Московский
государственный университет
им. М.В. Ломоносова

И. В. Владимиров
О. А. Гаврилов
С. В. Соколова

ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ СПУТНИКОВОЙ СВЯЗИ:
АНАЛИЗ, СРАВНЕНИЕ, ПРОГНОЗ РАЗВИТИЯ

Монография

SCIENTIA
ИЗДАТЕЛЬСКИЙ ДОМ

Санкт-Петербург
Издательский дом «Сциентиа»
2022

УДК 621.396.946
ББК 32.884.161.5
В57

Рецензент:

Яблочкин К. А. — к.т.н., доцент кафедры линий связи и измерений в технике связи Поволжского государственного университета телекоммуникаций и информатики

Владимиров, Игорь Викторович

В57 **Протоколы транспортного уровня спутниковой связи: анализ, сравнение, прогноз развития** : монография / И. В. Владимиров, О. А. Гаврилов, С. В. Соколова ; Московский государственный университет им. М. В. Ломоносова. — Электронные данные. — Санкт-Петербург : Сциентиа, 2022. — 4,1 Мб ; 76 с. — Режим доступа: https://scientia-pub.org/index.php/Sci/catalog/book/33 — Загл. с экрана.

ISBN: 978-5-6048667-0-2. — DOI: 10.32415/scientia_978-5-6048667-0-2.

Монография представляет собой систематический обзор современных и перспективных сетевых протоколов транспортного уровня в системах цифровой спутниковой связи. Специфика спутниковой связи связана с большими задержками передачи данных, высокой частотой ошибок по битам и ограничением полосы пропускания. Поэтому в работе уделено особое внимание организации эффективной передачи данных в сетях, содержащих спутниковые сегменты, а также выделены подходы к разработке новых решений на основе традиционных сетевых протоколов. Описано взаимодействие протокола TCP с протоколами нижележащего уровня. Проведен анализ протоколов с архитектурой, унаследованной от TCP: TCP Peach, TCP Westwood, TCP Hybla, CK-STP, SCTP. Описаны современные способы повышения производительности протокола TCP. Описаны перспективные протоколы для применения в системах спутниковой связи (QUIC, XSTP, STP).

УДК 621.396.946
ББК 32.884.161.5

ISBN: 978-5-6048667-0-2
DOI: 10.32415/scientia_978-5-6048667-0-2

Авторы благодарят Гурьева Дмитрия Евгеньевича, научного сотрудника лаборатории ОИТ факультета ВМК МГУ им. М.В. Ломоносова за ценные замечания при подготовке этой монографии.

ОГЛАВЛЕНИЕ

СПИСОК СОКРАЩЕНИЙ

ACK . . . . . Acknowledgment
ACM . . . . . Adaptive Coding and Modulation
AeNB . . . . . Aerial eNB
AIDA . . . . . Agile Integrated Downconverter Assembly
AIMD . . . . . Additive Increase Multiplicative Decrease
AMR . . . . . Automatic Meter Reading
AP . . . . . Access Provider
ARQ . . . . . Automatic Repeat reQuest
BDP . . . . . Bandwidth-Delay Product
BER . . . . . Bit Error Rate
BFN . . . . . Beam Forming Network
BH . . . . . Beam Hopping
BIC . . . . . Binary Increase Congestion control
BSM . . . . . Broadband Satellite Multimedia
CCSDS . . . . . Consultative Committee for Space Data Systems
CBQ . . . . . Class Based Queuing
CDN . . . . . Content Delivery Networks
CP . . . . . Content Provider
CR . . . . . Content Router
CRC . . . . . Cyclic Redundancy Check
cwnd . . . . . congestion window
DAMA . . Demand Assignment Multiple Access
DPI . . . . . Deep Packet Inspection
DRA . . . . . Direct Radiating Array
DTN . . . . . Delay/Disruption Tolerant Network
EPC . . . . . Evolved Packet Core
ESA . . . . . European Space Agency
FEC . . . . . Forward Error Correction
FN FR . . . . . Forwarding Node Fast Retransmit
FTP . . . . . File Transfer Protocol
GEO . . . . . Geostationary Orbit
GFP . . . . . Generic Flexible Payload
GW . . . . . Gateway
HAP . . . . . High Altitude Platform
HTTP . . . . . Hypertext Transfer Protocol
ICN . . . . . Information Centric Networking
IMUX . . . . . Input Multiplexer
IoT . . . . . Internet of Things
IP . . . . . Internet Protocol
IRIS . . . . . IP Routing in Space
ISL . . . . . Inter-Satellite Link
ISP . . . . . Internet Service Provider
LEO . . . . . Low Earth Orbit
LNA . . . . . Low Noise Amplifier
LRU . . . . . Least Recently Used
MAC . . . . . Media Access Control
MPA . . . . . Multi-Port Amplifier
MPLS . . . . . Multi-Protocol Label Switching
M-PSK . . . . . Multiple phase-shift keying
MSS . . . . . Maximum Segment Size
NFV . . . . . Network Function Virtualization
NMC . . . . . Network Management Center
NRS . . . . . Name Resolution Service

PBR. . . . . Policy-Based Routing
PEP. . . . . Performance Enhancing Proxy
PER. . . . . Packet Erasure Rate
PER. . . . . Packet Error Rate
PLA . . . . . Packet Level Authentication
PLMU . . . . . Portable Land Mobile Unit
PLR. . . . . Packet Loss Rate
PSI . . . . . Publish-Subscribe Internetworking
QoE . . . . . Quality of Experience
QoS. . . . . Quality of Service
RA . . . . . Random Access
RASE . . . . . Routing and Switching Equipment
RTO . . . . . Retransmission Time Out
RTT . . . . . Round-Trip Time
SACK . . . . . Selective Acknowledgment
SCPS-TP . . . . . Space Communications Protocol Specifications — Transport Protocol
SDN . . . . . Software Defined Networking
SFPB. . . . . Single Feed Per Beam
SIC . . . . . Successive Interference Cancelation
SNACK . . Selective Negative Acknowledgment
SR-ARQ. . . . . Selective-Repeat ARQ
SSCOP . . . . . Service Specific Connection Orientated Protocol
SSPA. . . . . Solid State Power Amplifier
STP. . . . . Satellite Transport Protocol
SVNO . . . . . Satellite Virtual network Operator
TCP . . . . . Transmission Control Protocol
TWTA . . . . . Travelling Wave Tube
UAV . . . . . Unmanned Aerial Vehicle

ВВЕДЕНИЕ

Спутники играют важную роль в услугах телефонной связи и телевещании с самого первого момента их появления на орбите Земли. Менее известно, что спутники также играют важную роль в услугах широкополосного доступа в Интернет и продолжат выполнять свои функции в сетях будущих поколений. Это связано с уникальными характеристиками спутников, которые заняли свою нишу в глобальной сетевой инфраструктуре.

В последнее время спутниковые сети — это особая важная тема наряду с другими сетевыми технологиями. Из-за характера спутниковых каналов (длительная задержка распространения, относительно высокая частота ошибок по битам и ограниченная полоса пропускания по сравнению с наземными линиями связи, особенно оптическими линиями) некоторые стандартные сетевые протоколы не работают должным образом и должны быть адаптированы для поддержки эффективного соединения через спутник. Более того, спутниковая орбита напрямую влияет на характеристики канала и существенно влияет на проектирование спутниковой сети.

Конечная цель спутниковых сетей — связь и поддержка множества различных приложений и услуг, доступных в наземных сетях. Эти приложения и услуги генерируют различные типы трафика, имеющие разные требования с точки зрения сетевых ресурсов и качества обслуживания, в частности, разработки по интеграции телекоммуникационных, широковещательных и компьютерных сетей, а также по интеграции телефона, телевидения, компьютера и терминалов глобальной системы позиционирования (GPS).

Со времени появления первого телекоммуникационного спутника спутниковые сети претерпели значительные изменения: от телефонных и вещательных до широкополосных и Интернет-сетей.

Спутники были адаптированы в процессе своего развития для ISDN, ATM, Интернета, цифрового вещания и так далее. Такая эволюция происходит благодаря исследованиям и разработкам в области бортовой коммутации и бортовой IP-маршрутизации. Есть также новые разработки и новые проблемы в развитии спутниковых сетей, такие как управление ресурсами, безопасность и качество обслуживания, новые услуги и приложения, включая VoIP, многоадресную передачу, видеоконференцию, DVB-S, DVB-RCS и IPv6 через спутник. Также всегда существует множество практических ограничений, таких как сложность технологии, стоимость её внедрения, эффективность космического и наземного сегментов связи при проектировании, реализации и эксплуатации. Исходя из этого, зачастую разработчикам и инженерам приходится идти на компромисс для достижения оптимального решения.

Развитие технологий в системах спутниковой связи стабилизировалось и повзрослело, так что современные спутниковые сети можно рассматривать как неотъемлемую часть глобальной сетевой инфраструктуры, а не как сложную систему как таковую. Таким образом, актуальное развитие спутниковых технологий неразрывно связано с развитием и стандартизацией протоколов.

Протоколы спутниковой связи — интересная область научно-технической разработки, которая сейчас развивается особенно динамично. Это можно видеть по тому, насколько оперативно поправляются отчёты и вносятся коррективы в уже действующие протоколы, цель которых обеспечить надежную передачу данных со спутниками, разворачиваемыми на низкой орбите Земли. Например, система телекоммуникационных спутников Starlink.

Спутники системы Starlink сейчас в большом количестве (около 2700) выведены на ближнюю орбиту Земли, и в рабочем режиме передают телекоммуникационную информацию. Телекоммуникационная информация — это информация, которая передается со спутника на Землю (DownLink), терабайты данных

в сутки, которые оперативно анализируются. По результатам анализа выходят необходимые прошивки, вносятся изменения в программы общения и протоколы. Со спутника на Землю передаётся телеметрическая информация, а с Земли на спутник (UpLink) передаются телекоманды. Эти потоки данных (DownLink и UpLink) отличаются частотным диапазоном, то есть рабочей частотой. Таким образом, можно сказать, на наших глазах формируются новые протоколы передачи данных.

Одним из авторитетных источников по каталогизации протоколов системы спутниковой связи является сайт Международного Консультативного Комитета по космическим системам передачи данных (Consultative Committee for Space Data Systems — CCSDS). В Международный Комитет по космической передаче данных входят специалисты из разных областей, которые привносят свой опыт в разработку новых протоколов спутниковой связи.

Развитие спутниковых систем напрямую сопряжено с трудностями передачи сигнала, его потерей и искажения. Также особую роль играет оптимизация работы протоколов связи. Если на физическом уровне целью оптимизации является повышение эффективности использования частотного ресурса (больше битов на 1 Гц), то на вышележащих (канальном, сетевом и транспортном) уровнях — эффективность упаковки пакетного трафика.

При передаче IP-трафика через спутниковые системы связи очень часто используют ускорители TCP. Как известно, этот протокол транспортного уровня предусматривает подтверждение (квитирование) получателем корректного приема определенной порции пакетов («окна TCP»). Если такая квитанция не получена, отправитель снижает скорость передачи следующих блоков информации, полагая, что в сети возникла перегрузка. В системах спутниковой связи, в которых информационные потоки между Центральной Земной Станцией и конечными пользователями преодолевают гигантское расстояние около 80 тысяч километров, такая логика TCP

приводит к уменьшению реальной пропускной способности канала и снижению надежности связи. Для их недопущения обычно используется целый ряд мер. В первую очередь, это алгоритм спуфинга (spoofing), когда квитанции формируются локально, эмулируя подтверждение о корректном принятии пакетов удаленной стороной. Кроме того, применяются «окна» TCP большего размера, механизмы ускоренного установления соединения, восстановления размеров «окна» и другие. Описанные проблемы, возникающие на транспортном уровне спутниковых систем связи, и послужили стимулом для написания данной работы.

ЭТАЛОННАЯ МОДЕЛЬ OSI

Прежде чем описывать протоколы спутниковой связи, необходимо вспомнить базовую модель OSI (The Open Systems Interconnection model). Протоколы важны для связи между объектами, но существует множество опций для их настройки. Более того, для глобальных коммуникаций крайне важно, чтобы протоколы были международно унифицированными. Очевидно, что Международная организация по стандартизации (ISO) сыграла очень важную роль в установке и стандартизации общеизвестной эталонной модели OSI. Благодаря этому любые реализации стеков протоколов, произведённые в согласии с архитектурой эталонной модели, могут объединяться в сеть и взаимодействовать друг с другом.

При разработке нового международного протокола, в принципе, легко прийти к соглашению о том, как определить эталонную модель, но всегда сложно прийти к соглашению о деталях протокола, таких как: количество уровней, которые должна иметь модель, какую длину (в байтах) должен иметь пакет, сколько заголовков должен иметь пакет. Для обеспечения большего количества функций, но в то же время для минимизации накладных расходов (предоставление максимально эффективных и гарантированных услуг, предоставление услуг с установлением соединения или услуг без установления соединения) существует бесконечное количество возможных вариантов реализации протоколов со многими альтернативными технологическими решениями.

ТЕРМИНОЛОГИЯ ПРОТОКОЛОВ

Протокол — это правила и соглашения, используемые в сеансе связи по соглашению между общающимися сторонами. Эталонная

модель OSI предоставляет сторонам общую модель распределения «ролей» так, чтобы все стороны могли бы общаться друг с другом, если они будут следовать этим «ролям», определенным в эталонной или модифицированной модели реализации.

Протоколы необходимы для того, чтобы взаимодействующие стороны могли понимать друг друга, а также чтобы понимать полученную информацию. Международные стандарты важны для достижения глобальной унификации, которая гарантирует взаимодействие аппаратуры и ПО разных производителей. Поэтому протоколы, описанные в стандартах, часто используются в качестве основы для разработки новых. На настоящий момент уже разработано множество различных модификаций этих стандартных протоколов.

Чтобы снизить сложность проектирования систем протоколов, все функции этих систем должны быть разделены на уровни, причём каждый уровень предназначен для предоставления определенных услуг более высоким уровням, ограждая эти уровни от деталей того, как эти услуги фактически будут реализованы.

Каждый уровень имеет интерфейс с примитивными операциями, которые можно использовать для доступа к предлагаемым услугам. Набор уровней и протоколов представляет собой архитектуру сетевого протокола.

Стек протоколов — это список протоколов разного уровня (на одном уровне может быть несколько протоколов). А сущность — это активный элемент на каждом уровне, к примеру, на пользовательских терминалах, коммутаторах и маршрутизаторах. Функции протокола очень разнообразны и могут включать (в зависимости от вида протокола и его уровня): сегментацию и повторную сборку, инкапсуляцию, управление соединением, упорядоченную доставку, управление потоком, контроль ошибок, а также маршрутизацию и мультиплексирование.

ПРИНЦИП РАЗБИЕНИЯ ПРОТОКОЛОВ НА УРОВНИ

Принцип разбиения протоколов на уровни — это важная концепция сетевых протоколов и эталонных моделей. В 1980-х годах ISO разработала семиуровневую эталонную модель (рис. 1), которая носит название «эталонная модель взаимодействия открытых систем» (OSI). Данная модель основана на ясных и простых принципах.

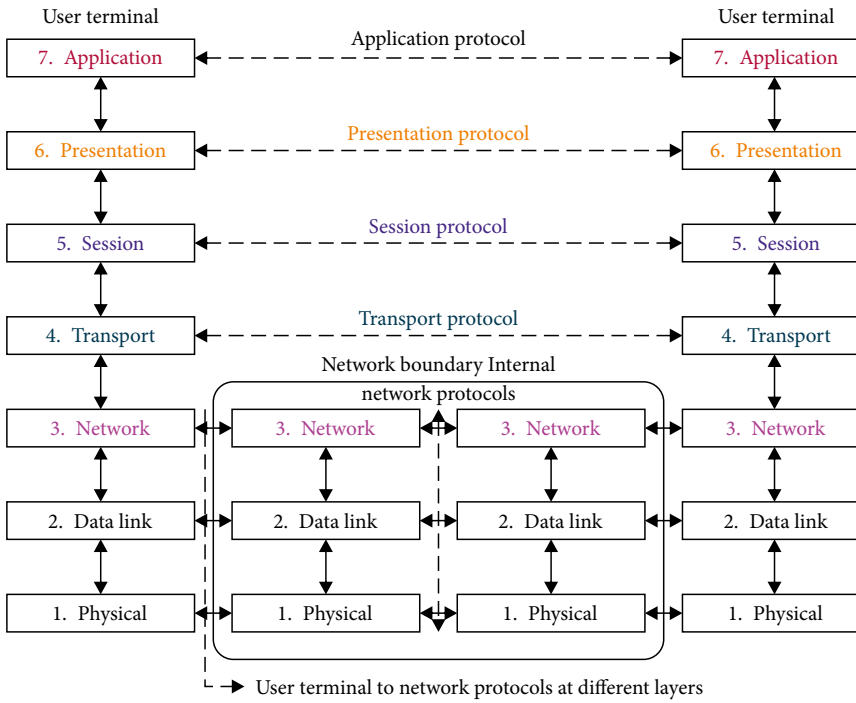

Рисунок 1. Семиуровневая эталоная модель OSI/ISO

7 — Прикладной (Application layer); 6 — Уровень представления данных (Presentation layer); 5 — Сеансовый уровень (Session layer); 4 — Транспортный уровень (Transport layer); 3 — Сетевой уровень (Network layer); 2 — Уровень каналов данных (Data link layer); 1 — Физический уровень (Physical layer)

Стоит сказать, что это была первая полная эталонная модель, разработанная в качестве международного стандарта. Принципы, которые были применены для достижения семи уровней, можно резюмировать следующим образом:

— Слой определяет уровень абстракции, который отличается от любого другого уровня.
— Каждый уровень выполняет чётко определённую функцию.
— Функция каждого уровня должна быть выбрана так, чтобы вести к протоколам, стандартизированным на международном уровне.
— Границы слоя следует выбирать так, чтобы минимизировать поток информации через интерфейс.
— Количество слоёв должно быть достаточно большим, но не слишком большим.

ФУНКЦИИ РАЗЛИЧНЫХ УРОВНЕЙ

Уровень 1 — физический уровень (поток битов) определяет механические, электрические и процедурные интерфейсы, а также физическую среду передачи. В спутниковых сетях физической средой передачи является радиосигнал. Модуляция и канальное кодирование позволяют передавать битовый поток в определенных сигналах и на определённых выделенных полосах частот.

Уровень 2 — канальный уровень передачи данных обеспечивает линию, которая свободна от необнаруженной передачи ошибки на сетевом уровне. У средств вещания есть одна проблема на канальном уровне — проблема контроля доступа к общей среде. Особый подслой, называемый уровнем доступа к среде (MAC), наряду с методами доступа Aloha, FDMA, TDMA, CDMA, DAMA, призван устранять эту проблему.

Уровень 3 — сетевой уровень маршрутизирует пакеты от источника к месту назначения. Функции данного уровня включают: се-

тевую адресацию, контроль перегрузки, учёт, разборку и повторную сборку, работу с гетерогенными сетевыми протоколами и технологиями. В широковещательных сетях проблема маршрутизации проста: протокол маршрутизации часто бывает несовершенен или даже отсутствует.

Уровень 4 — транспортный уровень обеспечивает доставку данных для пользователей верхних уровней (сервис пользовательских данных). При необходимости, транспортный уровень выполняет функции надежной доставки данных, контроля ошибок, управления потоком данных (контроля перегрузок).

Уровень 5 — сеансовый уровень обеспечивает средства взаимодействия объектов представления для организации и синхронизации их диалога, а также для управления обменом данными.

Уровень 6 — уровень представления связан с преобразованием и форматированием данных, а также и с синтаксисом данных.

Уровень 7 — прикладной уровень является высшим уровнем архитектуры OSI. Он обеспечивает сервисы для прикладных процессов.

Сегодня мы видим развитие множества типов новых приложений, сервисов, сетей и средств передачи информации. Никто не ожидал такого быстрого развития Интернета и порожденных им новых технологий (5G, Интернет Вещей, Blockchain, Wi-Fi, Облачные технологии). Технологическая конкуренция приводит к разнообразию технологий и не способствует унификации стандартов. Существует также и много других причин, включая технические, политические и экономические. Стоит отметить, что эталонная модель почти не используется в современных сетях. Но принципы многоуровневого протокола всё ещё широко используются в проектировании и реализации стеков протоколов и интерфейсов в различных сетях. Также стоит сказать, что современные разработчики стараются использовать классическую модель OSI в качестве справочника для описания функций протоколов.

ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ

Далее в работе будут описаны протоколы транспортного уровня, применяемые в современных системах спутниковой связи, а также алгоритмы контроля перегрузки и восстановления потерь стандартного протокола TCP (Transmission Control Protocol). Будут описаны компромиссы, подразумеваемые при использовании низкоуровневых смягчений для оптимизации производительности TCP, а также будут показаны наиболее многообещающие улучшения TCP, наглядно демонстрирующие, как можно эффективно решать различные сетевые проблемы спутникового соединения на транспортном уровне. Другие широко распространенные методы, основанные на внедрении PEP на транспортном уровне, будут также описаны.

Большая часть этого раздела взята из справочника [12], где описаны алгоритмы «медленного старта» (SS), «предотвращения перегрузки» (CA), «быстрой повторной передачи» и «быстрого восстановления», а также из другого справочника [13], где рассмотрены алгоритмы с политикой «таймаута повторной передачи».

Таблица 1. Определение главных параметров протокола TCP

ПАРАМЕТР	ОПРЕДЕЛЕНИЕ
Segment	Сегмент — это ЛЮБЫЕ данные TCP/IP или пакет подтверждения (или и то, и другое).
Congestion window (cwnd)	Окно перегрузки — переменная состояния TCP, ограничивающая объём отправляемых данных, которые отправитель может передать до получения подтверждения (ACK). Различные реализации поддерживают объём в байтах или сегментах.

ПАРАМЕТР	ОПРЕДЕЛЕНИЕ
Flight size (FS)	Объем данных, которые были отправлены, но еще не подтверждены. FS идентифицирует сегменты, которые все еще «летают» внутри сети.
Full-sized segment	Сегмент, содержащий максимальное разрешённое количество байтов данных (например, байты данных SMSS).
Initial window (IW)	Начальное окно, размер окна перегрузки отправителя после завершения трёхстороннего рукопожатия
Loss window (LW)	Размер окна перегрузки после того, как отправитель TCP обнаруживает потерю, используя свой таймер повторной передачи
Receiver maximum segment size (RMSS)	Размер самого большого сегмента, который может принять приемник
Receiver window (rwnd)	Последнее объявленное окно получателя: это ограничение на количество необработанных данных на стороне получателя.
Restart window (RW)	Размер окна перегрузки после перезапуска передачи TCP после периода простоя
Round-trip time (RTT)	Время приёма-передачи — это время, прошедшее с момента передачи сегмента и получения соответствующего ACK
Segment Sender maximum segment size (SMSS)	Размер самого большого сегмента, который отправитель может передать (зависит от типа используемой сети и других факторов). Представляет из себя любые данные TCP / IP или пакета подтверждения (в байтах)
ssthresh	Это пороговое значение размера окна перегрузки (в сегментах)

МЕДЛЕННЫЙ СТАРТ

Цель алгоритма медленного старта — зондировать сеть, чтобы проверить доступную пропускную способность, постепенно увеличивая Congestion window (cwnd), вместо того, чтобы начинать с высокого фиксированного значения, которое может вызвать перегрузку. Начальное окно (IW), начальное значение cwnd, должно быть меньше или равно двукратному SMSS в байтах. Для ускорения фазы запуска TCP можно использовать увеличенное Начальное окно:

$$IW = \min(4 \times SMSS, \max(2 \times SMSS, 4380 \text{ bytes})) \quad (1)$$

Начальное значение ssthresh может быть произвольно большим, но сразу же уменьшается в ответ на первый эпизод перегрузки. Алгоритм медленного старта используется при cwnd < ssthresh. Во время медленного запуска TCP увеличивает cwnd на байты SMSS для каждого полученного ACK, подтверждающего новые данные:

$$cwnd = cwnd + SMSS \quad (2)$$

Фаза медленного старта завершается, когда cwnd превышает ssthresh или когда пакет теряется.

ПРЕДОТВРАЩЕНИЕ ПЕРЕГРУЗКИ

Алгоритм предотвращения перегрузки используется, когда cwnd больше, чем ssthresh, и продолжается до обнаружения перегрузки. Во время предотвращения перегрузки для каждого входящего неповторяющегося ACK значение cwnd в байте обновляется в соответствии со следующим правилом:

$$cwnd = cwnd + SMSS \times SMSS / cwnd \quad (3)$$

Эта формула определена реализациями, которые поддерживают cwnd в единицах полноразмерных сегментов, а не в байтах, что считается более удобным (увеличить cwnd на 1 полноразмерный сегмент за RTT). Когда отправитель TCP обнаруживает потерю сегмента с помощью таймера повторной передачи, значение ssthresh не должно быть больше значения, указанного в следующей формуле, где FlightSize — это количество еще не подтвержденных данных в сети:

$$sstresh = max(FlightSize/2, 2 \times SMSS) \quad (4)$$

Кроме того, по истечении RTO (как поясняется ниже) для cwnd должно быть установлено значение не более, чем окно потерь LW, которое равно одному полноразмерному сегменту (независимо от значения IW). Следовательно, в этом случае после повторной передачи отброшенного сегмента отправитель TCP использует алгоритм медленного старта до тех пор, пока не будет достигнуто новое значение ssthresh, после чего снова вступит в действие предотвращение перегрузки.

БЫСТРАЯ РЕТРАНСЛЯЦИЯ И БЫСТРОЕ ВОССТАНОВЛЕНИЕ

Получатель TCP должен немедленно отправить дублированный ACK, когда прибывает неупорядоченный сегмент, указав на то, какой порядковый номер на самом деле ожидается. С точки зрения отправителя, повторяющиеся ACK могут быть следствием сетевых проблем (например, отброшенные сегменты, изменение порядка данных, репликация данных). Очевидно, что получатель TCP отправит немедленный и совокупный ACK, когда входящий сегмент заполнит пробел в пространстве последовательности. Отправитель TCP использует алгоритм Fast Retransmit для обнаружения и устранения потери, вызванной поступлением трех повторяющихся ACK. После входа в режим быстрой повторной передачи TCP выполняет повторную передачу того, что кажется недостающим сегментом, не дожидаясь истечения таймера повторной передачи. Затем алгоритм быстрого восстановления управляет восстановлением данных до тех пор, пока не будет подтвержден последний переданный сегмент при входе в режим быстрой повторной передачи. Этот аспект в основном отличает TCP Reno и NewReno (стандарт описан в 2004 г.) от TCP Tahoe и Old Tahoe (алгоритм управления потоком, ранний ва-

риант TCP) [12]. Алгоритмы быстрой повторной передачи и быстрого восстановления обычно реализуются вместе (TCP Reno) следующим образом:

1. Когда получен третий дублирующий ACK, значение ssthresh устанавливается на значение, указанное в формуле (4).
2. Потерянный сегмент передается повторно, и для cwnd устанавливается значение ssthresh плюс 3×SMSS.
3. Для каждого полученного дополнительного дубликата ACK cwnd увеличивается на SMSS.
4. Сегмент передается, если это разрешено новым значением cwnd и значением рекламируемого окна получателя.
5. Когда приходит следующий ACK, подтверждающий новые данные, для cwnd устанавливается значение ssthresh (значение, установленное на шаге 1).

Существует еще одно различие между спецификациями Reno и NewReno. Спецификация TCP Reno завершает фазу быстрого восстановления после того, как ACK подтверждает новые полученные данные, а затем переходит в фазу предотвращения перегрузки. А TCP NewReno, напротив, различает частичное и полное подтверждение. В первом случае полученный ACK не охватывает все отправленные данные и, следовательно, указывает на потери дальнейшего пакета. В этом случае фаза быстрого восстановления не завершается, и TCP NewReno реагирует повторной передачей первого неподтвержденного сегмента и аннулированием окна перегрузки. В последнем случае при получении полного подтверждения TCP NewReno ведет себя как TCP Reno. Цель модификации NewReno — избежать ненужных многократных сокращений cwnd при наличии всплеска потерь. Однако, в Reno можно восстановить не более одного сегмента за RTT. Чтобы преодолеть это ограничение, может быть принята опция избирательного подтверждения TCP (SACK) [15], которая позволяет восстанавливать множественные потери в пределах периода RTT с момента передачи сегмента.

ТАЙМ-АУТ ПОВТОРНОЙ ПЕРЕДАЧИ

TCP использует таймер повторной передачи, чтобы гарантировать доставку данных в отсутствие какой-либо обратной связи от удалённого приёмника данных. Этот таймер обеспечивает период ожидания, продолжительность которого обозначается RTO (таймаут повторной передачи). RTO возникает, когда отправитель пропускает слишком много подтверждений и решает взять тайм-аут и полностью прекратить отправку. По прошествии некоторого времени, обычно не менее одной секунды, отправитель осторожно начинает отправку снова, сначала с одним пакетом, затем двумя пакетами и так далее. Значение RTO обычно используется для определения числа попыток, а также тайм-аута повторной передачи неподтверждённых TCP-пакетов по уже установленному TCP-соединению, после истечения которого передача сбрасывается. Значение RTO определяется на основе постоянно проводимых измерений RTT. В вычислении RTO участвуют специальные величины SRTT (сглаженное RTT), RTTVAR (вариация RTT) и степень детализации таймера *G*.

Правила, определяющие вычисление SRTT, RTTVAR и RTO, следующие [13]:

1. До тех пор, пока не измерено RTT для отправленного сегмента, отправитель устанавливает RTO равным 3 секунды.
2. Когда сделано первое измерение RTT (*R*), хост устанавливает SRTT равным *R*, RTTVAR равным *R*/2, а RTO равным $R + \max(G, K \times \text{RTTVAR})$, где $K=4$.
3. Когда сделано следующее измерение RTT, хост отправляет:
   — Новое RTTVAR,
   — Новое SRTT.

На основе вычисления значения RTO таймером ретрансляции управляют следующие процедуры:

1. Для каждой передачи данных (или повторной передачи), если таймер не работает, он запускается так, чтобы он истёк через RTO секунд.
2. Когда все ожидающие данные подтверждены, таймер повторной передачи отключается.
3. Когда получен недублированный ACK, таймер перезапускается так, что он истекает через RTO секунд.

По истечении таймера повторной передачи:

1. Самый ранний сегмент, который не был подтвержден получателем TCP, будет ретранслирован.
2. Хост выполнит сброс таймера, устанавливая RTO на RTO×2 («откат таймера»). Это значение RTO сохранится до тех пор, пока не будет доступна новая оценка RTT или до истечения срока действия нового RTO.

ВЗАИМОДЕЙСТВИЕ TCP С ПРОТОКОЛАМИ НИЖЕЛЕЖАЩЕГО УРОВНЯ

Производительность TCP в спутниковых каналах связи можно улучшить за счёт оптимизации взаимодействия с нижними уровнями. Здесь стоит рассмотреть два важных примера: анализ пути максимальной единицы передачи (MTU, *Maximum Transfer Unit*) и оптимизация модуляции и кодирования прямой коррекции ошибок (FEC, *Forward Error Correction*).

ПРОЦЕДУРА АНАЛИЗА ПУТИ MTU

Анализ пути MTU (процедура) направлен на определение максимального размера пакета, который можно использовать по заданному соединению без фрагментации IP. Обычно, если пакет слишком большой для пересылки, шлюз фрагментирует пакет и пересылает фрагменты. Но используя механизмом обнаружения пути MTU, сообщение протокола ICMP возвращается отправителю и указывает на то, что исходный пакет не может быть передан без фрагментации, а также сообщает о самом большом размере пакета, который может быть отправлен шлюзом. Таким образом, накладные расходы снижаются, и отправитель TCP может увеличивать окно перегрузки быстрее (в байтах) [16]. Стоит отметить, что процедура анализа пути MTU обычно применяется при наличии TCP-соединений, но может выполняться и независимо от TCP. А вот описанная ниже модуляция и оптимизация кодирования FEC, напротив, потребует знаний о производительности TCP. То есть со-

вместная, более сложная оптимизация, затрагивающая как физический, так и транспортный уровень, должна выполняться специальным образом.

ОПТИМИЗАЦИЯ МОДУЛЯЦИИ И КОДИРОВАНИЯ FEC

Очень часто PER (*Packet Error Rate*) из-за ошибок беспроводного канала связи влияет на TCP соединение, что оказывает более серьезный эффект, чем при передаче потоковых данных, из-за взаимодействия с базовым механизмом контроля перегрузки TCP. Из-за такой высокой чувствительности крайне важно уменьшить PER, используя один или несколько методов: кодирование прямого стирания [17] на уровне пакетов (вариант FEC), автоматический запрос повтора (ARQ) на канальном уровне, коды FEC и адаптивная модуляция.

ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОИЗВОДИТЕЛЬНОСТЬЮ TCP И FEC

Коэффициент ошибок по битам (BER, *Bit Error Rate*) на физическом уровне зависит от различных параметров, большинство из которых не может быть легко изменено после того, как спутниковая система уже находится в эксплуатации. В частности, полоса пропускания, доступная для радиосигнала, часто фиксирована, как и максимальная передаваемая мощность. В настоящее время можно легко изменить схему FEC, которая обычно используется на физическом уровне, а также (в некоторых случаях) метод модуляции (например, количество уровней в фазовых модуляциях M-PSK или M-QAM). Фактически, для метода прямой коррекции ошибок устойчивость к ошибкам увеличивается с увеличением количества введённой избыточности, то есть за счёт уменьшения кодовой ско-

рости (R_c). Аналогично, для модуляций M-PSK и M-QAM производительность с точки зрения вероятности битовой ошибки обычно улучшается за счет уменьшения мощности совокупности сигналов M и, следовательно, количества битов на символ модуляции, заданного как $\log_2 M$. В обоих случаях платой за более низкий BER является снижение эффективности использования доступного спектра [18].

УМЕНЬШЕНИЕ КОЛИЧЕСТВА ОШИБОК В TCP-ПАКЕТАХ

Для спутниковой системы с ограниченной шириной полосы физического канала B (в Гц) и максимальной передаваемой мощностью C (с некоторой степенью гибкости в выборе FEC и схемы модуляции) можно найти компромисс между скоростью передачи информации, доступной для TCP, и PER. Хотя это интуитивно понятно, условия такого технического компромиса должны быть чётко определены на каждом этапе. Во-первых, имея дело с фазовыми модуляциями M-PSK или M-QAM, ширина полосы B канала приблизительно численно совпадает со скоростью B_s символов сигнала. Таким образом, эффективность модуляции η, определяемая как отношение скорости передачи битов к занимаемой полосе сигнала B, совпадает с количеством битов на символ. Доступная скорость передачи информационных битов B_r может быть получена как функция символьной скорости B_s следующим образом:

$$B_r = R_c \eta B_s , \tag{5}$$

где R_c — кодовая скорость, а B_s — эффективность модуляции.

Что касается ограничения на мощность C принятого сигнала, его можно напрямую преобразовать в ограничение на полученную энергию, приходящуюся на один бит информации, как $E_b = C/B_r$.

Затем, разделив оба члена на значение шума спектральной плотности N_o, эту формулу удобно переписать в виде отношения сигнал/шум:

$$E_b/N_o = (C/N_o)/R_c \eta B_s \quad (6)$$

Заключительный шаг состоит в рассмотрении связи между BER и отношением E_b/N_0, которое зависит от модуляции и принятой схемы FEC. В частности, мы можем наблюдать, что хорошо известные графики производительности (BER относительно E_b/N_0), представленные в некоторых источниках [19], показывают уменьшение BER для того же отношения E_b/N_0, если η (внутри той же схемы модуляции) или R_c (внутри того же семейства кодов) уменьшаются, потому что схема модуляции и кодирования становится более устойчивой к шуму.

Завершить этот раздел стоит соображениями относительно применимости только что описанной оптимизации модуляции и кодирования к недавнему стандарту цифрового телевидения второго поколения DVB-S2 [20].

DVB-S2 рассматривает очень мощные коды с исправлением ошибок, так что оптимизация, основанная на канальном кодировании, была бы бесполезной, потому что кривые, которые дают PER относительно E_b/N_0, настолько крутые, что мы имеем своего рода поведение включения-выключения физического канала: либо коэффициент пакетных ошибок (PER) мал, либо настолько высок, что снижает производительность TCP. Однако при использовании спутникового обратного канала [21] оптимизация обратного канала может быть действительно полезной. Дополнительная возможность состояла бы в том, чтобы встроить стирающий код (помехоустойчивый код, способный восстановить целые пакеты данных в случае их потери) [22, 23] канального уровня непосредственно над уровнем MAC,

что могло бы стать полностью рабочим решением на программном уровне, независимо от характеристик используемого оборудования.

УЛУЧШЕНИЯ TCP

В последние годы было предложено несколько решений для повышения производительности TCP как по проводным, так и по беспроводным каналам связи, включая спутниковые соединения. Некоторые предложения сосредоточены на ограниченных модификациях стандартных процедур или настройке параметров TCP [24, 25], в то время как другие предусматривают введение альтернативных процедур TCP, которые заменяют стандартные алгоритмы или дополняют их. Что касается первой категории, то имеются преимущества большего начального окна медленного старта (SS-IW) [14], и иногда предусмотрены модификации порога медленного старта и интервала между пакетами для предотвращения раннего переполнения буфера [26]. В некоторых справочниках [15] предлагаются две незначительные модификации алгоритма быстрой повторной передачи, как, например, принятие опции SACK (избирательный ACK) для быстрого восстановления после множественных потерь в окне данных. Хотя все эти усовершенствования безусловно полезны, они недостаточны для реального преодоления проблем, создаваемых беспроводными и особенно спутниковыми линиями связи, которые требуют более значительных улучшений. Количество предложений по модификаций протокола TCP довольно велико, включая, среди прочего: TCP Vegas, TCP Peach, TCP Westwood и TCP Hybla.

Однако, что касается протокола TCP Vegas [27], следует отметить, что он не решает основные проблемы, связанные именно со спутниковыми соединениями, поэтому его применение в спутниковой среде вообще не кажется многообещающим. Напротив, другие упомянутые усовершенствования TCP пытаются решить или, по крайней мере, смягчить снижение производительности, которое про-

исходит из-за длительного времени прохождения сегмента (RTT) или случайных ошибок, поэтому их конкретные цели и характеристики будут подробно рассмотрены ниже. Наконец, некоторое внимание будет уделено протоколу передачи управления потоком (SCTP) [5], который изначально был разработан для транспортировки пакетов сигнализации для приложений передачи голоса по IP, но постепенно стал реализовываться как общий транспортный протокол. Хотя SCTP и нельзя строго рассматривать как расширение TCP, поскольку он является альтернативой TCP, он всё же включен в этот раздел, поскольку большинство его процедур тесно связаны с базовыми алгоритмами TCP.

Тут также стоит упомянуть о проблемах, свойственных всем протоколам семейства TCP. Фактически, установив для начального ssthresh произвольное значение, TCP может столкнуться с двумя разными проблемами:

1. Если начальный ssthresh слишком низкий по сравнению с задержкой полосы пропускания, то TCP преждевременно переключается на этап предотвращения перегрузки, что приводит к неэффективному использованию полосы пропускания.
2. Если начальное ssthresh слишком велико, экспоненциальное увеличение окна перегрузки в фазе медленного запуска может вызвать множественные потери на узком маршрутизаторе с серией длительных фаз восстановления, часто сопровождаемых тайм-аутами.

TCP PEACH

TCP Peach состоит из двух новых алгоритмов, а именно алгоритма внезапного запуска и быстрого восстановления, а также двух традиционных алгоритмов TCP: предотвращения перегрузки и быстрой повторной передачи [2]. Внезапный запуск и быстрое восстановление предназначены для замены, соответственно, алго-

ритмов медленного запуска и быстрого восстановления стандартного TCP. Новые алгоритмы основаны на новой концепции использования фиктивных сегментов для проверки доступности сетевых ресурсов без передачи какой-либо новой информации отправителю. Фиктивные сегменты — это сегменты с низким приоритетом, генерируемые отправителем как копия последнего переданного сегмента данных, то есть они не несут никакой новой информации принимающей стороне. Отправитель же использует фиктивные сегменты для проверки доступности сетевых ресурсов. Если маршрутизатор на пути подключения перегружен, он сначала отбрасывает IP-пакеты, содержащие фиктивные сегменты. Следовательно, передача фиктивных сегментов не вызывает снижения пропускной способности фактических сегментов данных, то есть традиционных сегментов. Если маршрутизаторы не перегружены, то фиктивные сегменты смогут достичь получателя. Отправитель интерпретирует ACK для фиктивных сегментов как свидетельство того, что в сети есть неиспользуемые ресурсы, и может соответственно увеличить скорость передачи. Используя фиктивные сегменты, внезапный запуск обеспечивает быстрое открытие окна перегрузки в начале соединений независимо от фактического RTT, в то время как быстрое восстановление может противодействовать случайным ошибкам из-за зашумлённого канала. Основное требование TCP Peach — это реализация (в сетевых маршрутизаторах) некоторых форм политики организации очередей на основе классов (CBQ), чтобы различать трафик с высоким и низким приоритетом (между пакетами данных и фиктивными сегментами).

Преимущества TCP Peach в спутниковой среде с большими потерями изображены на рис. 2, где поведение протоколов TCP Peach и TCP NewReno сравнивается по динамике передачи сегментов TCP Peach. Можно заметить, что в начале передачи TCP Peach работает в фазе внезапного запуска, то есть идёт передача фиктивных сегментов для проверки доступного сетевого ресурса и, таким образом,

это гарантирует максимальную скорость передачи данных, достижимую на спутниковом канале.

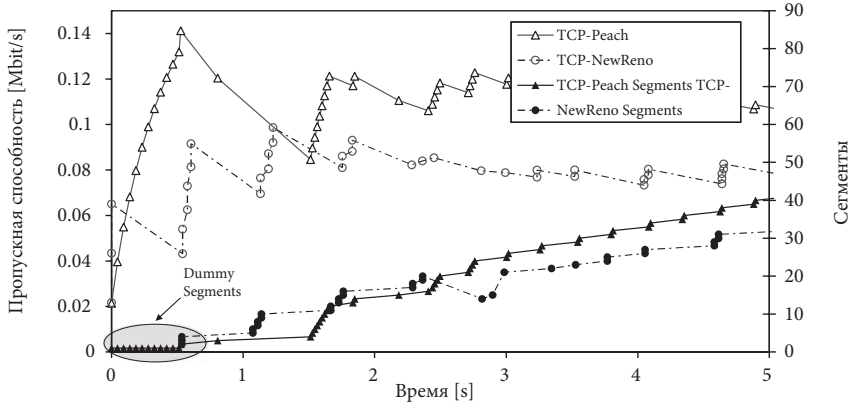

Рисунок 2. Поведение TCP Peach; RTT = 520 ms, PER = 1%, ёмкость канала = 2 Mbit/s, размер сегмента TCP = 1448 bytes

Когда потеря пакета обнаруживается с помощью трёх дублированных ACK, TCP Peach выполняет алгоритм быстрой повторной передачи, как это сделали бы протоколы TCP Reno и TCP NewReno. После завершения ретрансляции происходит этап быстрого восстановления: в этом случае, в отличие от стандартного TCP, для каждого полученного ACK отправляются два фиктивных сегмента для оценки доступной пропускной способности сети. Это различие является ключевым элементом повышения производительности по сравнению со стандартным TCP. Преимущество состоит в том, что при перезапуске передачи, управляемой алгоритмом предотвращения перегрузки TCP Peach, поступление ACK, подтверждающих фиктивные сегменты, переданные во время фазы быстрого восстановления, вызывает быстрое увеличение окна перегрузки, тем самым полоса пропускания канала используется более эффективно, чем в стандартном TCP.

TCP WESTWOOD

TCP Westwood [45, 28] был введён с целью ограничения последствий потерь, вносимых беспроводным каналом. С этой целью TCP Westwood представляет модификацию алгоритма быстрого восстановления. Вместо того, чтобы уменьшить вдвое окно перегрузки после трёх дублированных ACK и зафиксировать порог медленного старта на этом значении, TCP Westwood устанавливает пороговое значение размера окна перерузки (ssthresh) как функцию от предполагаемой доступной полосы пропускания. Таким образом, потери в канале не вызывают резкого замедления скорости передачи, типичного для стандартных версий. Пропускная способность оценивается путём измерения и усреднения скорости возврата ACK. В частности, прием ACK в момент времени t_k означает, что получатель TCP получил объем данных dk.

Следовательно, *k*-я выборка полосы пропускания, используемая данным соединением, равна:

$$b_k = \frac{d_k}{t_k - t_{k-1}} \tag{7}$$

где t_{k-1}—время поступления предыдущего ACK. Наконец, TCP Westwood использует фильтр сглаживания дискретного времени для оценки доступной полосы пропускания.

В случае маршрутизаторов со встроенной стратегией обрубания хвоста очереди, где может наблюдаться прерывистый трафик, такой метод оценки пропускной способности имеет тенденцию к несправедливому разделению полосы пропускания. Чтобы достичь высокой степени использования канала и в то же время гарантировать эффективность соединения, TCP Westwood реализует новый механизм управления перегрузкой, основанный на оценке допустимой скорости (RE), а не на оценке полосы пропускания [29]. Вкратце, альтернативная выборка доступной полосы пропускания

определяется как количество данных, доставленных за последний интервал времени *T*, делённое на *T*. Кроме того, в обновленной версии Westwood [30] реализован еще один механизм, называемый «адаптивный запуск» (ASTART), для настройки ssthresh на этапе запуска в соответствии с оценкой пропускной способности.

TCP HYBLA

Протокол TCP Hybla [4, 31] был задуман с целью противодействия ухудшению производительности, вызванного длительным RTT, типичным для спутниковых соединений. Протокол состоит из набора процедур, которые включают: усовершенствование стандартных алгоритмов управления перегрузкой как для фаз медленного запуска, так и для фазы предотвращения перегрузки, обязательное принятие принципа SACK, принятие оценки пропускной способности канала, использование временных меток и реализация методов разделения пакетов.

Модификация стандартных правил управления перегрузкой продиктована идеальной целью TCP Hybla — получить для соединений с длинным RTT ту же мгновенную скорость передачи сегмента, что и для сравнительно быстрого эталонного TCP-соединения (проводного). Принятие непрерывной модели для описания эволюции окна перегрузки во времени, $W(t)$ (выраженное в единицах SMSS), оказалось очень важным, что позволило строго определить скорость передачи сегмента (т. е. количество сегментов, передаваемых в секунду):

$$B(t) = W(t)/\mathrm{RTT} \qquad (8)$$

Уравнение (8) показывает, что для достижения вышеупомянутой цели — скорости передачи, независимой от фактического RTT, — требуется более высокая скорость $W(t)$ для соединений

с длинным RTT, в отличие от того, что фактически происходит в стандартном TCP, где длинное RTT приводит к более медленному увеличению скорости. Перед представлением новых правил управления перегрузкой необходимо ввести новый параметр — нормализованный RTT (ρ), определяемый как отношение между фактическим RTT и RTT эталонного соединения, к которому TCP Hybla стремится уравнять производительность, обозначенную как RTT_0:

$$\rho = RTT/RTT_0 \quad (9)$$

В справочных материалах показано, что тот же $B(t)$ эталонного соединения может быть достигнут с помощью соединений с наиболее длинным RTT при условии, что стандартные алгоритмы управления перегрузкой заменены на правила управления перегрузкой TCP Hybla.

Обратим внимание, что правило обновления для предотвращения перегрузки аналогично алгоритму постоянной скорости [31], который, с другой стороны, не учитывает фазу медленного старта. Другое важное отличие состоит в том, что в фактической реализации TCP Hybla устанавливается минимальное значение для ρ, равным 1, чтобы избежать замедления соединений, которые уже являются «быстрыми» (то есть соединения с $RTT < RTT_0$). Оценка пропускной способности используется для правильной установки начального ssthresh.

Что касается других функций, то поскольку алгоритм контроля перегрузки TCP Hybla намного эффективнее стандартного TCP в обеспечении удовлетворительной скорости передачи для соединений с длинным RTT, следовательно, необходимо большее среднее значение cwnd.

В результате многократные потери в одном окне будут более частыми, что предполагает обязательную политику SACK. Большие окна перегрузки также часто вызывают серьезные недостатки по-

литики «экспоненциального обратного отсчета» RTO. Этого можно избежать, используя временные метки [13], которые в настоящее время в значительной степени реализованы в наиболее распространенных операционных системах. Наконец, в Hybla предусмотрено использование интервалов между пакетами [32], чтобы противодействовать пакетности из-за больших окон перегрузки, с вытекающими возможными потерями в промежуточных очередях маршрутизатора.

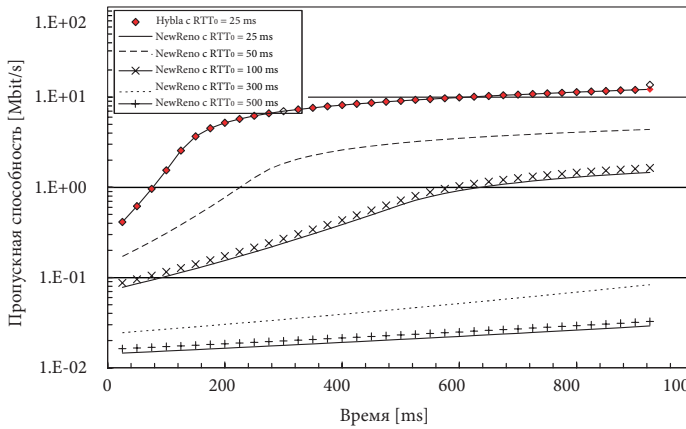

Рисунок 3. TCP Hybla в сравнении со стандартным TCP: независимость производительности TCP Hybla от фактического значения RTT; начальный ssthresh = 32 kB, PER = 0, размер сегмента TCP = 1448 bytes; аналитические данные, относящиеся к идеальному каналу.

На рис. 3 показано различное поведение TCP Hybla и TCP NewReno в присутствии разных RTT с учетом количества переданных данных с момента начала соединения на идеальном канале (аналитические результаты). В примере, который рассматривает начальную фазу медленного старта, за которой следует фаза предотвращения перегрузки (ssthresh = 32 кбайта), RTT_0 был установлен на 25 мс для целей сравнения, чтобы показать, что идеальная про-

изводительность TCP Hybla равна производительности соединения NewReno с RTT = 25 мс независимо от фактического RTT. Эта теоретическая независимость производительности TCP Hybla от RTT показана на рисунке перекрытием маркеров TCP Hybla (одинаковых для каждого RTT) с непрерывной линией NewReno 25. Другие кривые TCP NewReno показывают зависимость производительности от RTT. Конечно, выбор «правильного» значения для RTT_0 для TCP Hybla может быть проблемой, по крайней мере, в открытых сетях, где задержки на пути наземных соединений значительно различаются [4].

SCTP

Протокол управления потоком передачи (SCTP, Stream control transmission protocol) [5, 33] был разработан для выполнения передачи данных мультимедиа в реальном времени по IP-сетям для приложений передачи голоса по IP (VoIP). Однако вскоре было замечено, что SCTP можно успешно использовать в более широком диапазоне приложений. В частности, некоторые функции могут быть полезны для преодоления ограничений среды передачи спутниковых сигналов.

Архитектура SCTP сохраняет некоторые базовые стандартные функции TCP, такие как оконное управление перегрузкой, обнаружение ошибок и повторная передача, которые оказались надёжными в среде Интернета. Помимо внесения в них некоторых улучшений, как описано ниже, SCTP также добавляет несколько новых функций, которые недоступны в стандартном TCP, в частности, множественную адресацию и многопоточность. Множественная адресация позволяет двум конечным точкам устанавливать связь с несколькими IP-адресами для каждой конечной точки (в SCTP «ассоциация» — это имя для связи между конечными точками). Таким образом, можно ввести своего рода пространственное разне-

сение в сети, что может быть полезно для устранения отказов долгих соединений без прерывания передачи данных. Например, спутниковый канал можно использовать в качестве вторичного пути для резервного копирования наземного канала (первичный путь). Тем не менее, управление процессом передачи обслуживания является довольно сложным, поскольку время, необходимое для обнаружения отказа канала связи, приводит к большой задержке передачи обслуживания [33].

Многопоточная передача используется для устранения некоторых проблем, связанных со строгой политикой доставки байтов TCP. В SCTP каждый поток является подпотоком в общем потоке данных, и доставка каждого подпотока не зависит от других. В SCTP также есть улучшения, связанные с проблемами безопасности.

В завершении описания протокола стоит подробно описать основные усовершенствования, внесенные в стандартные механизмы TCP, а именно:

— SCTP включает алгоритм быстрой повторной передачи, основанный на отчётах о пропусках SACK, аналогично протоколу TCP SACK. В отличие от стандартного TCP, использование SACK является обязательным (как и в TCP Hybla), что позволяет повысить производительность в случае множественных потерь из одного окна данных.

— Во время медленного запуска или предотвращения перегрузки SCTP окно перегрузки увеличивается с учетом количества подтвержденных байтов, а не количества полученных ACK, что повышает точность управления перегрузкой.

— Во время предотвращения перегрузки SCTP окно перегрузки может быть увеличено только при использовании полностью. Этого ограничения в стандартном TCP нет. Однако следует отметить, что аналогичный механизм присутствует в реализациях TCP для Linux и TCP Hybla.

ПРОКСИ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ

Недостатки, возникающие из-за применения служб на основе TCP в спутниковой связи, обсуждались в предыдущих разделах вместе с некоторыми контрмерами, которые действительно могут повысить производительность TCP. Однако в большинстве сценариев необходимо сохранить стек протоколов на конечных терминалах без изменений, по крайней мере, в краткосрочной перспективе. Следует отметить, например, что в наиболее распространенных коммерческих операционных системах пользователь практически не может контролировать используемый вариант TCP. В этом случае оптимизация производительности может быть осуществлена только путем воздействия на спутниковый сегмент сети. С этой целью в конце девяностых было проведено большое количество исследований, в результате которых был разработан набор возможных архитектур и схем, способных уравновесить недостатки TCP и обеспечить оптимизацию производительности за счёт увеличения сложности оборудования и (по крайней мере, в некоторых случаях) также серьёзных нарушений семантики. Эти решения обычно называют прокси повышения производительности (PEP, *Performance-Enhancing Proxy*) [1]. В этом разделе представлены наиболее многообещающие методы PEP (а именно, спуфинг и разделение), для которых описана архитектура протокола, в которой используются специализированные решения транспортных протоколов для спутниковых каналов. Завершается раздел кратким обзором других основных методов, обычно используемых в архитектурах TCP PEP.

TCP SPOOFING

TCP spoofing — это эффективное решение для смягчения снижения производительности, связанного с длительными RTT [9]. Оно заключается в ускорении роста окна передачи TCP за счёт маскировки длительной задержки, наблюдаемой в спутниковом канале (рис. 4). Более конкретно, спутниковый шлюз прозрачно для конечных хостов отвечает за автоматическую генерацию поддельных ACK в соответствии с входящими сегментами TCP. Таким образом, задержка, необходимая хостам для передачи новых сегментов, является единственной задержкой, испытываемой наземным каналом (за исключением спутниковой части). Главный недостаток этого подхода: сквозная семантика больше не соблюдается, что создаёт проблемы для приложений, которые, наоборот, полагаются на эту характеристику. Более того, принятие этой схемы подразумевает, что «истинные» ACK, генерируемые хостом назначения, должны быть перехвачены спутниковым шлюзом и затем подавлены, чтобы избежать ложных дублированных ACK, поступающих в сторону отправителя. В случае потерянных сегментов шлюз также будет нести ответственность за повторную передачу пропущенных сегментов. Для этой задачи также необходимо выполнить симметричную маршрутизацию; другими словами, сегменты данных TCP

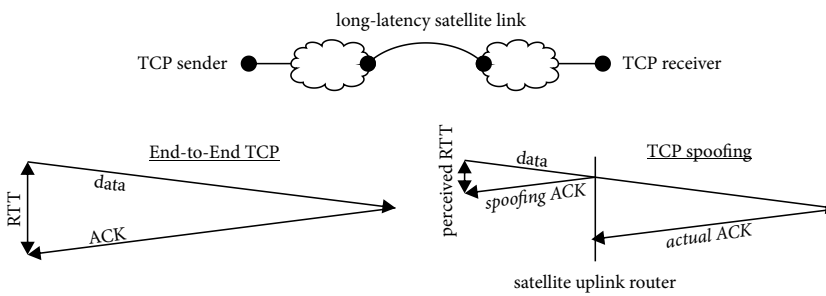

Рисунок 4. Применение TCP Spoofing через спутник

и связанные с ними ACK должны проходить по одним и тем же путям. Если это не так, то спутниковый шлюз, отвечающий за выполнение операций спуфинга (подмены), не сможет эффективно управлять TCP-соединениями на спутниковом канале.

SPLITTING

Обоснование подхода разделения (*splitting*) состоит в том, чтобы отделить спутниковую часть от остальной сети, чтобы иметь выделенные и отдельные соединения [9]. На практике этот подход может быть реализован путем рассмотрения функций разделения, действующих на спутниковые шлюзы. Таким образом, TCP-соединение, установленное между хостами, разделяется на три отдельных соединения, два из которых проходят по путям, связывающим терминальные хосты со спутниковыми шлюзами, а третий между шлюзами и спутником. С точки зрения сети определена новая архитектура, и ее дополнительная ценность заключается в возможности скрыть спутниковый канал для конечных пользователей, которые продолжают использовать стек TCP для подключения к спутниковым шлюзам, в то время как возможно воспользоваться преимуществами специально разработанного транспортного протокола. Далее приводится пример архитектуры стека спутниковых протоколов (SPS), а также подчеркивается роль агентов, участвующих в обмене данными.

Архитектура стека спутниковых протоколов (SPS). Общая архитектура представлена на рисунке 5. Сеть состоит из наземных компонентов и спутникового компонента. Последний изолируется от остальной сети с помощью объектов ретрансляции. Транспортный уровень этого нового SPS называется спутниковым транспортным уровнем (STL), и он реализует спутниковый транспортный протокол (STP), подходящий для конкретной среды. Предлагаемая архитектура может быть приемлемой альтернативой также в случае,

когда спутниковый компонент представляет сеть доступа, как показано на рисунке 6. Ретранслятор — это простой инструмент, непосредственно подключенный к ПК. Им также может быть плата с жёстким диском внутри ПК, сетевая или видеокарта. Модуль объекта ретрансляции можно просто встроить в программный модуль, чтобы загрузить его непосредственно в устройства. С точки зрения многоуровневого протокола, ключевая точка представлена двумя объектами ретрансляции, которые являются двумя шлюзами к спутниковой части сети. SPS воздействует на спутниковые каналы, используя необходимую информацию. Уровень ретрансляции гарантирует связь между спутниковым транспортным уровнем и протоколом, используемым в кабельной части (TCP).

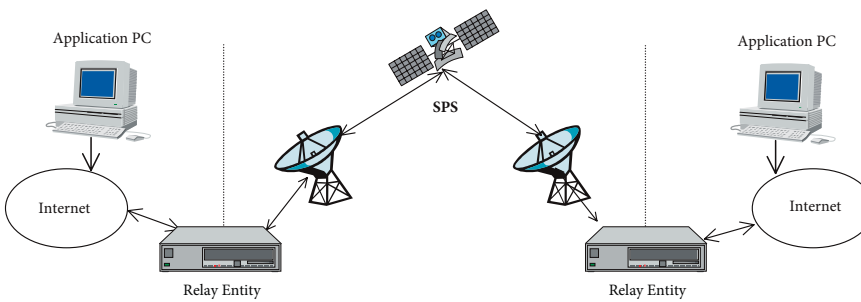

Рисунок 5. Архитектура SPS, общая организация

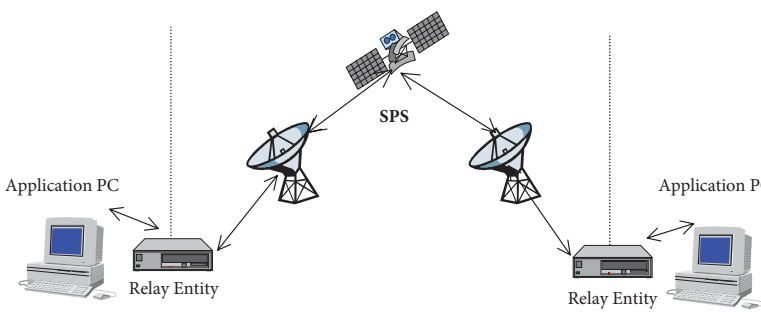

Рисунок 6. Архитектура SPS, процесс доступа к сети

С одной стороны, предложенная архитектура (рис. 7) использует возможность принятия в непроволочной части сети транспортного протокола, разработанного специально для спутниковой среды. С другой стороны, он нарушает сквозную семантику, возлагая на промежуточные шлюзы ответственность за проверку общей надёжности связи и поддержание состояния соединения. Отсутствие сквозной семантики означает, что даже если TCP-соединение установлено между одноранговыми узлами в начале связи, на самом деле существуют три отдельных соединения, а именно между первым ПК и первым ретранслятором, между первым и вторым объектами передачи, между вторым объектом передачи и вторым ПК.

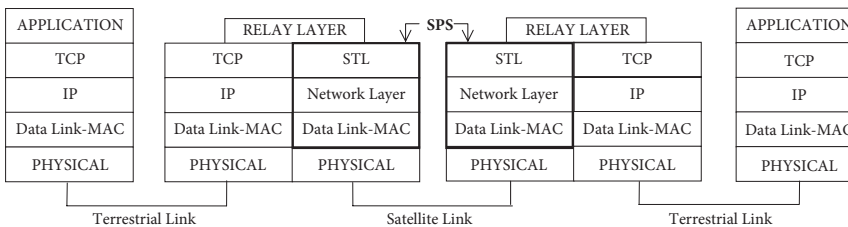

Рисунок 7. Архитектура прококола SPS

Более того, соединение на транспортном уровне разделено на три части, предназначенные для двух проводных линий связи и спутника, соответственно. Источник получает ACK от первого объекта ретрансляции, который открывает другое соединение с другими параметрами в зависимости от текущего состояния спутниковой части и распределяет доступные ресурсы. Ретранслятор на другой стороне спутникового канала работает аналогично в направлении пункта назначения, а транспортный уровень кабельных участков остается нетронутым.

Оптимизированные транспортные протоколы. Ниже показаны три протокола, которые могут использоваться для соединения двух

объектов связи, а именно: протокол *Xpress Transport Protocol* (XTP), протокол *Space Communications Protocol Specifications — Transport Protocol* (SCPS-TP) и протокол *Complete Knowledge Satellite Transport Protocol* (CK-STP). Транспортный протокол CK-STP будет описан как важный пример оптимизированных транспортных протоколов. Обратите внимание, что они также могут применяться отдельно, когда рассматривается только сквозное спутниковое соединение.

Транспортный протокол Xpress. XTP — это протокол, который был разработанн для условий большой задержки, высоких потерь и асимметричной полосы пропускания, типичных для спутниковой связи [10]. Протокол позволяет приложению легко выбирать тип запрашиваемой услуги. Например, выбирая соответствующую конфигурацию, XTP может предоставить приложению ту же услугу, которая обычно предоставляется TCP или UDP (но с большей эффективностью). Например, в режиме виртуального канала, подобно TCP, XTP применяет алгоритм выборочной повторной передачи для восстановления потерь. Обычно, когда получатель обнаруживает пропуски в последовательности принятых пакетов, он передает отправителю список пропущенных пакетов, позволяя отправителю быстро и эффективно выполнить всю необходимую повторную передачу. Кроме того, XTP предоставляет дополнительные услуги и функции, такие как надежный протокол многоадресной рассылки, управление скоростью и пакетными данными и, в целом, гибкость, позволяющую подобрать услугу в соответствии с конкретными потребностями приложения.

Также XTP регулирует поток данных с помощью механизма сквозного управления потоком окон, основанного на 64-битных порядковых номерах и 64- битном скользящем окне. XTP также обеспечивает управление скоростью, с помощью которого конечная или промежуточная система может указать максимальную полосу пропускания и размер пакета, который она может принять. Наконец, предлагаемая схема контроля ошибок основана как на по-

ложительных, так и на отрицательных ACK, чтобы воздействовать на повторную передачу отсутствующих или поврежденных пакетов данных. Повторные передачи могут быть выполнены либо с использованием политики «N-возврата», либо «выборочного повтора».

Стандарт протокола космической связи. Проект SCPS направлен на предоставление набора стандартных протоколов [11, 34], оптимизированных для космической связи. Стек протоколов SCPS показан в таблице 2. Каждый протокол совместим со стандартными интернет-протоколами. Транспортный протокол SCPS основан на TCP и UDP, а также на некоторых расширениях, например:

— расширения TCP для повышнной производительности;
— расширения для обработки запросов (транзакций);
— выборочное отрицательное подтверждение (SACK);
— сжатие заголовка.

Таблица 2. Стек протокола SCPS

SCPS file protocol (SCPS-FP)
SCPS transport protocol (SCPS-TP)
SCPS security protocol (SCPS-SP)
SCPS network protocol (SCPS-NP)

Чтобы справиться с потерями из-за повреждения данных, перебоев в каналах связи и перегрузки, SCPS-TP реализует различные стратегии восстановления после ошибок. Например, когда происходит изолированная потеря данных, скорость передачи не уменьшается, а значение RTO остается неизменным (явный ответ на повреждение). Управление перегрузкой может быть выполнено с использованием либо стандартных механизмов (немедленный за-

пуск, предотвращение перегрузки, экспоненциальный откат RTO), либо механизма управления перегрузкой TCP Vegas. При желании можно отключить контроль перегрузки SCPS-TP.

Complete Knowledge Satellite Transport Protocol. В протоколе CK-STP имеется возможность узнавать основные характеристики сетей, в которых осуществляется передача данных (доступная полоса пропускания или задержка распространения). Более того, если предположить, что в спутниковых сетях (в более общем смысле в беспроводных средах) события перегрузки могут быть относительно нечастыми (в противном случае могут быть приняты явные контрмеры, такие как явные методы уведомления о перегрузке), потеря пакетов происходит в основном из-за ошибок канала [7]. Чтобы гарантировать эффективное использование доступных ресурсов сети с точки зрения возможностей полосы пропускания, начальное окно устанавливается равным произведению полосы пропускания и задержки, и значение окна перегрузки больше не увеличивается при поступлении подтверждений. Более того, TCP-буферы на стороне получателя и отправителя должны быть правильно настроены, чтобы вышеупомянутые модификации были эффективными. Кроме того, механизм восстановления TCP был изменен путём включения режима быстрой повторной передачи после получения дублированного подтверждения.

Основное улучшение этого протокола состоит в том, что в конце фазы восстановления значение окна перегрузки не уменьшается, и, как следствие, скорость передачи поддерживается на уровне, установленном при открытии TCP- соединения, чтобы заполнить канал полосы пропускания. Также было немного изменено управление тайм-аутом. В частности, когда таймер истекает, окно перегрузки не уменьшается до одного сегмента, а после завершения фазы повторной передачи передача данных продолжается с максимально возможной скоростью, как в предыдущем случае.

Преимущества такого подхода очевидны по сравнению с поведением TCP. Фактически, в то время как TCP сокращает окно перегрузки после обнаружения потерянного пакета, CK-STP оставляет cwnd неизмененным, обеспечивая тем самым результаты с высокой производительностью.

ДРУГИЕ ТЕХНОЛОГИИ TCP PEP

Методы спуфинга/разделения часто считаются синонимами TCP PEP. Эти методы являются наиболее распространенными, но определены и другие механизмы [1, 37]. Фактически, некоторые аспекты, относящиеся к спутниковым линиям связи (т. е. эффекты больших пакетов сегментов или асимметрии сетевого тракта), требуют принятия дальнейших усовершенствований. Ниже перечислены некоторые из наиболее распространенных механизмов, применяемых в TCP PEP:

— Интервал ACK. В средах с большой задержкой полосы пропускания пакеты с ACK, как правило, достигают источника TCP сгруппированно, вызывая нежелательные пакеты сегментов. Передача больших пакетов может привести к потере многих сегментов в буфере шлюза, вызывая очень длинные периоды восстановления после ошибок. Чтобы избежать этой проблемы, метод разнесения ACK сглаживает поток TCP ACK.

— Фильтрация и реконструкция ACK. В случае каналов с очень высокой асимметрией полосы пропускания поток ACK в низкоскоростном направлении может испытывать потери из-за событий перегрузки, ограничивая скорость передачи в широкополосном прямом канале. Механизм фильтрации и восстановления ACK позволяет уменьшить количество ACK, проходящих по обратному каналу, за счёт фильтрации их на одной стороне соединения. Затем правильная последовательность ACK будет восстановлена на про-

тивоположной стороне канала соединения, прежде чем достигнет источника TCP.

— Туннелирование. Туннелирование — это метод PEP, который инкапсулирует сообщения, чтобы заставить их пересечь определенное соединение. Агент PEP, установленный в конце «туннеля», удалит оболочки инкапсуляции перед доставкой сообщения в итоговую конечную систему. Такой метод может быть очень полезен в случае разделенных соединений, чтобы гарантировать прохождение пакетов через распределенные агенты PEP.

— Сжатие. Метод сжатия направлен на уменьшение количества байтов, отправляемых по ссылке. В результате, в случае каналов с ограниченной полосой пропускания, такой метод приводит к следующим преимуществам: повышение пропускной способности канала, уменьшение задержки и снижение PER (*packet encoding rules*).

— Обработка периодов разрыва соединения с TCP. На беспроводные каналы могут влиять периоды простоя, которые резко сказываются на производительности TCP. Фактически, когда отправитель TCP не получает ожидаемых ACK, происходит тайм-аут повторной передачи, и, следовательно, окно перегрузки сокращается до одного сегмента. Чтобы смягчить эту проблему, агент TCP PEP может отслеживать трафик, идущий от отправителя к получателю, чтобы всегда сохранять последний ACK и в конечном итоге использовать его для установки объявленного TCP окна равным нулю, если обнаружен период отключения. То есть, отправитель TCP перейдет в режим удержания, что предотвратит истечение времени таймера RTO (*Retransmisson Time Out*).

— Мультиплексирование на основе приоритета. Обычно канал связи используется соединениями, поддерживающими приложения с различными характеристиками. Целью мультиплексирования на основе приоритетов является предоставление привилегий «срочным» передачам данных (то есть интерактивным приложе-

ниям) путем задержки одновременных низкоприоритетных массовых передач.

СЕТИ С УСТОЙЧИВОСТЬЮ К ЗАДЕРЖКАМ

Недостатки, связанные с применением транспортных протоколов на основе TCP в спутниковых сетях, более выражены и критичны в случае сред, в которых наблюдаются большие задержки и высокие значения BER. Транспортировка данных по проблемным сетям требует принятия специальных архитектур протоколов, направленных на снижение воздействия ухудшения канала передачи. Это относится к протоколам, разработанным для более высоких уровней, а именно для транспортного уровня, поэтому появление концепции сети с устойчивостью к задержкам DTN (*Delay Tolerance Network*) может открыть новые пути для спутниковой связи и расширить саму концепцию сети.

КОНЦЕПЦИЯ DTN

Архитектура DTN [38] решает проблему передачи данных в так называемых сложных средах. Действительно, опыт показал, что устаревший TCP неэффективен, когда сети включают в себя соединения или узлы, которые работают с перебоями, подвержены ошибкам или имеют большую задержку распространения сигнала. Чтобы справиться с этой проблемой, архитектура DTN производит ретрансляцию с сохранением/пересылкой «пакетов» (данных приложения). Концепция DTN была впервые задумана в рамках Исследовательской группы межпланетных сетей (IPNRG) и Исследовательской группы по технологиям Интернета (IRTF), чтобы решать проблемы именно космической связи. Позже область применения сетей с устойчивостью к задержкам была расширена,

чтобы охватить все «проблемные сети» в независимости от механизма передачи сигнала (кабель или радиоволна). С этой целью в 2002 году в IRTF была создана исследовательская группа *Delay Tolerant Networks Research Group* (DTNRG). Среди различных документов, опубликованных этой группой, стоит упомянуть спецификацию пакета протокола (версия 3.0), реализацию DTN API (версия 2.1.1), а также несколько проектов, использующих технологии DTN.

Основная идея DTN — это разделение большой и разнородной сети на однородные области и введение нового *связанного уровня* (между прикладным и транспортным уровнями). Длинные сквозные соединения на уровне передачи пакетов, которые должны проходить через множество однородных подсетей, затем разделяются на множество сегментов, каждый из которых передаётся в однородной подсети (в качестве общей сквозной подсети) как соединение на транспортном уровне. На первый взгляд эта идея напоминает архитектуру разделения, описанную в предыдущем разделе, и на самом деле её можно рассматривать как её мощное расширение. Однако есть много различий. Во-первых, большую разнородную сеть можно разделить более чем на два региона. Во-вторых, нет неявного и подверженного проблемам нарушения сквозной семантики транспортного протокола. В DTN реальное сквозное управление назначается новому слою пакета. На транспортный протокол теперь явно возложена задача сквозного контроля над той же подсетью. Это чёткое подразделение очень естественным и удобным способом приводит к использованию версий TCP, оптимизированных для определённых характеристик промежуточных подсетей, или к использованию совершенно другого стека. Таким образом, преимущества различных подходов, рассмотренные в предыдущих разделах (усовершенствования TCP, архитектуры разделения, оптимизированные транспортные протоколы спутниковой связи), могут быть сохранены и расширены, в то время как большинство не-

достатков фактически устранены. Более подробная информация об архитектуре DTN представлена далее.

АРХИТЕКТУРА DTN

Ядро DTN представлено пакетным протоколом, который работает поверх традиционного стека протоколов. Уровень конвергенции обеспечивает гладкий интерфейс между пакетным уровнем и нижележащим стеком, который может быть (или не быть) обычным стеком TCP/IP. Пакеты отправляются, перенаправляются и принимаются пакетными агентами. Каждому агенту назначается идентификатор, называемый кортежем, который используется для маршрутизации. Он состоит из двух частей: идентификатора области и идентификатора агента. Идентификатор агента уникален в пределах одной области и используется только тогда, когда кортеж достигает узла целевой области (позднее связывание). Транспортировка пакетов осуществляется в режиме хранения и пересылки. Каждый агент, участвующий в DTN, имеет возможность сохранить пакет, если возникнет необходимость.

Можно указать, что архитектура DTN (рис. 8) позволяет разделить всю сеть на отдельные области, которые принимают свой собственный стек протоколов (обозначенный на рисунке как уровни, специфичные для области). С этой точки зрения можно сказать, что подход DTN создает «сеть сетей» для обмена данными между хостами приложений на основе следующей парадигмы: с одной стороны, узлы DTN (граничные шлюзы) обеспечивают сквозную надёжность связи посредством передачи учёта и функций повторной передачи пакетов, реализованных на уровне пакетов, а с другой стороны, связь «внутренней» сети управляется региональными уровнями. Совершенно очевидно, что в целом в областях также может быть принят набор протоколов TCP/IP. В этом случае важно провести некоторые аналогии с архитектурой разделения, пред-

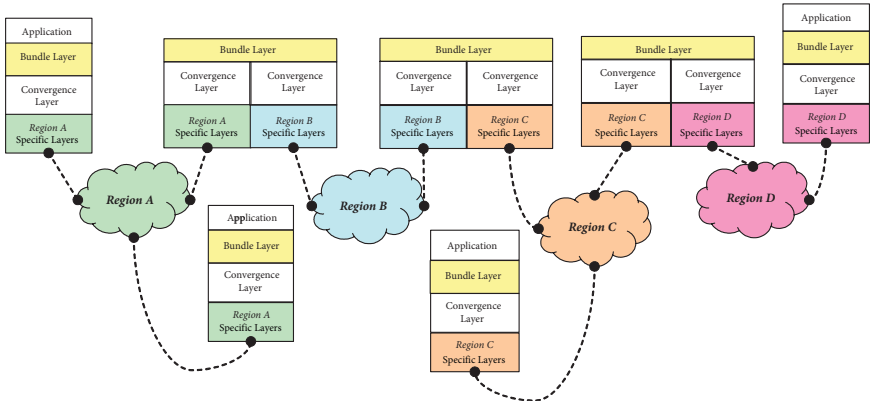

Рисунок 8. Архитектура протокола DTN

ставленной в предыдущем разделе. Фактически, мы можем указать, что обе архитектуры могут разделить сеть на отдельные части, чтобы принять в каждой из них стек протоколов «произвольно». Но также присутствуют и некоторые отличия. В случае архитектуры разделения хосты приложений не знают об операциях разделения/очереди, выполняемых в середине. Более того, промежуточные агенты отвечают за перехват TCP-соединений и, следовательно, нарушают сквозную семантику. С другой стороны, в архитектуре DTN хосты приложений реализуют уровень связки и уровень конвергенции. Следовательно, они «знают», что TCP- соединение будет установлено с первым шлюзом DTN, присутствующим на пути. Кроме того, каждая область, ограниченная шлюзами DTN, действует как отдельный административный домен, в котором вопросы именования, маршрутизации и безопасности решаются независимо от других областей.

Основным преимуществом описанной архитектуры является то, что она хорошо справляется с передачей данных в средах, в которых не всегда обеспечивается сквозное соединение. Однако, задержка связи не означает ненадёжную связь. Пакетный протокол обеспе-

чивает доставку и пересылку уведомлений. К другим транспортным функциям относятся срочная доставка данных, фрагментация пакетов и службы безопасности (аутентификация). Типичные сценарии применения, которые строго коррелируют со спутниковой средой передачи, представлены межпланетными сетями и связью в дальнем космосе. С этой точки зрения фундаментальное значение DTN состоит в том, что мы обнаруживаем потребность в новых протоколах, способных хранить данные и передавать их только в том случае, если канал считается надежным. Эта необходимость оправдана тем фактом, что межпланетные линии связи имеют очень высокий BER, и, следовательно, готовность передачи данных к месту назначения рекомендуется только при наличии физической среды передачи. Более того, хранение данных в локальных устройствах позволяет избежать большого количества повторных передач, которые в противном случае требовались бы всякий раз, когда качество канала связи было очень низкое. Одновременно решаются также вопросы, касающиеся энергопотребления. Фактически, возможность задержки передачи и значительное сокращение количества необходимых повторных передач приводит к значительной экономии энергии. С другой стороны, внедрение архитектуры DTN и, в частности, службы хранения, может создать проблему событий перегрузки в буфере уровня пакета. Эта проблема не является необычной и может возникнуть, когда канал передачи недоступен, и шлюз DTN должен хранить данные до тех пор, пока канал связи снова не станет доступным. Учитывая долгую продолжительность перебоев в работе каналов дальнего космоса, буфер DTN, вероятно, будет страдать от событий переполнения и, следовательно, сбросит несколько пакетов. Поэтому встаёт задача разработки схем управления перегрузками, подходящих для DTN. Однако стоит сказать, что длительные задержки делают невозможным применение схем с обратной связью. Более того, одного контроля за заполнением буфера недостаточно, чтобы справиться с этой проблемой. С этой

целью дополнительный мониторинг изменения времени заполнения буферной очереди и нагрузки от входящего трафика может дать значительные преимущества.

Другой важный вопрос, который необходимо решить, — это реакция на события потери пакетов. В этом случае, если услуга DTN сконфигурирована должным образом, сам пакетный уровень будет отвечать за повторную передачу пропущенной информации. Возможная альтернатива представлена механизмами восстановления, реализованными на нижележащих уровнях. Возможный кандидат для выполнения этой задачи — протокол передачи Ликлайдера (LTP), который по сути является протоколом «точка-точка» и наследует свои основные процедуры из спецификации протокола доставки файлов Международного Консультативного Комитета по космическим системам передачи данных (CCSDS) [39]. Рассмотрим этот протокол подробнее.

ПРОТОКОЛ ПЕРЕДАЧИ ЛИКЛАЙДЕРА

Licklider transmission protocol (LTP) разработан для обеспечения надёжности на основе повторной передачи данных в проблемных Интернет-средах, демонстрирующих чрезвычайно длительный RTT, частые перебои в подключении и высокие значения BER [40]. Связь через межпланетное пространство является наиболее ярким примером такого рода среды, а протокол LTP может обеспечивать надёжную передачу данных в этой среде, хотя он может быть применён и в других средах.

В межпланетном Интернете LTP может быть использован в качестве надёжного уровня конвергенции по однонаправленным радиочастотным каналам в дальнем космосе, фактически работая как протокол уровня канала данных непосредственно через физический интерфейс. LTP реализует ARQ (automatic repeat request)

передачи данных, запрашивая у адресата отчёты о получении на основе технологии SACK.

С точки зрения архитектуры, протокол LTP был разработан для работы в рамках связанного протокола, чтобы гарантировать надежную передачу данных, тогда как более высокие уровни могут не поддерживать функции восстановления или повторной передачи. В частности, применение такого решения может быть предусмотрено во всех случаях, когда пакетный протокол настроен только на выполнение операций сохранения и пересылки, пренебрегая возможными операциями повторной передачи пакета и, следовательно, работая в неподтвержденном режиме. Согласно этой точке зрения, протокол LTP будет нести ответственность за обеспечение устойчивой связи, используя процедуру повторной передачи и процедуру ускоренной повторной передачи.

ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОТОКОЛОВ ТРАНСПОРТНОГО УРОВНЯ ДЛЯ СПУТНИКОВОЙ СВЯЗИ

Коммуникационные технологии, используемые для подключения к Интернету, радикально изменились с 1980-х годов, когда была представлена архитетура стека TCP/IP. Большая часть трафика вскоре будет генерироваться мобильными устройствами, подключёнными к различным поколениям беспроводных локальных сетей и сотовых сетей, которые теперь могут обеспечивать конечным пользователям скорость передачи данных в несколько гигабит в секунду. Мобильные устройства становятся всё более и более разнообразными и поддерживают возможность подключения через различные интерфейсы и сети. Более того, пропускная способность фиксированных и транзитных сетей резко увеличилась благодаря достижениям в области оптической связи.

Достижения в области сетевых технологий поспособствовали разработке новых информационных технологий, таких как виртуальная и дополненная реальность, облачные игры, послужив основным драйвером для технологической революции 4.0. Описанные технологии, среди прочих, предъявляют строгие требования к сквозной производительности сетей, например, с точки зрения пропускной способности канала, задержки сигнала и надежности передачи. Наряду с возросшей неоднородностью сети, это потребовало эволюции парадигм транспортного уровня, которым уже несколько десятилетий, что снова привлекло внимание к транс-

портному уровню. Действительно, сквозная производительность и качество обслуживания, воспринимаемое пользователями, зависит не только от возможностей технологий связи, но также от взаимодействия между сетью, приложениями и транспортным уровнем. Поэтому в последнее время возрос интерес к разработке и оценке производительности новых транспортных протоколов, механизмов управления перегрузками и решений для максимизации сквозной производительности беспроводных и проводных сетей следующего поколения.

Этот раздел описывает последние достижения в разработке протоколов транспортного уровня, удовлетворяющие требованиям к производительности приложений в современных и будущих сетях в следующем десятилетии. Итак, современная наука ожидает изменений по трём основным направлениям:

1. разработка новых транспортных протоколов, которые смогут представлять собой альтернативу общему протоколу управления передачей (TCP);

2. развитие методов управления перегрузкой с учетом прокси-серверов, повышающих производительность сети, и межуровневых подходов или подходов с использованием искусственного интеллекта и машинного обучения;

3. введение возможностей многолучевого распространения радиосигнала на транспортном уровне.

Актуальные процессы разработки и исследования транспортных протоколов спутниковой связи в настоящий момент в основном сосредоточены на следующих пунктах:

— требования к транспортным протоколам в будущих сетевых сценариях;

— новые тенденции и результаты в управлении перегрузкой (подходы на основе AI/ML, стратегии с низкой задержкой и высокой надёжностью, межуровневое управление);

— разработка новых протоколов (QUIC и XSTP, см. ниже) и улучшение существующих;

— многолучевой подход для транспортного протокола в радиосвязи: проектирование, выбор пути, оценка производительности;

— контроль перегрузки для многолучевых транспортных протоколов;

— проектирование, оптимизация и производительность транспортных протоколов для сетей 5G и 6G;

— сетевые решения для повышения пропускной способности (прокси);

— усилия по стандартизации и развитию протоколов.

STP ПРОТОКОЛ

Спутниковый транспортный протокол — это протокол спутникового транспорта (STP), предложенный Кацем и Хендерсоном [41, 42]. Данный протокол специально оптимизирован для уникальных ограничений спутниковой сетевой среды. Обнаружено, что STP превосходит TCP в средах, характеризующихся высоким BER, серьёзной асимметрией и различными RTT, типичными характеристикам спутниковых каналов на низкой земной орбите.

Основные особенности STP [6] следующие:

— обеспечение разделения между данными и контролем информации для минимизации накладных расходов на управление в меньших сегментах данных;

— механизм адаптации к объему управления скоростью, требуемому в сети, от отсутствия управления скоростью до явного управления скоростью. В отличие от TCP, который использует свойство самосинхронизации, STP зависит от таймера отложенной отправки для равномерного ускорения передачи в течение расчетного RTT. Основным преимуществом такого механизма является снижение риска внесения в сеть больших пакетов;

— механизм управления перегрузкой для быстрого старта повторной передачи;

— эффективная стратегия подтверждения.

STP использует механизм автоматического запроса на повтор передачи (ARQ) и выборочное отрицание уведомления (NACK). При использовании этого механизма повторно передаются только те сегменты, об отсутствии которых сообщили получатели. Преимущество заключается в более низком трафике обратной линии связи, когда потери незначительны, а также в быстром восстановлении, когда потери являются серьезными. В отличие от TCP, в STP отсутствует механизм RTO, что делает его более устойчивым к изменениям RTT.

Наконец, важно отметить, что даже если STP включает в себя многие из основных принципов TCP, он эквивалентен ему только функционально, но не семантически. К сожалению, протокол STP наследует предвзятость управления перегрузкой от своих протоколов-предков (то есть TCP, SSCOP) [43]. Однако протокол может эффективно восстанавливаться после нескольких потерь за один и тот же цикл приёма-передачи, хотя сама тактика восстановления после ошибок может отрицательно сказаться на его общей производительности.

XSTP ПРОТОКОЛ

Расширенный спутниковый транспортный протокол XSTP (*eXtended Satellite Transport Protocol*) — это программная реализация протокола STP для Linux-подобных операционных систем [44]. Протокол используется для реализации новой стратегии контроля ошибок, называемой XSTP-зондированием. Обычно протокол XSTP может быть развёрнут поверх сетевого протокола (например, IP). Протокол обеспечивает надёжную службу потоковой передачи

байтов с установлением соединения для протоколов приложений (например, FTP).

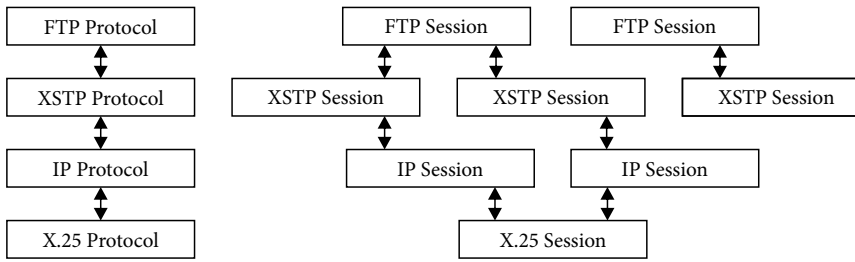

Рисунок 8. Стек протоколов XSTP/IP и конфигурация сессии с XSTP [8]

Сеанс XSTP состоит из нижней и верхней сессии. На рисунке 8 показана типичная конфигурация стека протоколов, включающего XSTP. Как объясняет Магед Э. Элаасар в своей статье [8], когда такой стек инициализируется, создаётся, настраивается и затем встраивается в иерархию протоколов, протоколы уровня приложений могут использовать службу протокола для управления сеансами XSTP. Сессия XSTP играет двойную роль (отправитель и получатель), что подразумевает определение двух новых классов: отправителя XSTP и получателя XSTP. Экземпляр каждого из этих классов создаётся в состоянии private в объекте сессии.

Протокол XSTP является перспективной технологией для спутниковых систем, обеспечивая более высокую эффективную пропускную способность, гораздо меньшие накладные расходы и лучшую эффективность канала по сравнению с клонами TCP. Но несмотря на все эти характеристики, протокол XSTP для спутниковых сетей не идеален. Например, служебные данные передачи по обратному каналу значительны в условиях высокого BER, и их необходимо уменьшать. Кроме того, на уровне механизма зондирования необходимо улучшить принцип принятия решения, чтобы различать перегрузку и другие типы ошибок, которые могут быть обнаружены

в спутниковых сетях. Другой важный аспект — энергия, затрачиваемая во время цикла зондирования.

Будущие исследования могут быть направлены на производительность XSTP по сравнению с типом топологии спутников (цветочное созвездие, кластеры, гибридное созвездие). Другое предложение — это сравнительное исследование между механизмами проверки XSTP и TCP, учитывая, что оба протокола могут быть настроены с одинаковым набором параметров. Это сравнение может показать наиболее эффективный механизм с точки зрения адаптации к различным ошибкам спутниковых линий. Наконец, механизм зондирования может быть изучен в контексте беспроводной связи или в аналогичной области, характеризующейся различными типами ошибок связи.

QUIC ПРОТОКОЛ

QUIC — это новый перспективный зашифрованный по умолчанию мультиплексированный транспортный протокол для спутникового Интернета, который ускоряет трафик HTTP [3]. QUIC в конечном итоге призван заменить TCP. Первоначально протокол был предложен «Google», а IETF в настоящее время активно ведёт работу по определению и стандартизации ряда RFC (рабочих предложений).

Протокол QUIC был разработан таким образом, чтобы его можно было легко развернуть и защитить, а также сократить задержки при рукопожатии и задержки блокировки очереди. Для обеспечения безопасности и возможности развертывания в существующих сетях служит одно из ключевых решений QUIC — расположение над протоколом UDP (рис. 9) [3].

Поскольку транспорт QUIC полностью шифрует заголовки, связанные с транспортными функциями, и аутентифицирует конечные точки соединения, то широко распространённые прокси-серверы

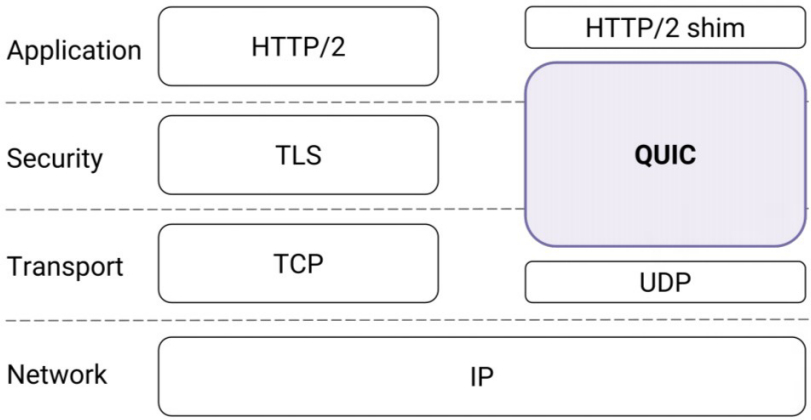

Рисунок 9. Протокол QUIC в традиционном стеке HTTPS

для повышения производительности (PEP) больше не могут использоваться для повышения производительности TCP.

QUIC объединяет криптографическое и транспортное подтверждение в один цикл приёма-передачи при настройке безопасного транспортного соединения. Если соединение установлено успешно, клиент может кэшировать информацию об источнике, к которому он подключился. При повторном подключении к тому же источнику, эта кешированная информация может использоваться для установления зашифрованного соединения без дополнительных циклов передачи. Данные могут быть отправлены сразу после отправки приветствия клиенту, даже не дожидаясь ответа сервера (RTT_0). Если кэшированная информация клиента устаревает, то это рукопожатие RTT_0 возвращается к варианту RTT-1 (рис. 10).

Реальное взаимодействие QUICv1 с использованием спутниковой службы уже активно изучается. Первая версия стандарта была опубликована 3 июня 2021 года [42]. В настоящее время принимаются поправки для формирования стандарта QUICv2 [43].

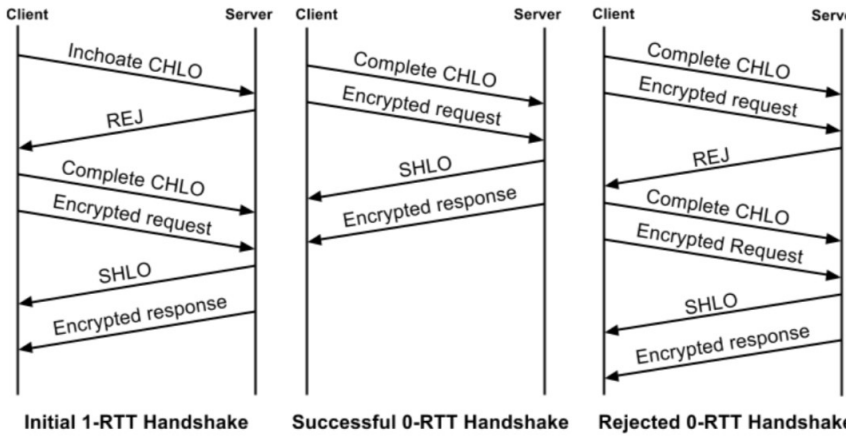

Рисунок 10. Рукопожатие в протоколе QUIC (3 случая)

Влияние на процесс стандартизации QUIC требует хорошего знания структуры IETF и процессов стандартизации. Кроме того, регулярные встречи разработчиков по стандартам IETF увеличивают влияние на отрасль, что дает возможность получать отзывы об исследованиях. Это особенно важно, учитывая, что спутниковая индустрия представляет собой лишь незначительную заинтересованную сторону в огромном интернет-сообществе, среди компаний «Google», «Amazon» и «Microsoft», также участвующих в процессе стандартизации QUIC [3].

В частности, компания «Google» отмечает, что дальнейшая разработка протокола QUIC опирается на постоянные масштабные эксперименты по изучению функций и настройке параметров, а также на эксперименты по аналитике сервисов приложений. Это позволяет компании количественно оценить сквозное влияние многих факторов на производительность, ориентированную на пользователей и приложения. В табл. 3 показано уменьшение задержки при браузерном поиске и просмотре видео, достигаемое при использовании

протокола QUIC. Улучшение в основном связано с уменьшением задержки при рукопожатии *(handshake)*.

Спутниковая связь в настоящее время уже активно использует интернет-протоколы, уделяется большое внимание безопасности соединения и шифрованию. Все это также влияет на дальнейшую раработку протокола QUIC. Так, по состоянию на май 2022 года продолжается ещё одно исследование (проект «MTAILS» Европейского космического агентства, длительностью 19 месяцев), которое ставит своей целью оценить производительность QUIC и оценить улучшения в реалистичной среде. Расширение проекта использует уникальную базу спутников-эмуляторов, разработанных в проекте «MTAILS».

Таблица 3. Уменьшение задержки при браузерном поиске и просмотре видео, %

		Lower latency					Higher Lower	
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop	8.0	0.4	1.3	1.4	26	5.8	10.3	16.7
Mobile	3.6	–0.6	–0.3	0.3	0.5	4.5	8.8	14.3
Video								
Desktop	8.0	1.2	3.1	3.3	4.6	8.4	9.0	10.6
Mobile	5.3	0.0	0.6	0.5	1.2	4.4	5.8	7.5

БЛИЖАЙШЕЕ БУДУЩЕЕ И ПРОТОКОЛЫ СПУТНИКОВОЙ СВЯЗИ

Трудно предсказать будущее, но не так уж сложно предсказать тенденции будущего развития, если у нас достаточно прошлых и текущих знаний. Помимо интеграции спутников в глобальную интернет-инфраструктуру, одной из основных задач является создание новых услуг и приложений для удовлетворения потребностей людей. На рисунке 11 показано примерное развитие спутниковых сетей ближайшего будущего.

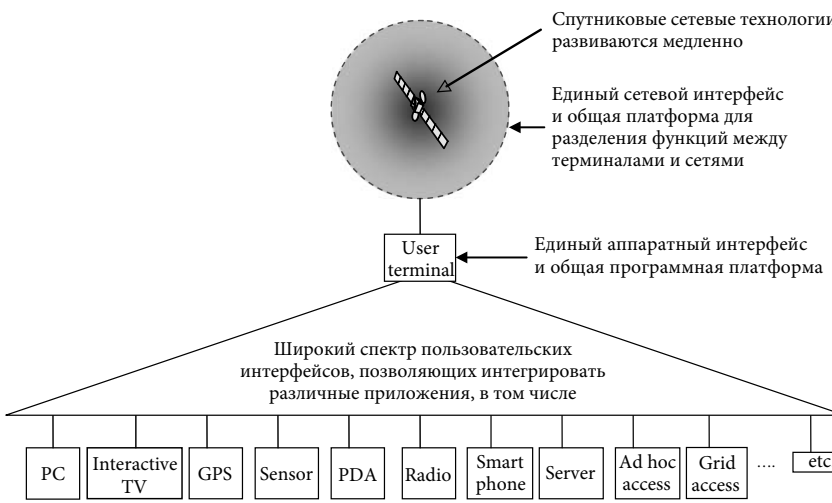

Рисунок 11. Развитие спутниковых сетей ближайшего будущего

За последние 25 лет возможности спутниковых сетей значительно увеличились в связи с развитием технологий. Вес спутников в зависимости от выполняемой задачи как увеличился (до 3000 кг), так и уменьшился (до 10 кг). Мощность спутников также, в зависимости от функции, показала рост как в большую (до 1000 Вт), так и в меньшую (до 10 Вт) сторону. Причём с развитием технологий в ближайшее время данные характеристики спутников будут и дальше расширяться в обе стороны. Что касается наземных спутниковых терминалов, то они уменьшились с 20–30 м до 0,5–1,5 м. Также идёт развитие портативных терминалов. Такие тенденции будут продолжаться, но, возможно, будут реализовываться по-разному, например, в зависимости от количества спутников и их так называемого созвездия, в зависимости от орбиты движения и, наконец, в зависимости от их размера и используемых частот связи. Пользовательские терминалы также могут функционировать как устройства взаимодействия с частными сетями или концентраторами сенсорных сетей.

С точки зрения спутниковых сетей мы увидим развитие: спутников-серверов, предоставляющих услуги непосредственно с орбиты, мультимедийных терминалов на борту спутников для наблюдения за нашей планетой, маршрутизаторов на борту космических аппаратов (в качестве сетевых узлов) для расширения доступа в Интернет. Спутники — это наши загадочные минизвёзды, которые мы создаём сами. Возможности спутниковых технологий в самое ближайшее время превзойдут все наши ожидания.

НАНОСПУТНИКИ КАК ТЕХНОЛОГИЯ БЛИЖАЙШЕГО БУДУЩЕГО

Научные движения в области сверхмалых орбитальных аппаратов уже сейчас являются самыми перспективными в области спутниковых технологий. Специалисты уверены, что их следующее поколение — наноспутники — изменит экономику космической

отрасли кардинально. Малые спутники доступны даже тем странам, которые только начинают осваивать космическое пространство. Например, Индонезия при помощи Германии запустила в 2007 году аппарат LAPAN-TUBSAT весом 56 килограмм. В течение года он вёл наблюдения за Землей.

До сих пор конструкторы наибольшее внимание уделяли спутникам класса мини и микро, наноспутники существовали в единичных экземплярах. Их делали в основном в образовательных целях, а коммерческие перспективы были весьма туманны. Сейчас же в понятие наноспутник входит орбитальный аппарат весом до 10 килограмм. Один из первых подобных аппаратов — SNAP-1, производства фирмы SSTL, запустили в 2000 году с космодрома Плесецк. Он нёс на борту систему картографирования Земли и системы наблюдения за другими орбитальными аппаратами. В конструкции использовали матричную ПЗС камеру с разрешением 1 километр и матрицей 512 х 512 элементов, 3-осевую ориентацию, GPS позиционирование. SNAP-1 весил 6,5 кг и обошёлся разработчикам в 1,5 миллиона долларов. Этот эксперимент показал, что наноспутники могут выполнять работы по дистанционному зондированию Земли не хуже более тяжеловесных и громоздких аналогов. Более того, возможности наноспутников постоянно совершенствуются, и уже в 2007 году американцы испытали сразу несколько таких наноспутников. Вскоре аппарат весом 3 килограмма сконструировали в Израиле, а в Берлинском университете запустили первый пикоспутник BEESAT весом всего 1 килограмм. Назначение этого крохи сводилось к испытанию миниатюрных элементов конструкции и приборов.

В России, как и во всем мире, ведутся активные разработки наноспутников. Первый отечественный аппарат этого класса запустили на орбиту в 2005 году. Селижан Шарипов, работая на МКС, просто вытолкнул его рукой в открытый космос. Наноспутник ве-

сом 5 килограмм сконструировали в НИИ космического приборостроения.

Возможности нынешних наноспутников многократно возросли по сравнению с первыми образцами. В 2015 году Россия запустила на орбиту второе поколение наноспутников, которые работают от солнечных батарей, что значительно продлевает срок их активного существования — ведь именно эта характеристика делает аппараты коммерчески привлекательными. Каждый наноспутник решает свою узкую задачу: отработка технологии управления аппаратом через телекоммуникационные сети, испытание полезной нагрузки для космического сегмента международной системы AIS (совместно с Европейским космическим агентством), исследование ионосферы, эксперименты с микродвигательными установками, которые придадут наноспутникам маневренность.

Наноспутники — это прорывная технология, которая меняет концепцию космического аппарата и вообще экономику космической отрасли. Впервые в космосе появился недорогой и доступный многим прибор. К тому же закон Мура, применяя его к этому классу аппаратов, предполагает дальнейшее уменьшение их стоимости и сроков внедрения.

ЗАКЛЮЧЕНИЕ

В заключение стоит сказать, что изучение и анализ механизмов надёжной передачи данных для спутниковой связи — это сложные задачи, поскольку они требуют разработки и реализации новых протоколов и архитектурных решений, хорошо подходящих для проблемной среды ближнего и дальнего космоса. Учитывая мировые тенденции по активному освоению космоса, любые исследования в области космической связи представляются весьма актуальными, особенно учитывая тот факт, что космическая среда является крайне агрессивной для радиосвязи, и, следовательно, инженерные разработки устройств связи с новой архитектурой — это наукоёмкая и сложная инженерная задача.

В данной же работе особое внимание было сосредоточено на решениях протоколов транспортного уровня. На основе последних научных статей и справочных данных (спецификации, стандарты) был сделан обзор важных для спутниковой связи спецификаций протоколов, унаследованных от TCP (TCP Peach, TCP Westwood, TCP Hybla и SCTP), а также были рассмотрены и новые интересные и перспективные протоколы (QUIC, STP, XSTP). Все они обращаются с разными целями и разными решениями к основным проблемам, связанным со спутниковой связью. Также были исследованы различные архитектурные решения, рассмотрены два важных примера PEP, такие как TCP-spoofing и TCP- разделение. Были выделены их плюсы и минусы с учетом повышения производительности, сложности и соответствия стандартной семантике. Наконец, была рассмотрена архитектура DTN, которая может гарантировать высокую степень надёжности доставки данных в сетевых сценариях с непредсказуемостью и недоступностью ресурсов.

Благодаря произведённому анализу различных подходов к решению проблемы передачи данных в системах спутниковой связи на транспортном уровне (и не только) в среднесрочной перспективе возможно разработать интеграционную (синергическую) концепцию совершенно новой архитектуры протокола. Также данная работа может иметь практическую ценность для разработчиков и инженеров, занимающихся точечным усовершенствованием уже действующих протоколов спутниковой связи.

Человечество все активнее осваивает околоземное космическое пространство, поэтому совершенствование действующих и разработка совершенно новых протоколов связи — это задача для новых практических исследований и аналитических обзоров.

СПИСОК ЛИТЕРАТУРЫ

1. Akyildiz I.F., Morabito G., Palazzo S. TCP-Peach: a new congestion control scheme for satellite IP networks. IEEE/ACM Trans Networking. 2001;9(3):307-321. doi:10.1109/90.929853.
2. Allman M., Dawkins S., Glover D., et al. Ongoing TCP Research Related to Satellites. RFC Editor; 2000:RFC2760. doi:10.17487/rfc2760.
3. Allman M., Floyd S., Increasing TCPs initial window. IETF RFC 2414, September 1998.
4. Allman M., Glover D., Sanchez L. Enhancing TCP Over Satellite Channels Using Standard Mechanisms. RFC Editor; 1999:RFC2488. doi:10.17487/rfc2488.
5. Allman M., Stevens W. TCP congestion control. IETF RFC 2581, April 1999.
6. Balakrishnan H., Padmanabhan V., Fairhurst G., Sooriyabandara M. TCP Performance Implications of Network Path Asymmetry. RFC Editor; 2002:RFC3449. doi:10.17487/rfc3449.
7. Barakat C., Altman E. Bandwidth tradeoff between TCP and link-level FEC. Computer Networks. 2002;39(2):133-150. doi:10.1016/S1389-1286(01)00305-X.
8. Barakat C., Chaher N., Dabbous W., Altman E. Improving TCP/IP over geostationary satellite links. In: Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99. (Cat. No.99CH37042). Vol 1b. Rio de Janeireo, Brazil: IEEE; 1999:781-785. doi:10.1109/GLOCOM.1999.830173.
9. Barbeau M. Protocol implementation framework for Linux, Technical report, 2002.
10. Border J., Kojo M., Griner J., Montenegro G., Shelby Z. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC Editor; 2001:RFC3135. doi:10.17487/rfc3135.

11. Braden R. T/TCP — TCP Extensions for Transactions Functional Specification. RFC Editor; 1994:RFC1644. doi:10.17487/rfc1644.
12. Brakmo L.S., Peterson L.L. TCP Vegas: end to end congestion avoidance on a global Internet. IEEE J Select Areas Commun. 1995;13(8):1465-1480. doi:10.1109/49.464716.
13. Burleigh S., Ramadas M, Farrell S. Licklider Transmission Protocol-Motivation. RFC Editor; 2008:RFC5325. doi:10.17487/rfc5325.
14. Caini C., Firrincieli R. Packet spreading techniques to avoid bursty traffic in long RTT TCP connections [satellite link applications]. In: 2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No.04CH37514). Vol 5. Milan, Italy: IEEE; 2004:2906-2910. doi:10.1109/VETECS.2004.1391456.
15. Caini C., Firrincieli R. TCP Hybla: a TCP enhancement for heterogeneous networks. Int J Satell Commun Network. 2004;22(5):547-566. doi:10.1002/sat.799.
16. Casetti C., Gerla M., Mascolo S., Sanadidi M.Y., Wang R. TCP Westwood: end-to-end congestion control for wired/wireless networks. Wireless Networks. 2002;8(5):467-479. doi:10.1023/A:1016590112381.
17. Celandroni N., Potortì F. Maximizing single connection TCP goodput by trading bandwidth for BER: MAXIMIZING SINGLE CONNECTION TCP GOODPUT. Int J Commun Syst. 2003;16(1):63-79. doi:10.1002/dac.580.
18. DTN Research Group web page: http://www.dtnrg.org.
19. ETSI. Digital video broadcasting (DVB); interaction channel for satellite distribution systems. ETSI EN 301 790 V1.4.1. ETSI, September 2005.
20. ETSI. Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications. ETSI EN 302 307 V 1.1.1. ETSI, June 2004.

21. Floyd S. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. SIGCOMM Comput Commun Rev. 1991;21(5):30-47. doi:10.1145/122431.122434.
22. Gerla M., Ng B.K.F., Sanadidi M.Y., Valla M., Wang R. TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs. Computer Communications. 2004;27(1):41-58. doi:10.1016/S0140-3664(03)00114-2.
23. Henderson T., Katz R. Satellite transport protocol (STP): An SSCOP-based transport protocol for datagram satellite networks, Proceedings of 2nd Workshop on Satellite-Based Information Systems, 1997.
24. Henderson T.R., Katz R.H. Transport protocols for Internet-compatible satellite networks. IEEE J Select Areas Commun. 1999;17(2):326-344. doi:10.1109/49.748815.
25. Ioannidis J., The coherent file distribution protocol. IETF RFC 1235, June 1991.
26. Katz R. Satellite Transport Protocol, PhD thesis, Dec. 1999, 17th Annual AIAA/ USU Conference on Small Satellites.
27. Larsen K.J. Short convolutional codes with maximal free distance for rates 1/2, 1/3 and 3.
28. Maged E.E., Zheyin I.I., Barbeau M., Kranakis E. The eXtended Satellite Transport Protocol: Its Design and Evaluation. In: 17th Annual AIAA/USU Conference on Small Satellites. Ottawa, Canada; 2003.
29. Marchese M. TCP/IP-based protocols over satellite systems: a telecommunication issue. In: Stavroulakis P, ed. Reliability, Survivability and Quality of Large Scale Telecommunication Systems. Chichester, UK: John Wiley & Sons, Ltd; 2002:167-198. doi:10.1002/0470860812.
30. Mathis M., TCP selective acknowledgment options. IETF RFC 2018, October 1996.
31. Mogul J., Deering S. Path MTU discovery. IETF RFC 1191, November 1990.
32. Paxon V., Allman M. Computing TCP's retransmission timer. IETF RFC 2988, November 2000.

33. PIE. XTP Protocol Definition Revision 3.6. PEI 92–10, Protocol Engines Incorporated, Mountain View, CA, 11 January 1992.
34. Ren Wang, Pau G., Yamada K., Sanadidi M.Y., Gerla M. TCP start-up performance in large bandwidth networks. In: IEEE INFOCOM 2004. Vol 2. Hong Kong, China: IEEE; 2004:796-805. doi:10.1109/INFCOM.2004.1356968.
35. Ren Wang, Valla M., Sanadidi M.Y., Gerla M. Adaptive bandwidth share estimation in TCP Westwood. In: Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE. Vol 3. Taipei, Taiwan: IEEE; 2002:2604-2608. doi:10.1109/GLOCOM.2002.1189101.
36. Rizzo L. Effective erasure codes for reliable computer communication protocols. SIGCOMM Comput Commun Rev. 1997;27(2):24-36. doi:10.1145/263876.263881.
37. Shaojian Fu, Atiquzzaman M. SCTP: state of the art in research, products, and technical challenges. IEEE Commun Mag. 2004;42(4):64-76. doi:10.1109/MCOM.2004.1284931.
38. Sklar B. Digital Communications (2nd edn). Prentice-Hall: Englewood cliffs, NJ, 2001.
39. Space Data and Information Transfer Systems. Space Communications Protocol Specification (SCPS). Transport Protocol (SCPS-TP): BSI British Standards doi:10.3403/30169382U.
40. Stewart R., Xie Q., Morneault K., et al. Stream Control Transmission Protocol. RFC Editor; 2000:RFC2960. doi:10.17487/rfc2960.
41. Zhang Y. Interworking and Computing over Satellite Networks. Kluwer Academic Publisher: Dordrecht, 131–154, 2003.
42. RFC 9000 - QUIC: A UDP-Based Multiplexed and Secure Transport. IETF. doi:10.17487/RFC9000.
43. QUIC Version 2. Standards Track - 11 July 2022. Google LLC. Available: https://quicwg.org/quic-v.

Научное издание

Владимиров Игорь Викторович
Гаврилов Олег Алексеевич
Соколова Светлана Владимировна

ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ СПУТНИКОВОЙ СВЯЗИ: АНАЛИЗ, СРАВНЕНИЕ, ПРОГНОЗ РАЗВИТИЯ

Монография

Корректор: А. В. Сорокин
Компьютерная верстка: Р. И. Газизов
Дизайн обложки: Д. А. Щеглов

Подписано к использованию 09.09.2022 г.
Объем данных — 4,1 Мб.
Сист. требования: Adobe Reader.

Издательский дом «Сциентиа»
г. Санкт-Петербург, пер. Дегтярный, д. 22, литер А
Тел. +7 (812) 649-93-76
www.scientia-pub.org
info@scientia-pub.org