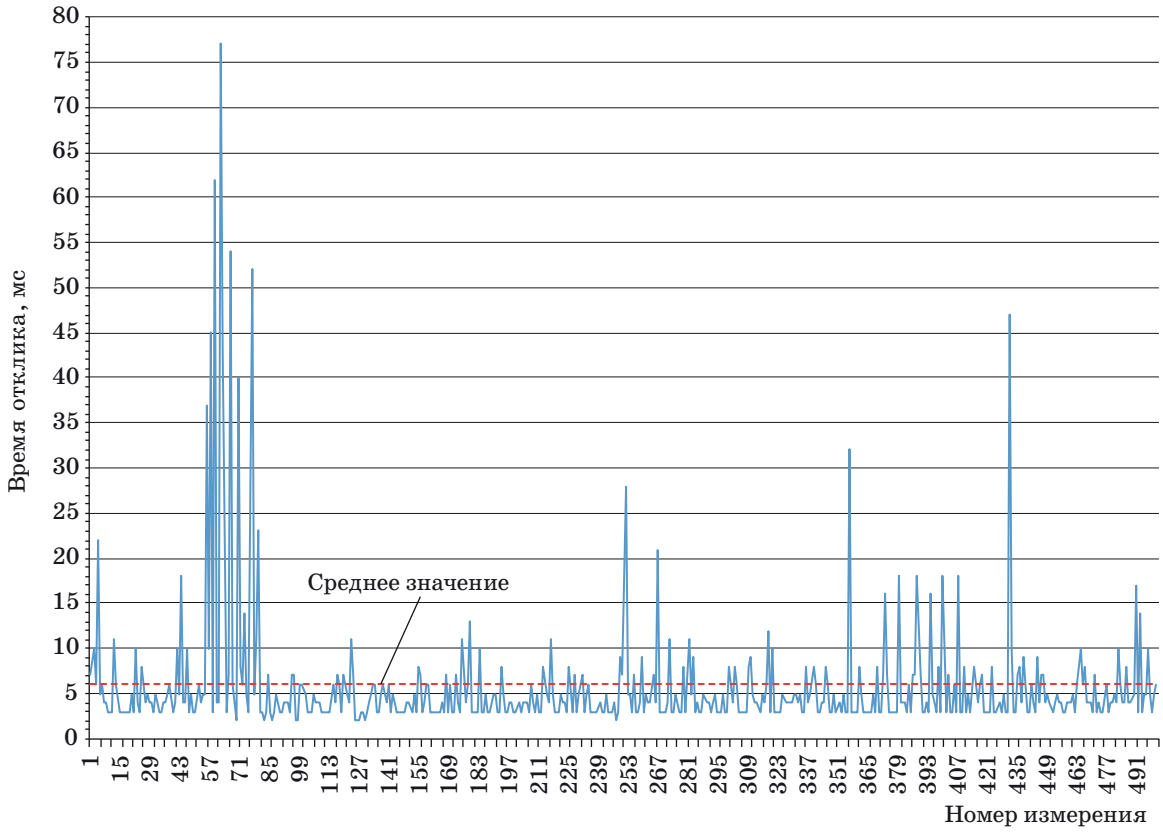
Разделение сети на несколько уровней, действующих в своих рамках практически независимо от остальных, позволяет приложению продолжать осуществлять свои функции при отказе ключевых узлов. В случае отказа сервера приложения работа пользователей не будет нарушена, поскольку между ними по-прежнему будет существовать прямое соединение. Таким образом, активные видеоконференции не будут прерваны. Помимо этого, дополнительная функциональность может быть включена в реализацию каждого из уровней сети как часть композиции модуля, обеспечивая простой алгоритм масштабирования сетевой архитектуры приложения.

В то же время использование описанной схемы соединения обладает недостатками. Второй слой сетевого соединения, обеспечивающий прямое соединение пользователей приложения, подразумевает создание сетевого соединения по схеме «каждый-с-каждым», формируя при этом избыточное количество активных соединений участников. Тем не менее данный подход обеспечивает надежное соединение пользователей в сети, поскольку отключение отдельных узлов сети не сможет повлиять на соединение оставшихся.

Предложенная в работе схема соединения пользователей приложения ВКС, используемая в разработанном ПО, позволила добиться значений времени отклика соединения, представленных на рис. 10. Измерения отклика выполнялись с интервалом 500 мс. В среднем отклик для двух пользователей, находящихся в одном здании, со-



■ ***Рис. 9.*** Организация слоев архитектуры сетевого соединения на стороне клиента
■ ***Fig. 9.*** Organization of architectural layers of the network connection on the client side



■ ***Рис. 10.*** Время отклика пирингового соединения, установленного посредством разработанного ПО
■ ***Fig. 10.*** The response time of the peer-to-peer connection established by means of the developed software

ставил 5,81 мс. Передача осуществлялась по Wi-Fi внутри локальной сети.

Возникновение пиков (выбросов) на графике, наблюдаемое для измерений под номерами 40–76, 252, 357, может быть связано с особенностями порядка исполнения программы интерпретатором кода. Кроме того, поскольку передача осуществлялась по Wi-Fi, увеличение задержки сигнала могло произойти вследствие возникновения помех в окружающей среде.

Заключение

Разрабатываемая архитектура соединения пользователей в приложении ВКС обладает рядом преимуществ, способствующих конкурентоспособности разработанного приложения на фоне аналогов. Предложенная организация модулей ПО сетевого соединения приложения ВКС предоставляет пространство для внедрения дополнительных функций в приложение. В рамках КФС управляемые аккаунты могут использоваться в модулях обеспечения безопасности и управления доступом, например, для обеспечения контроля за рабочими местами, производством и другими объектами. Организация хранения и обработки данных позволяет внедрить модуль ВКС в КФС, а также сделать удобным его использование как на терминалах и мобильных робототехнических платформах, так и на клиентских мобильных устройствах.

Исследование выполнено при поддержке Российского научного фонда (грант № 16-19-00044).

Литература

1. **Решетняк В. И.** Применение видеоконференц-связи в арбитражном судопроизводстве // Российский юридический журнал. 2013. № 1 (88). С. 154–159.
2. **Борисов Д. Н., Иванов В. В.** Организационная телемедицина // Врач и информационные технологии. 2017. № 3. С. 112–120.
3. **Тулемисова Д. З.** Использование видеоконференций в современном образовании // Современные проблемы и направления развития системы подготовки кадров: сб. науч. тр. по материалам I Междунар. науч.-практ. конф./ НОО «Профессиональная наука», 2016. С. 129–135.
4. **Марченков С. А., Вдовенко А. С., Корзун Д. Ж.** Расширение возможностей совместной деятельности в интеллектуальном зале на основе сервисов электронного туризма // Тр. СПИИРАН. 2017. № 1 (50). С. 165–189.
5. **Савельев А. И.** Архитектуры, алгоритмы и программные средства обработки потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи: дис. ... канд. техн. наук / СПИИРАН. — СПб., 2016. — 135 с.
6. **Wu J., Shang Y., Yuen C., Cheng B., Chen J.** TRADER: A Reliable Transmission Scheme to Video Conferencing Applications over the Internet // Journal of Network and Computer Applications. 2014. Vol. 44. P. 161–171.
7. **Cheng R., Wu W., Lou Y., Chen Y.** A Cloud-based Transcoding Framework for Real-Time Mobile Video Conferencing System // Proc. of 2nd IEEE Intern. Conf. “Mobile Cloud Computing, Services, and Engineering” (MobileCloud). 2014. P. 236–245.
8. **Saveliev A. I., Ronzhin A. L., Vatamaniuk I. V.** Architecture of Data Exchange with Minimal Client-Server Interaction at Multipoint Video Conferencing // Lecture Notes in Computer Science. 2014. Vol. 8638 LNCS. P. 164–174.
9. **Левоневский Д. К., Ватаманюк И. В., Савельев А. И.** Многомодальная информационно-навигационная облачная система МИНОС для корпоративного киберфизического интеллектуального пространства // Программная инженерия. 2017. № 3. С. 120–128.
10. **Vatamaniuk I., Levonevskiy D., Saveliev A., Denisov A.** Scenarios of Multimodal Information Navigation Services for users in Cyberphysical Environment // Proc. of SPECOM-2016, Budapest, Hungary. — Springer, 2016. Vol. 9811 LNAI. P. 588–595.
11. **Савельев А. И., Соменков Н. В.** Архитектура клиентской части пирингового приложения видеоконференцсвязи // Математические методы в технике и технологиях. 2016. № 9(91). С. 176–179.
12. **Савельев А. И.** Алгоритмы обработки данных в контролируемых аккаунтах системы видеоконференцсвязи // Информационно-управляющие системы. 2016. № 3. С. 15–23. doi:10.15217/issn1684-8853.2016.3.15
13. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб.: Питер, 2015. — 368 с.

UDC 004.773.5
doi:10.15217/issn1684-8853.2018.1.2

Architectural Solutions for Integrating a Video Conferencing Module into Cyberphysical Intelligent Space

Karasev E. Yu.[a], Programmer, gnomskg@gmail.com
Vatamaniuk I. V.[a], Post-Graduate Student, vatamaniuk@iias.spb.su
Saveliev A. I.[a], PhD, Tech., Senior Researcher, saveliev@iias.spb.su
Ronzhin A. L.[a], Dr. Sc., Tech., Professor, ronzhin@iias.spb.su
[a]Saint-Petersburg Institute for Informatics and Automation of the RAS, 39, 14 Line, V. O., 199178, Saint-Petersburg, Russian Federation

Introduction: Cyberphysical systems are a wide class of systems whose main characteristic is high integration of computing resources into each physical component of a system. One of the tasks of a cyberphysical intelligent space is to provide convenient interaction between users via various system nodes. Modern videoconferencing applications are expected not only to use various modalities like text, audio/video data or graphic objects, but also to meet the criteria of modularity and cross-platformness. **Purpose:** To develop an application architecture which would provide a video conferencing service for cyberphysical intelligent space users. **Results:** The paper discusses architectural solutions for a videoconferencing module, providing the possibility of its integration into a MINOS cyberphysical system, as well as a generalised structure of a videoconferencing module with the implementation of private and controlled public accounts. The network interaction is presented in the form of three separate layers responsible for managing the network connection to the server, the connection between the users, and the connection for transferring multimedia data to the video conference participants. The connection response time during the organization of the video conferencing has been measured. On average, the response for two users is 5.81 ms.

Keywords — Peering Interaction, Client-Server Interaction, Teleconferencing, Videoconferencing, Modular Software Architecture, Cyberphysical Systems, Intellectual Space.

Citation: Karasev E. Yu., Vatamaniuk I. V., Saveliev A. I., Ronzhin A. L. Architectural Solutions for Integrating a Video Conferencing Module into Cyberphysical Intelligent Space. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2018, no. 1, pp. 2–10 (In Russian). doi:10.15217/issn1684-8853.2018.1.2

References

1. Reshetnyak V. I. The use of Videoconferencing in Arbitration Proceedings. *Rossiiskii iuridicheskii zhurnal* [Russian Juridical Journal], 2013, no. 1(88), pp. 154–159 (In Russian).
2. Borisov D. N., Ivanov V. V. Organizational Telemedicine. *Vrach i informatsionnye tekhnologii* [Physicians and IT], 2017, no. 3, pp. 112–120 (In Russian).
3. Tulemisova D. Z. The use of Videoconferencing in Modern Education. *Sbornik nauchnykh trudov po materialam I Mezhdunarodnoi nauchno-prakticheskoi konferentsii "Sovremennye problemy i napravleniia razvitiia sistemy podgotovki kadrov"* [Proc. on the Materials of the First Intern. Scientific and Practical Conf. "Modern Problems and Directions of Development of the System of Personnel Training"]. NOO "Professional'naia nauka" Publ., 2016, pp. 129–135 (In Russian).
4. Marchenkov S. A., Vdovenko A. S., Korzun D. G. Enhancing the Opportunities of Collaborative Work in an Intelligent Room using e-Tourism Services. *Trudy SPIIRAN* [SPIIRAS Proceedings], 2017, vol. 50, pp. 165–189 (In Russian).
5. Saveliev A. I. *Arkhitektury, algoritmy i programmnye sredstva obrabotki potokov mnogomodal'nykh dannykh v piringovykh veb-prilozheniiakh videokonferentssviazi* Dis. kand. tehn. nauk [Architectures, Algorithms and Software for Processing Multimodal Data Streams in Peer-To-Peer Web Applications of Video Conferencing. PhD tech. sci. diss.]. SPIIRAN Publ., 2016. 135 p. (In Russian).
6. Wu J., Shang Y., Yuen C., Cheng B., Chen J. TRADER: A Reliable Transmission Scheme to Video Conferencing Applications over the Internet. *Journal of Network and Computer Applications*, 2014, vol. 44, pp. 161–171.
7. Cheng R., Wu W., Lou Y., Chen Y. A Cloud-based Transcoding Framework for Real-Time Mobile Video Conferencing System. *Proc. of 2nd IEEE Intern. Conf. "Mobile Cloud Computing, Services, and Engineering" (MobileCloud)*, 2014, pp. 236–245.
8. Saveliev A. I., Ronzhin A. L., Vatamaniuk I. V. Architecture of Data Exchange with Minimal Client-Server Interaction at Multipoint Video Conferencing. *Lecture Notes in Computer Science*, 2014, vol. 8638 LNCS, pp. 164–174.
9. Levonevskiy D. K., Vatamaniuk I. V., Saveliev A. I. MINOS Multimodal Information and Navigation Cloud System for the Corporate Cyber-Physical Smart Space. *Programmnaia inzheneriia* [Software Engineering], 2017, vol. 8, no. 3, pp. 120–128 (In Russian).
10. Vatamaniuk I., Levonevskiy D., Saveliev A., Denisov A. Scenarios of Multimodal Information Navigation Services for users in Cyberphysical Environment. *Proc. of SPECOM-2016*, Budapest, Hungary, Springer, 2016, vol. 9811 LNAI, pp. 588–595.
11. Saveliev A. I., Somenkov N. V. Architecture of Client Part of Peer-To-Peer Videoconferencing Application. *Matematicheskie metody v tekhnike i tekhnologiiakh* [Mathematical Methods in Technique and Technologies], 2016, no. 9, pp. 176–179 (In Russian).
12. Saveliev A. I. Algorithms of Data Processing in Supervised Accounts of a Videoconferencing System. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2016, no. 3, pp. 15–23 (In Russian). doi:10.15217/issn1684-8853. 2016.3.15
13. Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 2000. 403 p.