Особености в разработката на VoIP приложения при iOS платформи

гл. ас. д-р Филип АНДОНОВ, Кирил АНГЕЛОВ

НБУ, Департамент Телекомуникации, София, 1618, Монтевидео 21, kiril.angelov@abv.bg

Резюме: *Настоящия доклад описва основните положения при разработване на VoIP приложения, валидни и при разработката на VoIP приложения за операционната система iOS. Наред с общите положения са очертани и спецификите при разработката, засягащи само приложенията за мобилни устройства на Apple с iOS. За пример е използвана разработена авторска VoIP система, обхващаща както сървърна част, така и клиентска част под платформата iOS .*
Ключови думи: *мобилни VoIP приложения, VoIP iOS, VoIP за iOS, VoIP*

Abstract: *This paper describes the design choices when developing a VoIP application and the specifics for an iOS application. All examples are based on the our own developed VoIP system, including both the server and the client side.*
Keywords: *mobile VoIP applications, VoIP iOS, VoIP for iOS, VoIP*

УВОД

Интернет телефония или IP телефония (на английски: Voice over Internet Protocol - глас чрез интернет протокол или накратко VoIP) е технология за предаване на гласови данни в цифров вид чрез капсулирането им в IP пакети, използвайки за транспортиране инфраструктурата на интернет или частна мрежа за данни (LAN, WAN/VPN). Ето и някои предимства и недостатъци на VoIP:

Предимства

- Намаляване цената на телефонните разговори;
- Многофункционалност на линията за връзка;
- Възможност за добавяне на допълнителни функционалности като: препращане на обаждане, аудио конференции, запис на разговори, поддръжка на много на брой активни обаждания, задържане на обаждане, аудио-видео разговори, изпращане на файлове, изпращане на текстови съобщения и т.н.;

Недостатъци

- Загуба на пакети;
- Забавяне на пакети;
- Ехо ефект;
- Проблеми при осъществяване на аудио връзка при клиенти които са зад NAT (Network Address Translation);

Целта на настоящата публикация е да разгледа проблемите и особеностите, възникнали при разработването на VoIP решение за операционна система iOS, както и да очертае някои общи положения в сферата. За примери ще бъдат приложени взетите решения от авторско VoIP решение, включващо както сървърната част, така и клиентската iOS част.

Общи специфики при разработката на VoIP приложения засягащи всякакви VoIP приложения включително и такива за iOS

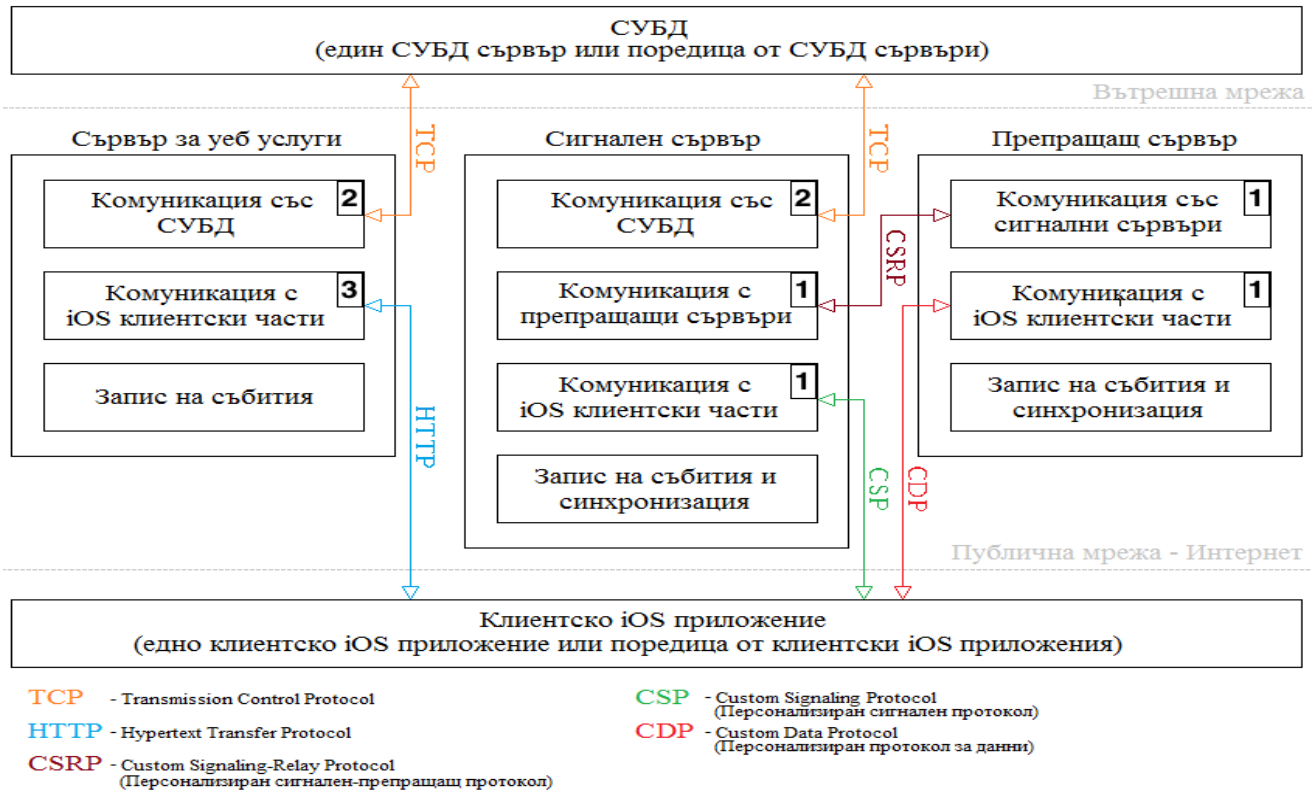
Протоколи за комуникация

Една VoIP комуникация се разделя базово на два слоя - сигнален и слой за транспортиране на медия

(данни). Сигналният слой, отговаря за иницииране на едно обаждане/изграждане на една сесия, а слоя за транспортиране на медия е предназначен за предаването на аудио данните между две точки. Протоколите които се използват за транспорт са от транспортният слоǐ - TCP и UDP. Наǐ-използваните, утвърдени и стандартизирани протоколи за иницииране на обаждане са SIP и H.323 [1]. Тези протоколи използват за пакетиране на данни RTP протокола, след установяване на сесията между два хоста. Следователно това прави RTP наǐ-разпространен протокол за предаване на аудио-видео данни [2]. Въпреки възможността си да използва и двата транспортни протокола, наǐ-често се използва само с UDP. Най-използваните библиотеки с отворен код (заради своя BSD лиценз и удобен програмно-приложен интерфейс) са: reSIProcate за SIP протокола и JRTPLIB за RTP протокола. Въпреки тези стандартизирани протоколи има и приложения които използват собствени авторски протоколи, както за сигналният слой, така и за транспортиране на медия. Поради спецификата си разработваното приложение няма за цел да осигурява свързаност със стандартни VoIP телефонни централи. Така отпадат някои проблеми - например поддръжка на различни протоколи според търсените функционалности, доразработка на стандартизираните протоколи според дадените RFC изисквания и др. Освен това, криптиране на персонализиран (частен) протокол би повишило сигурността на системата [3] и [4]. В резултат на това в авторското VoIP решение беше разработена комуникация с персонализирани комуникационни протоколи. Там сървърната страна е разделена на три сървърни приложения които могат да се мащабират с цел разпределение на натоварването. Тези сървърни части са:

- сигнален сървър - служи за иницииране на обажданията;
- препращащ сървър - играе роля на мост между две точки на една аудио връзка. Използва се когато не е възможно изграждането на P2P връзка;
- сървър за уеб услуги - най-основното му приложение е да осигури информация с кой сигнален сървър трябва да се свърже дадения клиент. Този сървър играе роля на разпределител на натоварването, тъй като връща най-малко използваните сигнални сървъри;

На фигура 1, като пример са илюстрирани връзките и използваните протоколи между разработените части на VoIP решението. Правоъгълниците с номера 1 обхващат комуникацията използвайки персонализираните протоколи. Номерираните с 2 и 3 илюстрират комуникацията с неперсонализирани протоколи като TCP се използва за комуникацията с базата данни, а HTTP се използва за връзката на iOS приложението със сървъра за уеб услуги.



Фиг.1. Връзки и използвани протоколи в разработеното VoIP решение.

Проблеми при P2P комуникация

P2P (Peer-to-peer) свързаността представлява директна Unicast връзка (съобщенията използват асоциация „един към един", като "destination" адресът дефинира една-единствена крайна точка [5]) между два хоста в която те комуникират помежду си. Този тип свързаност е за предпочитане в VoIP приложенията когато връзката се отнася към обмена на аудио данните, тъй като директната свързаност гарантира по-бърз обмен на данни от използването на клиент-сървър архитектурата. Проблемите при P2P комуникацията идват от така наречените NAT (Network Address Translation). При NAT технологията IP адресите на подателя и получателя се пренаписват от маршрутизатор или защитна стена. Хостовете на една частна мрежа може да се свързват помежду си, но не и с хостове от друга частна мрежа. За да бъде реализирана една връзка между два хоста трябва поне един от хостовете да бъде видим (достъпен) в интернет. Когато един хост направи опит да се свърже с външен за мрежата адрес, NAT-а пренаписва неговият адрес с друг който е видим. По този начин всички заявки към интернет могат да излизат с един и същи адрес. За разрешаване на този проблем VoIP (и друг P2P софтуер) използват различни похвати. Най-използваните са:

- UPnP (Universal Plug and Play), NAT PMP (NAT Port Mapping Protocol), NAT PCP (NAT Port Control Protocol) - Това са протоколи които позволяват на мрежовите устройства да се свързват помежду си. С тяхно съдействие на NAT-а се отварят портове, които спомагат мрежовата свързаност. Тези портове са отворени докато вътрешно-мрежовото устройство ги използва;
- Hole punching - При този метод се ползва външен сървър. Външният сървър служи за установяването и разменянето на вътрешните и външни адреси и портове на и между двата хоста. След това всеки един от двата хоста започва да пращат пакети на NAT-а на другия, при което всеки от двата NAT-а решава, че е имало връзка между тях и разрешава комуникацията;
- Super Nodes (супер възли) - Това е метод при който всеки възел (мрежово устройство, използващо дадения софтуер) може да стане супер възел за друг който е в подмрежа. По този начин супер възела изпълнява ролята на proxy и разпределя

трафика или към друг супер възел (на по-горно ниво от него) или към целевия хост. Например: ако имаме два хоста (които не са видими един за друг, заради наличието на NAT-тове) в различни частни мрежи, те могат да се свържат помежду си чрез използването на супер възел, който е с реален адрес в публичната интернет мрежа;

- Използване на препращащи (Relay) сървъри - Този тип сървъри играят роля на супер възли, с тази разлика, че те са там само и единствено за да предоставят възможност за обмен на данни между две приложения. Използването на препращащ/и сървъри обикновено се налага когато нито един от предишните методи не сработи. Това е така понеже за препращане на аудио данни един сървър се натоварва значително, когато клиентите свързани с него станат много и затова се използва като последна възможност. За това и обикновено се използват различни препращащи сървъри според натоварването им. Самото разпределяне на натоварването се поема от система (обикновено сървър) за разпределяне на натоварването (Load balancing);

Във фигура 1, по-горе ясно се вижда, че за разработеното VoIP приложение е използван варианта с препращащите сървъри. Най-широко използваният метод е “Hole punching”. За това той е предвиден за бъдещо развитие на системата.

Аудио

Аудио данните се записват чрез микрофон, като звука се трансформира от аналогов в цифров вид. При възпроизвеждането е точно обратното, от цифров вид данните се трансформират в аналогов, като се възпроизвеждат посредством високоговорител или слушалки. Това трансформиране се извършва на хардуерно ниво. На софтуерно ниво програмиста трябва да окаже на аудио драйвера в какъв формат иска аудио данните или в какъв формат ще ги подава, както и честотата на дискретизация. Някои от основните термини при използване на аудио са:

- аудио канал - индивидуален аудио сигнал. Когато има един канал се нарича - моно звук. А ако каналите са два - стерео звук;
- sample, сампъл (проба) - това е една единствена числова стойност от цифровия сигнал;
- frame (фреим) - фреим представлява аудио данни от по един сампъл от всички канали. Ако имаме само един канал, то фреима ще съдържа само един сампъл. Ако каналите са два, то фреима ще е съставен от два сампъла;
- sample rate (честота на дискретизация) - броя на фреймовете за една секунда. Колкото повече сампъла са взети за една секунда, толкова по-точен е звука;
- sampling (семплиране) [6] - процесът при който звуковата вълна (непрекъснат сигнал) се превръща в последователност от сампъли (дискретен сигнал);
- aliasing [7] - това е ефекта при който се получават различни сигнали от оригиналният когато те се семплират. Тези сигнали са с по-лошо качество;

За да се избегне ефекта на получаване на изкривявания на сигнали (Aliasing), трябва да се има предвид правилото: Най-високата честотна компонента измерена в Херци (Hz) - fmax, за даден аналогов сигнал, трябва да бъде поне два пъти по-малка от честотата на дискретизация - fs. Или: fs >= 2 * fmax. Всъщност това правило е доказано от теоремата на Найкуист-Шенън (Nyquist–Shannon), наречена Nyquist Theorem. Повече информация може да бъде намерена на [6], [7] и [8]. В едно VoIP iOS приложение, частта с аудиото се реализира посредством допълнението “Audio Units” в софтуерната рамка “Core Audio” на Apple. Чрез него се настройва и използва аудио драйвера (компютърна програма на ниско ниво, комуникираща директно с хардуера) на даденото мобилно устройство. Извличането и установяването на аудио данни съответно за запис и възпроизвеждане се постига чрез вграденият механизъм за известяване с функции за обратно извикване от страна на аудио драйвера.

Аудио кодеци

За да се намали обема на аудио данните които се записват или възпроизвеждат се налага компресирането и декомпресирането им с цел предаване на по-малко количество данни по мрежата. Тук на помощ идват аудио кодеците. Чрез тяхна помощ може да се намали размера на аудио данните, като по този начин може да се изпратят или получат повече данни. Така един аудио разговор би бил в реално време. Не компресирането на аудио данните би довело до забавяне особено

при по-бавна мрежа, при по-малка пропускливост на мрежата.

В разработеното iOS приложение е използван аудио кодека “Opus”. Той има реализация с отворен код и BSD лиценз който е съвместим с разработването на iOS приложения, тъй като позволява библиотеките да бъдат компилирани и използвани като статични. От друга страна Apple разрешават използването само на статично свързване при компилирането на iOS приложение за AppStore. Статичното свързване се извършва по време на компилацията от свързваща програма когато се изгражда изпълнимият файл. Статично свързани библиотеки не могат да бъдат подменяни в дадена компилирана програма. Използването на аудио кодека става чрез функции за кодиране и декодиране на потока от данни. Има само една важна подробност. При инициализацията на аудио кодека, трябва да му бъдат подадени същите настройки като тези които се подават на аудио драйвера. Това се прави с цел аудио драйвера и кодека да работят на едни и същи честоти и с еднакви дължини на фреймовете.

Методи за входно-изходни известия

Този тип методи се използват в Event-Driven programming (събитийно-задвижвано програмиране). Този тип програмиране е парадигма, както обектно-ориентираното програмиране. Специфичното в него е, че програмният поток се определя от събития. Такива събития могат да бъдат: потребителски (натискане на клавиш, натискане на бутон на мишката), сензорни (генерирани от устройство), мрежови (събития които се случват при промяна в мрежовата архитектура, например състояние на сокет, загуба на свързаност) и тн. Във VoIP програмирането, методите за входно-изходни известия са полезни за решаването на така наречения “The C10K Problem” [9]. Този термин описва проблема с оптимизацията на мрежови сокети така, че да може да се поддържат и да се комуникира с голям на брой сокети. Методите за известявания позволяват на дадено приложение да бъде известявано за всяко едно събитие което е възникнало на даден сокет. Такова събитие може да бъде: отворен сокет, получени нови данни от другата страна, готовност за изпращане и др. Тези са най-използваните методи за известяване:

- select - Многоплатформен метод за известяване. Работи с битови карти на файловите дескриптори. При възникване на събитие, операционната система презаписва битовата карта на дадения сокет и след това за да се регистрира отново същия сокет (за да се провери отново за събитие) трябва да се пресъздаде отново цялото множество от файлови дескриптори, и да се извика отново select. За да извести за промяна операционната система трябва да обходи всички налични битови карти. След което приложението също трябва да обходи всички битови карти за да разбере кой сокет е променен. Поради тази причина този метод е по-удобен за известяване при работа с по-малък брой сокети. Select има ограничение до 1024 битови карти, което ще рече, че може да следи до 1024 сокета. Също така select не позволява промяната на файловия дескриптор докато трае извикването му. Друг недостатък е, че няма известие за това дали отдалечения сокет е затворен. За да се разбере трябва даден сокет да се регистрира за четене, което не е много удобно;
- poll - Posix вариант за известяване. Той работи не с битови карти, а с масиви от структури съдържащи файлови дескриптори, събития за които да се регистрира и възникнали събития. Poll няма ограничение за това колко сокета да следи. Този метод не връща по никакъв начин само тези сокети по които има промяна. Следователно след неговото извикване трябва да се обходи целия масив и да се провери във всяка една структура в кой от сокетите има промяна. Освен това операционната система също трябва да обходи всеки един сокет за да провери за промяна и тогава да установи съответните битове. Също като select, той е подходящ за по-малко на брой сокети. Също така не позволява масива да бъде променян или да се затвори сокет който е следен в масива докато не приключи Poll извикването;
- epoll - Linux вариант за известяване. Той се различава от предишните два метода по това, че пази информация за следените сокети (и свързаните с тях събития) в ядрото на операционната система и предоставя функции за манипулиране на сокетите. Едно от предимствата му е в предоставянето на функция за следене на определен брой сокети, като връща само променените сокети. По този начин не е нужно да се обхождат големи масиви и да

се проверява за възникнали събития, понеже върнатите резултати винаги ще са сокети в които са възникнали събития. Също така тази функция позволява да и се зададе колко резултата да върне. Ако броя на зададените резултати е по- малко от реално възникналите събития, то ще върне колкото му е изискано, а останалите ще останат на опашка за следващо извикване на функцията. Това му предимство да връща само сокетите с възникнали събития, прави този метод много удобен за работа и по-бърз от предишните два метода при по-голям брой сокети. Бързодействието се подсилва и от друга важна част на този метод. Това е възможността на операционната система да маркира дадени сокети когато са променени, дори и да не е извикана функция за запитване от приложението. При това положение когато се извика такава функция, ядрото на операционната система няма тепърва да проверява сокетите, а само ще копира резултата и извикването на функцията ще приключи. Друго предимство е възможността да се променят, изтриват и добавят събития по всяко време. Един от основните му недостатъци е, че за всичко трябва да се използват предоставените функции. Например ако трябва да изтрием 5000 сокета от добавените за следене, то ще трябва да се направят 5000 системни обръщения използвайки функция за изтриване. За разлика от EPoll, при предишните два метода това би могло да се направи с просто обхождане на масива при pool или с изтриване от битовата карта при select. Следователно този метод не е подходящ за приложения които имат такъв тип връзки които да бъдат създавани за кратко (за малък трафик) и след това да се освобождават;

- Input/Output completion port - Windows вариант за известяване. Също като epoll, този метод е удобен при работа с голям брой сокети. Основната разлика между двата споменати метода е начинът има на работа. EPoll работи като известява за готовност на сокета в дадено отношение, например това може да е: готов за четене (има постъпили данни в сокета), готов за писане(има свободно пространство за запис на данни) и тн. При IOCP нещата седят по друг начин. Той известява след като дадена операция вече е изпълнена (данните са прочетени, изпратени и тн.). Този метод, както и epoll могат и са удобни за използване в многонишкова обработка. Друга разлика е, че IOCP блокира изпълнението чрез извикването на GetQueuedCompletionStatus, докато само една операция не завърши или чрез GetQueuedCompletionStatusEx за повече от една операция. Освен това IOCP се нуждае от буфери, които не трябва да бъдат променяни докато не извести за приключване, тъй като първо се пуска изпълнението на дадена операция а след това се чака тя да бъде изпълнена. Заради това се налага използването на други временни буфери които да съхраняват информация докато операцията приключи и буфера на дадената операция може да бъде променян. IOCP също като epoll предлага промяна на операция, като добавяне на събитие, но това става малко по-сложно тъй като трябва предишната операция да бъде отказана при това от нишката в която е създадена. Тук резултата от операцията не е ясен и трябва да бъде допълнително проверен. При epoll не е нужно да се отказва и промяната може да се направи от всяка нишка;
- /dev/poll - Solaris вариант за известяване, наличен и при Unix и Linux (въпреки, че Linux имплементацията не е добра). Той представлява драйвер, чрез който софтуерното приложение може да използва събитийното известяване за зададени сокети. Този метод е ограничен само за сокети, позволява следене на голям брой сокети, но промяната им в реално време не е удобно. Това е така понеже се отваря като външна системна програма. По този начин /dev/poll е неудобен за работа в многонишкова среда, заради проблеми със синхронизацията;
- event ports - Също Solaris вариант за известяване. Този метод е създаден за да реши недостатъците на стандартния poll. Разбира се, позволява работа в многонишкова среда и е удобен е за работа с голям брой сокети. За целта предоставя собствен приложно-програмен интерфейс. Този метод много наподобява epoll, като предимства и недостатъци;

- kqueue - FreeBSD вариант за известяване, наличен и в други BSD базирани операционни системи, както и Mac OS X. Той не е създаден за да решава проблеми при някои от другите методи, а по-скоро като основен механизъм за редица типове на събития на операционната система. Този метод има едно много голямо предимство спрямо другите описани методи. А именно актуализация на множествата от събития чрез едно единствено извикване на kevent(). При epoll например това става с epoll_ctl(), като се налага извикването му толкова пъти колкото промени ще се направят в следеното множество. Това му предимство го прави едно от най-мощните и бързи инструменти за известяване. Разбира се, kqueue също е удобен за работа с голям брой сокети. Позволява работа в многонишкова среда;

Споменатите методи засягат по-скоро сървърната организация от колкото клиентската част на едно VoIP решение, понеже сървърната част поема най-голямото натоварване. Въпреки това, те са много полезни и в клиентската част, тъй като биха улеснили и ускорили времето за разработка на комуникацията със сокети. Тези методи са разработени в различни библиотеки с отворен код, като: libevent, libev, libuv, Boost.Asio и др. В разработеното iOS приложение се използва библиотеката libevent и съответно метода kqueue.

Особености при разработката на VoIP приложения за операционна система iOS

IP адреси и DNS64/NAT64

Apple изисква всички VoIP приложения качени в AppStore да могат да работят с IPv6 мрежи. Поради тази причина вече, VoIP приложенията трябва да поддържат и двете мрежи - IPv4 и IPv6. За целта Apple предоставят на разработчиците възможност за тестване на приложенията, за да се установи дали отговарят на това изискване. Самата възможност представлява поддръжка на DNS64/NAT64 мрежи в операционната система Mac OS X 10.11.x. Операционната система позволява на разработчиците да симулират реална среда с IPv6 адреси, като за тестовите цели те трябва да свържат устройствата към дадената Mac машина. Ако едно VoIP приложение трябва да се свърже към IPv4 мрежа, то трябва да пренапише IPv4 адреса като IPv6 такъв. Това става лесно с функцията getaddrinfo (на езика C), която от примерен IPv4 адрес “192.0.2.1” ще го пренапише като “64:ff9b::192.0.2.1”. Когато NAT64 види пренаписания адрес ще разбере, че трябва да продължи комуникацията нагоре като IPv4.

Режим на работа във фонов режим

Едно VoIP приложение трябва да може да поддържа позвънявания във фонов режим. В противен случай, би трябвало потребителя да държи приложението стартирано на преден план. Това би довело до бързо изтощаване на батерията на едно мобилно устройство. Освен това би ограничило потребителя да не използва други приложения. За това поддръжката на фоновия режим е много важен елемент от VoIP при мобилните устройства. До преди версия 10 на iOS се позволяваше на разработчиците да оставят отворени TCP сокети в фонов режим, по които да може да се инициира обаждане. Като самите сокети се задават чрез флаг, че ще се използват за VoIP. След версия 10 на iOS, тази функционалност вече не работи. Това е така защото Apple умишлено ограничават работата на VoIP сокетите с цел да се използва новата възможност, а именно “VoIP Push Notifications” (VoIP пуш известия). Те се поддържат от версия 8 на iOS и са с по-висок приоритет от стандартните пуш известия. Също така Apple гарантират, че VoIP известията ще бъдат доставяни на време, за разлика от стандартните. Недостатъка е, че на разработчиците ще се налага изграждане на допълнителна сървърна част която да комуникира с Apple APNS сървърите.

Бутони за звука на Apple устройство с iOS

Всички приложения не само VoIP, ще бъдат отказвани за AppStore, ако са разработени така, че да променят стандартната функция на бутоните за звука на едно Apple устройство.

ЗАКЛЮЧЕНИЕ

Разработката на VoIP приложения за операционна система iOS изисква няколко допълнителни стъпки, които неминуемо удължава процеса по програмиране и тестване. Дори и да се използват библиотеки с отворен код, те трябва да бъдат тествани дали отговарят на последните изисквания на Apple. Разгледаните особености вероятно може да бъдат въведени и при други мобилни операционни системи, особено изискванията за поддръжка на IPv6 мрежи, поради

изчерпването на IPv4 адресите и все по-зачестеното използване на IPv6. Наред с това удобството и предимствата на VoIP пуш известяванията също не са за подценяване. Разработеното VoIP софтуерно решение имплементира голяма част от разгледаните добри практики за разработка на VoIP приложения. То е създадено с конкретна насоченост и се използва като пример за използваните методи.

ЛИТЕРАТУРА

[1] https://www.packetizer.com/ipmc/papers/understanding_voip/voip_protocols.html, October 2017

[2] Jose Saldana, Jenifer Murillo, Julián Fernández-Navajas, José Ruiz-Mas, Eduardo Viruete Navarro, José I. Aznar. “Evaluation of Multiplexing and Buffer Policies Influence on VoIP Conversation Quality”, Dpt. IEC. Ada Byron Building. CPS Univ. Zaragoza. 50018 Zaragoza, Spain

[3] Пасарелски Р., Механизми за защита и криптиране на VoIP трафика, Сборник научни трудове от годишна научна конференция 30 септември - 01 октомври 2010г. на национален военен университет “Васил Левски” – гр. Велико Търново, том 5, ISSN: 1314-1937, Велико Търново, 2010

[4] Пасарелски Р., Системи за откривания на проникванията при VoIP, Академично списание “Управление и образование” 2011, книга 3, том 7 на Университет “Проф. д-р Асен Златаров” - Бургас ", ISSN: 13126121, Бургас, 2011

[5] Г. Петров, РАЗВИТИЕ НА ИНТЕРНЕТ И ОТВОРЕНИТЕ СИСТЕМИ, Част 1, Глава 2, (163р. - 180р), Авангард Прима София, 2017, ISBN 978-619-160-834-8

[6] http://support.ircam.fr/docs/AudioSculpt/3.0/co/Sampling.html, October 2017

[7] https://en.wikipedia.org/wiki/Aliasing, October 2017

[8] https://en.wikipedia.org/wiki/Nyquist–Shannon_sampling_theorem, October 2017

[9] http://www.kegel.com/c10k.html, October 2017

За повече информация пишете на e-mail
kiril.angelov@abv.bg
fandonov@nbu.bg