Архитектура промышленных приложений IoT и протоколы AMQP, MQTT, JMS, REST, CoAP, XMPP, DDS

С.П. Селезнёв, В.В. Яковлев

Аннотация—Наиболее важные протоколы обмена сообщениями, предлагаемые в качестве основы нового поколения Интернета вещей (IOT) и, более конкретно, промышленных интернет - приложений (IIoT) рассмотрены в данной статье. Понимание архитектуры и требований разделения сообщений в каждой целевой системе влияет на выбор наиболее подходящего решения для операционных технологий (OT) или АСУ ТП.

Если целевая система должна обрабатывать от тысячи и более других сетевых устройств выбор протоколов AMQP, MQTT, JMS, REST или XMPP может привести к снижению производительности и особой сложности. CoAP поддерживает IP Multicast, позволяя одним запросом опубликовать сообщение на нескольких устройствах одновременно.

В отличие от DDS, который обеспечивает поддержку динамического обнаружения (Discovery), настройка системы, которая использует AMQP, MQTT или JMS через брокера обязательна. Доступ к брокеру реализуется, как правило, с помощью известного сетевого адреса или поиска услуг, таких как JNDI.

Протокол DDS может обеспечить режим реального времени, многие-ко-многим, управляемые соединения, требуемые высокопроизводительным приложениям (устройство-устройство). DDS также становится ключевым протоколом обмена сообщениями для подключения в режиме реального времени сетей устройств к облачным ЦОД (Cloud – Fog – Cloudlet). Конкретная реализация DDS (на сегодня в мире около десятка независимых производителей) может предложить уникальный способ обмена данными высокой производительности.

Обеспечение того, что система отказоустойчивая и безопасная является ключевым фактором в мире IoT, состоящей потенциально из многих тысяч устройств обмена сообщениями. Ведущие реализации, как правило, обеспечивают собственные решения, основанные на SSL или TLS, а также s/Mime. AMQP и XMPP специфицируют использование SASL, чтобы обеспечить подключаемый интерфейс аутентификации сообщений и в принятой спецификации OMG DDS стандартизована комплексная система безопасности.

Статья получена 9 апреля 2019.

Селезнёв С.П., ктн, системный архитектор ООО «Фактор-ТС», ssp@factor-ts.ru.

Яковлев В.В., ктн, председатель совета директоров ООО “Фактор-ТС”, yvv@factor-ts.ru

Ключевые слова—Интернет вещей, IoT, IIoT, протоколы, сообщения.

I. Введение

Концептуально Интернет вещей относится к общей идее вещей, особенно предметов быта, которые находятся, читаются, узнаются, передвигаются, адресуются и управляются через Интернет, будь то с помощью RFID меток, беспроводной локальной сети, WAN или другими средствами. Повседневные объекты включают в себя не только электронные устройства, с которыми мы сталкиваемся каждый день, не только продукты высшего технологического развития, такие, как транспортные средства и оборудование, но также то, что мы обычно и не определяем как электронное вообще - такие, как продукты питания, одежда и кров, материалы, детали, сборочные узлы, умные дома и города

Эта статья рассматривает протоколы обмена сообщениями, которые появляются в качестве наиболее важных для промышленного Интернет, а также в широком IoT, для целевых систем, состоящих из тысяч устройств, соединенных в реальном времени по протоколу машина-машина (M2M).

Определение промышленного Интернета включает в себя два основных компонента:

- Подключение датчиков промышленных машин и приводов для локальной обработки и к Интернету (Интранету).
- Поступательное подключение к другим важным промышленным сетям, которые имеют самостоятельную ценность.

Основное различие между потребителем/социальным IoT и IIoT в том, как создается стоимость. Для потребительских/социальных IoT большинство стоимости идёт от рекламы. Стоимость IIoT создаётся из трех компонентов:

- повышения эффективности промышленного оборудования завода и долгосрочного технического обслуживания и управления оборудованием. От 10% до 25%.
- Снижение затрат на оборудование и его техническое обслуживание.
- развитие новых бизнес-моделей предприятия.

Приложения IIoT уже использующие эти технологии включают в себя: аэрокосмической и оборонной (управления воздушным движением, системы боевого управления, моделирование), здравоохранение (Smart

Devices), транспорт (железнодорожные системы управления, управление автомобилем), Smart Energy (крупномасштабные SCADA системы) и Смарт-Города.

Есть много различных протоколов обмена сообщениями. Однако существует подмножество наиболее важных протоколов, чтобы усилить IIoT. К ним относятся:

- DDS OMG
- AMQP OASIS
- MQTT OASIS
- JMS
- REST
- CoAP
- XMPP

Эта статья сравнивает и противопоставляет ключевые архитектурные особенности каждой парадигмы обмена сообщениями и обеспечивает контекст выбора правильного протокола для поддержки требований для конкретного типа системы.

II. Существующие протоколы обмена

DDS - OMG DDS стандарт является дата-центрическим pub/sub протоколом, который вышел из сообщества аэрокосмической и оборонной отраслей и стал применяться для критически важных систем в промышленности. Это позволяет строить масштабируемый, в режиме реального времени, надежный, высоко-производительный и совместимый обмен данными между издателями и подписчиками. DDS предназначен для удовлетворения потребностей критически важных бизнес-приложений, таких как финансовый трейдинг, управление воздушным движением, управление электросетями smart grid и других крупных приложений передачи данных. Он все чаще используется в широком диапазоне IIoT приложений.

DDS спецификация определяет:

- Дата-центрический Pub/Sub (DCPS) слой предоставляющий набор интерфейсов, которые представляют собой комплексный набор стандартизированных "профилей" ориентации в реальном времени информацией доступность доменов, начиная от небольших встроенных систем управления, вплоть до крупномасштабного управления корпоративной ИС.
- DDS совместимый проводной (wire) протокол (DDSI)
- расширяемые и динамические темы (Типы) для DDS стандарта

DDS является языком и независимой ОС. API-интерфейсы DCPS были реализованы в диапазоне различных языков программирования, включая Ada, C, C++, C#, Java, JavaScript, CofeeScript, Скала, Lua и Ruby. Использование стандартизированных API, помогает гарантировать, что приложения DDS могут быть с легкостью перенесены с реализаций различных поставщиков.

DDS также определяет протокол *wire*, в DDS Interoperability Wire Protocol [2], называемый DDSI. Протокол wire относится к механизму для передачи данных точка-точка. Протокол *wire* нужен, если более чем одно приложение должно взаимодействовать. В отличие от протоколов транспортного уровня (как TCP или UDP), термин wire протокол, используется для описания общего способа представления типов данных на уровне приложений. Все реализации DDS отвечающие DDSI будет взаимодействовать между собой. Протокол также поддерживает автоматическое "Дискавери", который позволяет участникам DDS объявить в сети данные о себе, что они могут предоставлять или какие данные они хотели бы получить, с точки зрения топика, типа и QoS. Протокол будет автоматически подключать соответствующих издателей к подписчиков. Это значительно упрощает процесс конфигурирования системы со многими узлами и многими устройствами, обменивающимися данными.

Недавнее дополнение к стандартам DDS является новой версией спецификации, включающей расширяемые и динамические типы топика [8], которая определяет топики как типы данных, которые могут быть расширены динамически, обеспечивая мобильность и взаимодействие приложений.

AMQP - это сообщение-центрический протокол, который возник из финансового сектора с целью освобождения пользователей от проприетарных и не совместимых систем обмена сообщениями. AMQP предписывает поведение провайдера сообщений и клиента в той мере, что решения от различных поставщиков действительно совместимы. Предыдущие попытки стандартизировать middleware произошло на уровне API (например, JMS) и, таким образом, не обеспечило совместимости. В отличие от JMS, который просто определяется в API, AMQP поддерживат wire протокол. Следовательно, любой продукт, который может создавать и интерпретировать сообщения, которые соответствуют этому формату данных, может взаимодействовать с любым совместимым другим протоколом (реализацией), независимо от языка программирования.

AMQP является двоичным протоколом уровня приложения, предназначеным для эффективной поддержки широкого спектра приложений для обмена сообщениями и паттернов (шаблонов). Он обеспечивает поток контролируемых сообщений с гарантиями доставки сообщений, такие как AT-MOST-ONCE (где каждое сообщение доставляется один раз или никогда), AT-LIST-ONCE (где каждое сообщение, несомненно, будет доставлено, но может оказаться так, что несколько раз) и EXACTLY-ONCE (когда сообщение будет всегда, конечно, доставлено и только один раз) и аутентификацию и/или шифрование на основе SASL и/или TLS. Это предполагает основной надежный протокол транспортного уровня, такой как протокол управления передачей (TCP).

MQTT - это сообщение-центрический wire протокол предназначен для M2M-коммуникаций, который позволяет передачу данных телеметрии в виде сообщений от устройств, с высокой латентностью и сетевыми ограничениями, к серверу или небольшому брокеру сообщений. Устройства могут варьироваться от датчиков и исполнительным механизмам, к мобильным

телефонам, встроенным системам на авто, или ноутбуки и полномасштабных компьютеров. Он поддерживает pub/sub стиль общения и очень прост.

JMS - является одним из наиболее широко используемых pub/sub технологий при обмене сообщениями. Это сообщение-центрический API для отправки сообщений между двумя или несколькими клиентами. JMS является частью Java Platform, Enterprise Edition (Java EE), и определяется в спецификации [5], разработанной в рамках процесса Java Community, как JSR 914. Это стандарт обмена сообщениями, который позволяет компонентам приложения, основанным на Java EE для создания, отправления, получения и чтения сообщений. JMS позволяет коммуницироваться между различными компонентами распределенных приложений, чтобы быть слабо связаными, надежными и асинхронными. JMS поддерживает точка-точка и pub/sub в стиле маршрутизации. Основным ограничением является то, что JMS является стандартом API Java только и не определяет wire протокол. Поэтому JMS решения от различных поставщиков не будет взаимодействовать.

REST - мигрировал как преобладающая модель проектирования Web API. Архитектура REST условно состоит из клиентов и серверов. Клиенты инициируют запросы к серверам; серверы обрабатывают запросы и возвращают соответствующие ответы. Запросы и ответы строятся вокруг передачи представлений ресурсов. Ресурс может быть по существу любым последовательным и содержательным понятием, которое может быть адресовано. Представление ресурса, как правило, это документ, который фиксирует текущее состояние или предназначенное ресурса.
REST был первоначально описан в контексте HTTP, но не ограничивается этим протоколом. RESTful архитектура может базироваться на других протоколах прикладного уровня, если они уже обеспечивают богатый и равномерный словарь для приложений, основанный на передаче значимой репрезентативной информации состояния.

CoAP - является протоколом передачи документов, который был разработан для использования с очень простыми электронными устройствами, что позволяет им общаться через Интернет. IETF CORE - рабочая группа, в настоящее время работает над стандартизацией CoAP.
CoAP предназначен для небольших маломощных датчиков, выключателей, клапанов и интернет устройств, таких как беспроводные сети датчиков (WSNs) и предназначен для легкой трансляции на HTTP для упрощения RESTful веб-интеграции. CoAP легкий, простой и использует UDP (не TCP) с поддержкой групповой адресации(multicast). Он часто используется в сочетании с WSNs исползующей IPv6 IETF через 6LoWPAN стандарт. Этот новый стандарт позволяет использовать IPv6 в маломощных и с потерями сетях (LLLNs), таких как те, которые основаны на IEEE 802.15.4.(WarelessHART)
CoAP поддерживает модель программирования клиент/сервер на основе RESTful архитектуры, в которой ресурсы являются сервером контролирующим абстракции, предоставляющие прикладные процессы и идентифицируемые посредством Universal Resource идентификаторов (URI). Клиенты могут манипулировать ресурсами, используя HTPP: GET, PUT, POST и DELETE методы. Он также обеспечивает в построении поддержку "дискавери" ресурсов в рамках протокола. Отображение между CoAP и HTTP также определяется, что позволяет прокси, чтобы быть построенным, чтобы обеспечить доступ к ресурсам в CoAP единообразно через HTTP.

XMPP - является протоколом для передачи потоковых XML элементов для того, чтобы обмениваться сообщениями и информацией о присутствии в близком к реальному времени. Этот протокол передачи документов был разработан для использования с очень простых электронных устройств, таких как мобильный телефон. Он является расширяемым, чтобы использоваться для реализации обмена мгновенными сообщениями, легкого middleware, голосовых и видео звонков, передачи файлов, игр, социальных сетей и приложений IoT. (ssp – *мдификации этого протокола лежат в основе многих современных мессенджеров*)

III. Какие проблемы мы пытаемся решить?

Протоколы обмена сообщениями, рассмотренные в этой статье, могут быть использованы для подключения различных устройств (например, датчики, мобильные устройства, одноплатные компьютеры, микро контроллеры, настольные компьютеры, локальные серверы, серверы в центре обработки данных) к распределенной сети (LAN или WAN) с помощью ряда проводных и беспроводных коммуникационных протоколов, включая: - Ethernet, Wi-Fi, RFID, NFC, ZigBee, Bluetooth, GSM, GPRS, GPS, 3G, 4G).

Проблема имеет ряд вариантов, которые могут быть классифицированы следующим образом:

- Inter Device communication - обмен сообщениями между узлами устройств на локальной сети (LAN),
- Device to Cloud communication - обмен сообщениями между узлом устройства и ЦОД в Интернет или между устройствами через Интернет
- Inter Data Center communication - обмен сообщениями между ЦОД через Интернет
- Intra Device communication, где сообщениями обмениваются процессы внутри того же устройства, хотя обычно это не считается случай использования IoT

Каждый протокол обмена сообщениями рассмотренный в данном документе, подходит для решения одной, нескольких или всех проблем, указанных выше и показан на рис.1.

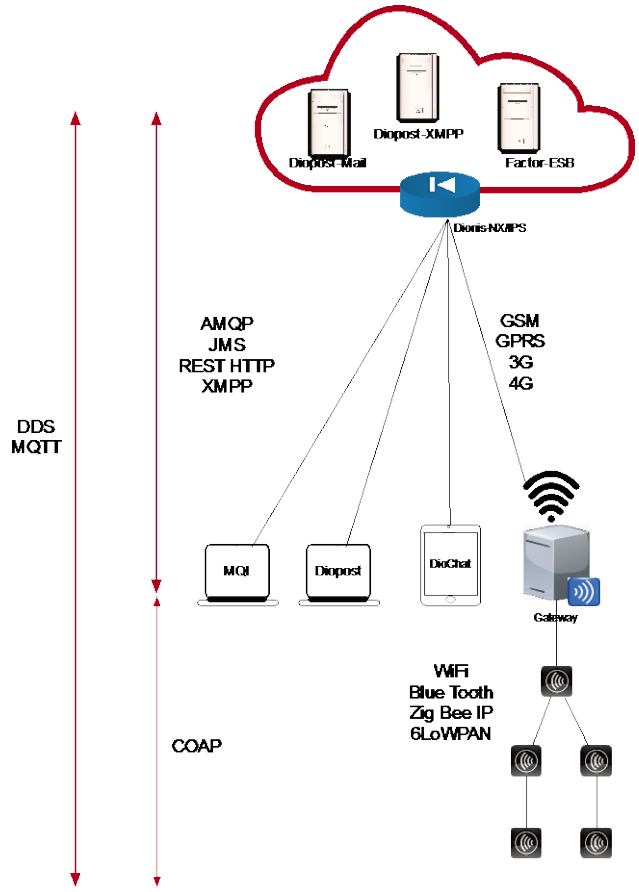

Рис. 1 Пространство протоколов

AMQP, MQTT, JMS, REST и XMPP были предназначены для запуска в сетях, использующих протокол TCP/IP в качестве основного транспорта. AMQP, MQTT и JMS поддерживают обмен сообщениями между узлами устройств через брокер сообщений в модели pub/sub (Inter Device). JMS ориентирован на Java-ориентированные системы, хотя существует ряд поставщиков, которые разработали собственные C и C++ JMS API интерфейсы которые могут быть использованы с провайдером JMS. REST поддерживает клиент-сервер (запрос / ответ) шаблон с использованием протокола HTTP. XMPP поддерживает клиентские устройства которые могут связываться друг с другом асинхронно с использованием протокола TCP/IP через сервер XMPP, который является посредником компонент (другие протоколы обращаются к данному компоненту в качестве посредника), который обеспечивает маршрутизацию между передающим и принимающим клиентами.

CoAP также базируется на RESTful архитектуре и схеме взаимодействия клиент/сервер. Он использует UDP в качестве основного транспортного протокола, а также может поддерживать IP multicast для того, чтобы разрешить групповые коммуникации между устройствами. CoAP был разработан, чтобы свести к минимуму накладные расходы на сообщение и уменьшить фрагментацию по сравнению с сообщением HTTP. При использовании UDP всё сообщение должно поместиться на одной дейтаграмме или одном IEEE 802.15.4 кадре, когда используется с 6LoWPAN.

AMQP, MQTT и JMS используют брокер для маршрутизации сообщений между издателями и подписчиками. Они могут сталкиваться с аналогичными вопросами относительно снижения производительности (увеличения задержки, снижения производительности) и в режиме реального времени предсказуемости как масштабных систем увеличивается (когда число издателей, подписчиков и узлов растёт). XMPP предоставляет общую основу для обмена сообщениями по сети и обычно используется для поддержки асинхронного обмена сообщениями между клиентами в сети. XMPP имеет децентрализованную архитектуру с каждым клиентом, подключенному к локальному серверу в своей области. Связь между доменами осуществляется через сервер-сервер федеративный протокол.

DDS был разработан для поддержки крупномасштабного обмена данными в режиме реального времени между устройствами в сети. Он используется во многих критически важных системах с большим устройство-устройство обменом данными, требующих эффективного, предсказуемого, с низкой задержкой и надежного обмена данными. Он может быть использован с надежными и ненадежными сетями. Надежность связи обеспечивается самим wire протоколом DDSI и не зависит от физического транспорта. По умолчанию DDS использует UDP в качестве его основного транспорта, но другие транспорты также могут поддерживаться (например, IP-multicast, TCP/IP, разделяемая память и т.д.). DDS является языком и независимой ОС и может работать на очень небольших встроенных устройствах (например, простой беспроводной датчик) и в крупномасштабных корпоративных системах.

DDS обеспечивает децентрализованную без брокерную (см раздел 6) архитектуру с прямой одноранговой сетью связи между издателями и подписчиками. DDS QoS позволяет пользователям точно настроить приоритеты обмена данными, чтобы обеспечить максимальную пропускную способность и уменьшить нагрузку на процессор и увеличить пропускную способность сети.
Все протоколы, может поддерживать распределённый обмен сообщениями между процессами на одном узле (внутри устройства). В случае AMQP, MQTT, JMS, REST и XMPP они обязаны использовать надежный транспорт, например, такой как TCP/IP, или UDP в случае DDS и CoAP. Тем не менее, CoAP, REST и XMPP не были предназначены для обмена сообщениями высокопроизводительных внутри одного узла и более целесообразно их использовать для обмена данными между узлами или с интернет-приложениями.

При проектировании установления соединения по архитектуре DDS приложение масштабируется лучше, чем другие протоколы, когда число приложений на узеле возрастает по передаче и приёму данных. Vortex PrismTech OpenSplice DDS обеспечивает опцию 'федеративного' развертывания, где несколько приложений на компьютере разделяют информацию

через разделяемую память и где сетевой трафик в / из, что федерация является арбитражном уникальным сети-планировщик на основе срочности и важности каждой обменять часть Информации. Это архитектура на основе развертывания разделяемой памяти имеет ультра-низкой задержку между основной коммуникации наряду с крайней узловой масштабируемостью. Это приводит к лучшей масштабируемости, более эффективному совместному использованию данных и больший детерминизм точка-точка на том же узле и между узлами.

AMQP, MQTT и JMS может обеспечить устройство для подключения центра обработки данных через Интернет с использованием протокола TCP/IP для соединения с брокерами, развернутыми в центре обработки данных. RESTful приложения могут реализовать обмен сообщениями запрос/ответ от клиента к серверу в центре обработки данных с помощью протокола HTTP. Опять JMS ограничивает этот тип конфигурации для приложений, написанных для платформы Java, которая может исключить его использование в самых ограниченных ресурсами окружающих сред. AMQP, MQTT и CoAP не определяют конкретный языковsq API для использования протокола и поставщики могут предоставлять реализации, которые могут поддержать ряд различных языков (например, Java, C++, C, C# и другие). Например, если ограниченое ресурсом бортовой сети устройство должно опубликовать сообщение (возможно, содержащее информацию сигнализации) для приложения управления, работающего в облачном центре обработки данных, то, как правило, имеет больше смысла, если приложение, запущенное на устройстве, написано на языке C, а не Java, для того, чтобы свести к минимуму его объем памяти.

CoAP узлы строятся для подключения устройства к ЦОД с помощью HTTP прокси, используя стандартное отображение.

XMPP клиенты могут общаться с серверами XMPP размещёнными в центре обработки данных с использованием протокола TCP/IP. Кроме того расширение (проект) [12] в основе стандарта XMPP позволяет использовать протокол для использования с HTTP соединениями. Благодаря децентрализованной архитектуре серверы XMPP могут быть федеративными между центрами обработки данных, если требуется. XMPP поддерживает широкий спектр языков программирования и сценариев, таких как Java, C++, C, C#, Erlang, Lisp, Ruby, Perl, Python и TCL.

Как следует из названия MQTT (Message Queuing телеметрии транспорт) цель - в сборе данных с устройств. Его цель заключается в сборе данных из многих устройств и транспортировки данных в центр обработки данных. Когда роль Центра данных является просто собирать и обрабатывать данные без необходимости совместного использования данных в режиме реального времени или других требований QoS, MQTT и CoAP полезные протоколы, так как они оба очень легкие и могут работать на мелых ресурсах устройства (например, низкая по мощности беспроводная сенсорная сеть), в то же время обеспечивая подключение к Интернет на основе приложений.

Благодаря вычислениям и платформе (например, Java для JMS) ресурсы, необходимые для хостирования AMQP, JMS и REST приложения лучше подходят для поддержки обмена сообщениями между приложениями, запущенными на серверах в локальной сети или в центре обработки данных или между Центров обработки данных через Интернет и не сильно ограниченными ресурсами встраиваемых сред. И REST/HTTP и клиенты XMPP поддерживают простые приложения обмена сообщениями, работающие на мобильных устройствах, таких как смарт-телефоны или планшеты.

По умолчанию, совместимые DDS реализации должны поддерживать UDP в качестве основного транспортного использую DDSI wire протокол. Тем не менее, ряд производителей также поддерживают DDSI реализации, которые также можно использовать TCP/IP (например, Vortex PrismTech в OpenSplice DDS), что позволяет подключение ЦОД через Интернет. DDS имеет то преимущество, что он может поддерживать низкую задержку, обмен данными в режиме реального времени, независимо от местоположения. Это включает в себя обмен данных устройство-устройство (в том числе через Интернет) и большой поток данных, где одно устройство публикует данные, которые потребляются многими абонентскими устройствами или даже других Центров обработки данных. Легкие реализации DDS могут поддерживать большинство ресурсов ограниченных средах с поддержкой API, доступных в диапазоне различных языков программирования.

IV. MESSAGE BRIKER ИЛИ DATABUS

Большинство реализаций AMQP, MQTT, JMS и XMPP базируются на основе брокера (рисунок 2).

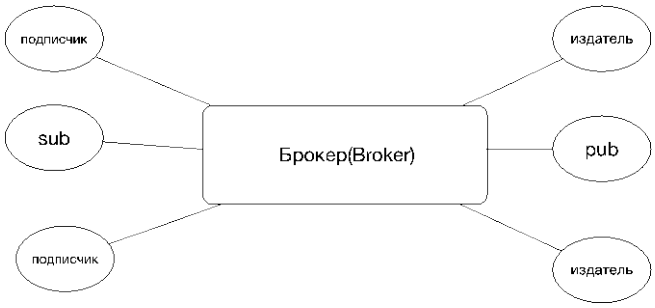

Рис.2 Message Broker архитектура

Издатели отправляют сообщения в доверенной среде маршрутизации сообщений и службы доставки, или брокер (стандарт XMPP относится к этому компоненту в качестве сервера), и подписчики зарегистрируют подписки с брокером, который также выполняет фильтрацию сообщений. Брокер обычно выполняет функцию "запомнить и передать" маршрутизации сообщений от издателей к подписчикам. Кроме того, брокер может приоритезировать сообщения в очереди до маршрутизации. Подписчики могут

зарегистрироваться для конкретного сообщения во время сборки, времени инициализации или выполнения.

Брокеры могут иметь одну очередь, несколько очередей сообщений, распространяемых среди них, копии каждого сообщения дублируются в каждой очереди. Возможны и другие шаблоны доставки. Гибкие шаблоны маршрутизации являются преимуществом использования брокера. Возникнув из финансового сектора, в котором обмен сообщениями является часто транзакционным, AMQP и JMS обеспечивают транзакционные режимы работы, которые позволяют им принимать участие в многофазных последовательностях операций. MQTT и AMQP брокеры поддерживают связь между издателем и подписчиком через TCP/IP, чтобы обеспечить надежную связь. Клиенты XMPP для связи с серверами и другими клиентами также используют TCP/IP, хотя XMPP не дает никаких гарантий надежности. Большинство брокеров JMS также реализованы на основе TCP, хотя это не является обязательным стандартом.

Брокеры могут быть развернуты в различных конфигурациях в сетевой среде в соответствии с конкретными потребностями системы. Общая конфигурация брокера включает в себя:

- Централизованный брокер - брокер, размещённый на одном централизованном сервере, и все потоки трафика идут через этот сервер. Эта модель проста в реализации и управлении, и требует наименьшего количества сетевых подключений. Однако брокер может стать единой точкой отказа, или горлышком бутылки, что не масштабируемо, а в системе в режиме реального времени не является предсказуемым.
- Несколько централизованных брокеров - в этой конфигурации каждая очередь или топик размещены на различных серверах. Эта модель является более сложной в реализации, но позволяет количество очередей и топиков масштабировать лучше. Больше клиентских соединений потенциально нужны, и каждый брокер становится единой точкой отказа. Когда многие издатели или подписчики обмениваются с одной и той же очередью, масштабируемость может стать проблемой. При такой конфигурации балансировка нагрузки может быть реализована путем федерализации топиков или очередей через другие брокеры.
- Децентрализованный брокер - наиболее децентрализованная архитектура в настоящее время используют IP Multicast на сетевом уровне. Система обмена сообщениями на основе многоадресной не имеет централизованного сервера сообщений. Некоторые функции сервера, такие как persistent, безопасность и транзакционность встраиваются в локальные части клиента, в то время как маршрутизация сообщений реализуется на сетевом уровне с помощью протокола IP Multicast. Децентрализованная брокерская модель, как правило, имеет гибридную архитектуру в том смысле, что издатель или подписчик при первом подключении к процессу демона, используют протокол TCP/IP, который в свою очередь взаимодействует с другими брокерскими процессами с использованием IP Multicast групп. Эта модель может масштабироваться до большого количества очередей, с низким количеством соединений при одновременном снижении единых точек отказа. Тем не менее, эта модель предлагает худшие задержки и предсказуемость в режиме реального времени по сравнению с другими моделями.

MQTT, AMQP и JMS не обеспечивают автоматическое обнаружение в отличие от DDS. Это означает, что конфигурация распределенной системы, которая использует одну из этих технологий, происходит через брокера. Издатели и подписчики обмениваются сообщениями через хорошо известные им очереди (и топики в случае JMS) и адреса брокеров/серверов. В случае JMS начальная ссылка на брокера JMS (провайдера), как правило, извлекаются из службы поиска, такой как Java Naming и Directory Interface (JNDI). XMPP определяет расширение протокола [13] для обнаружения информации о других серверах с XMPP. Два вида информации могут быть обнаружены: (1) идентичность и возможности субъекта, в том числе протоколов и функций поддержки; и (2) элементы, связанные с объектом, например, список номеров размещенных на услуге чата многопользовательской.
DDS поддерживает децентрализованную безброкерную архитектуру, с тем чтобы бесшовно обмениваться данными между производителями и потребителями. DDS основан на идее виртуальной "глобальной области данных", где производители пишут в пространство данных, и потребители, считывают из области данные. Модель данных, состоящая из наименований топиков, их определённых пользователем типов данных и связанными QoS используется инфраструктурой DDS, чтобы контролировать, как данные передаются. DDS соединяет производителей к потребителям по шине данных, как показано на рисунке 3.

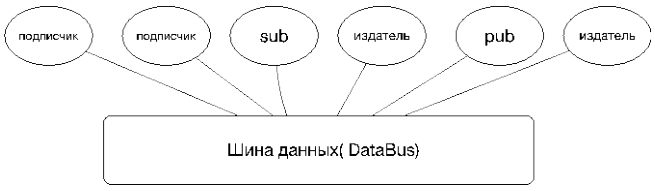

Рис. 3 Архитектура DataBus

DDS обеспечивает динамический механизм обнаружения автоматического соответствия DataReaders и DataWriters. DDS может работать через многие транспорты в том числе TCP/IP, UDP (одноадресный или многоадресной), разделяемую память или любой другой специализированный транспорт. Это не зависит от лежащего в основе транспорта для надежности, так как это предусмотрено wire протоколом DDSI. Стоит отметить, что есть DDS проекты, которые также используют брокера, но это исключения, и как правило, не оптимальны для большинства случаев.

V. DATA-CENTRICITY ИЛИ MESSAGE-CENTRICITY

AMQP, MQTT, JMS, REST, CoAP и XMPP - все технологии сообщение центрические. DDS с другой стороны - это технология дата-центрическая. Они обе могут делать подобные вещи в отношении обеспечения взаимодействия в распределенной системе, однако они делают это совсем по-разному. В системе сообщение-центрической акцент делается на доставку самого сообщения, независимо от полезной нагрузки данных и роль инфраструктуры является обеспечение, что сообщения доберётся до своих получателей.
В системе передачи дата-центрической акцент делается на определенных пользователем данные (модель данных). Единицей обмена в системе этого типа является значение данных. Middleware понимает контекст данных и гарантирует, что все заинтересованные подписчики получат правильное и последовательное представление данных. Это похоже на концепцию базы данных, которая может обеспечить глобальный взгляд (Рисунок 4) на данные и может управлять доступом.

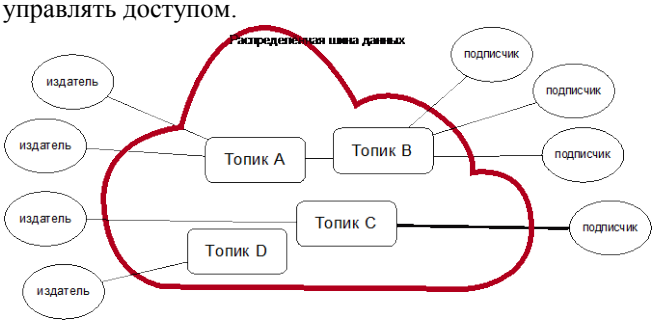

Рис 4. Концепция распределённой базы данных.

Одним из ключевых преимуществ дата-центрической технологии, такой как DDS является то, что совместное использование данных обеспечивает гораздо более высокий уровень абстракции для пользователей технологии. Данные является то, что пользователи понимают, так как он представляет собой что-то в своей области. Сообщение-центрические системы, с другой стороны обеспечивают более низкий уровень абстракции, как это требует, чтобы пользователи реализовали обмен путем обмена сообщениями данных. Дата-центрические системы поэтому проще поддерживать и расширять, что позволяет пользователям сосредоточиться на развитии их бизнес-логики, а не на написании логики обработки сообщений. DDS была разработана для поддержки сложных систем, по сути обеспечивая превосходную масштабируемость, фан-OUT характеристики и возможности управления состоянием. В дата-центрической системе приложения взаимодействуют с моделью данных, а не непосредственно друг с другом. Это помогает уменьшить сцепление и позволяет системе развиваться намного легче и динамичнее.

VI. ВЗАИМОДЕЙСТВИЕ

DDS позволяет разделять данные и определяет DDSI [2] wire протокол для обмена сообщениями между издателями и подписчиками. Протокол определяет стандартный формат представления данных, основанный на расширенных (CDR общее представление данных) правилах. DDS реализации, которые поддерживают DDSI полностью совместимы. Сообщения могут быть обменены и понятны различным реализациям DDS при условии, что они соответствуют стандарту DDSI.

MQTT - это проводной протокол сфокусированный на совместимый обмен сообщениями был разработан, чтобы быть открытым, простым, легким и просто реализуемым. Эти характеристики делают его идеальным для использования в стеснённых условиях. Например, когда сеть стоит дорого, имеет низкую пропускную способность или ненадежно или при запуске на встроенном устройстве с ограничениями на процессор или ресурсы памяти. MQTT представляет транспорт обмена сообщениями, который является независимым от содержания полезной нагрузки. Он не указывает, как данные представлены в сообщении. Хотя издатели и подписчики могут обмениваться сообщениями, приложения должны договориться о схеме сериализации в противном случае сообщения не могут быть поняты. В крупномасштабных распределенных системах это может быть трудно и дорого реализовать. MQTT был разработан для использования с TCP/IP.

Где JMS обеспечивает стандартный API обмена сообщениями для платформы Java, AMQP обеспечивает стандартный протокол обмена сообщениями для всех платформ. Как MQTT, AMQP не обеспечивает спецификацию для стандартного API промышленности, хотя в настоящее время некоторые работы, чтобы определить стандартные соответствия между протоколом AMQP и общих API, программирования (например, JMS). Он, тем не менее, обеспечивают спецификацию для стандартного wire протокола, чтобы описать, как сообщения должны быть структурированы и передается по сети.

AMQP сообщения имеют полезную нагрузку (данные, которые они несут), который AMQP брокеры обрабатывают в виде непрозрачного массива. Брокер не будет проверять или изменять полезную нагрузку. Вполне возможно сообщения содержат только атрибуты и не имеют полезной нагрузки. Оно является общим для использования форматов сериализации JSON и Thrift, Протокол буферов и MessagePack сериализовать структурированные данные, чтобы опубликовать их в качестве полезной нагрузки сообщения. AMQP как правило, использует "Content-Type" и "Content-Encoding" поля, чтобы сообщить эту информацию, но это лишь конвенции. Это означает, что, хотя издатели и подписчики могут обмениваться сообщениями, если схема сериализации данных не понятна обеим сторонам полезной нагрузки, то данные не могут быть интерпретированы. Один из вариантов здесь использовать AMQP системы типа, чтобы отправить структурированные, самостоятельно описания данных.
JMS обеспечивает стандартный API обмена сообщениями для платформы Java. С JMS можно заменить JMS-совместимый брокер сообщений с другой реализации с несколькими или без каких-либо изменений в исходном коде (обычно изменения

конфигурации требуется). Она также позволяет взаимодействовать между других языков платформы Java, таких как Скала и Groovy и обеспечивает уровень абстракции, который освобождает вас от необходимости беспокоиться о конкретных поставщика wire протоколы и различных JMS брокерах. JMS не обеспечивает стандарт для взаимодействия за пределами платформы Java или между других языках.

RESTful клиенты и серверы на базе HTTP могут взаимодействовать, поскольку все, что необходимо для поддержки обмена сообщениями в стек HTTP есть (либо на клиенте или на сервере). Почти каждая платформа и устройство, что сегодня так совместимости не является проблемой. Как HTTP, CoAP также поддерживает согласование содержания. Клиенты могут выражать предпочтительный представление ресурса и серверы могут сообщить клиентам, что они будут получать (Content-Type). Это позволяет клиентам и серверам независимо развиваться, добавляя новые представления, независимо, не влияя друг на друга.

XMPP поддерживает совместимость между различными клиентскими и серверными реализациями. Протокол использует TCP/IP сокеты для отправки и получения сообщений в формате XML. XMPP поддерживает асинхронную коммуникацию, используя потоки XML и строфы. Поток XML инкапсулирует конверт связи между двумя субъектами. Строфа содержит сообщения XML в виде текстовой строки, а также информацию о присутствии (например, я доступен или я занят). Клиенты и серверы XMPP совместимы и могут обмениваться сообщениями без предоставления смысла данных, которыми они обмениваются. Подобно MQTT, кодирования сообщение полезной нагрузки рассматривается на уровне приложения.

VII. Политики Quality of service

DDS обеспечивает чрезвычайно богатый набор политик качества обслуживания (QoS) для управления потоками данных через систему. Есть более чем 20 + QoS, определенные стандартом [1], что может быть использовано для управления надежностью, волатильностью, живучестью, использования ресурсов, фильтрации и доставки, право собственности, резервирование синхронизации сроков и задержки данных.

Объекты данных в системе DDS идентифицируются через топики, как показано на рисунке 5.

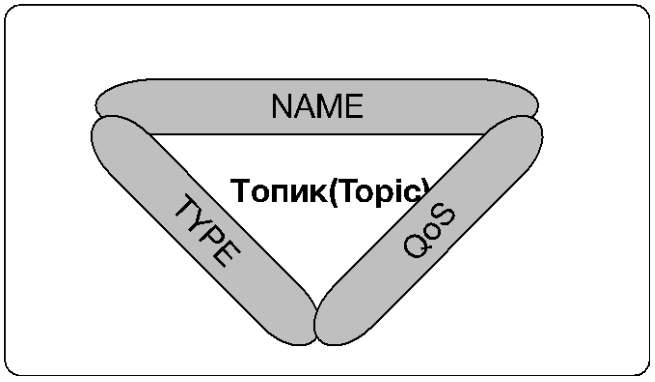

Рис 5. Идентификация объектов данных.

Когда DataReader топик совместим с DataWriter топиком, то "публикация" и "подписка" стали ассоциированными и данные опубликованы между ними. Топики совместимы, если они имеют то же имя, тот же тип данных и политики QoS не находятся в конфликте. Топик, DataReader, DataWriter, Издатель и подписчик - все имеют QoS политики. Политика QoS издателя, DataWriter, и топика управляет данными на передающей стороне. Политика QoS Подписчика, DataReader, и топика управляет данными на принимающей стороне.

Политики QoS DDS обеспечивают чрезвычайно мощный и гибкий механизм, который может использоваться, чтобы обеспечить потоковую оптимизацию потоков данных.

Протокол MQTT предоставляет очень базовую поддержку QoS для доставки сообщений между клиентами и серверами. QoS является атрибутом отдельного публикуемого сообщения с MQTT. Приложение устанавливает QoS для определенного сообщения, устанавливая поле MQTTClient_message.qos до требуемого значения. Подписчик-клиент может установить максимальное качество обслуживания сервера. В MQTTClient_subscribe () и MQTTClient_subscribeMany () функции установите этот максимум. QoS сообщений, присланных к абоненту, может отличаться на QoS данных к сообщению первоначального издателя. Нижний из двух значений используется для пересылки сообщения.

Три настройки QoS, предоставляемые MQTT являются:

- at-most-once - сообщение будет доставлено как минимум раз, или он не может быть доставлено вообще. Его доставка по сети не признана. Сообщение не сохраняется. Такое сообщение может быть утрачено, если клиент отключается, или если сервер сбойнул. Это самый быстрый режим передачи. Это иногда называют "выстрелил и забыл". Протокол MQTT не требует серверов для пересылки публикаций для клиента. Если клиент отключается на время, сервер получает публикацию, издание может быть отброшено, в зависимости от реализации сервера.
- at-least-once - сообщение всегда поставляется, по крайней мере, один раз. Оно может быть доставлено несколько раз, если происходит сбой, прежде чем подтверждение получено отправителем. Сообщение должно быть сохранено локально на стороне отправителя, пока отправитель не получит подтверждения, что сообщение было получено по назначенному получателю. Сообщение сохраняется в случае, если сообщение должно быть отправлено снова.
- Exactly-once - сообщение всегда поставляется только один раз. Сообщение должно быть сохранено локально на стороне отправителя, пока отправитель не получает подтверждение, что сообщение о том, что сообщение было

получено по назначенному получателю. Сообщение сохраняется в случае, если сообщение должно быть отправлено снова. Это самый безопасный, но медленный режим передачи. Более сложный handshaking и asknowledgement последовательность происходит для используется обеспечения отсутствия дублирования сообщений.

Сообщения JMS имеют ряд свойств QoS, которые могут быть установлены. Эти свойства QoS включают следующее:

- JMSDeliveryMode - Есть два типа режимов доставки в JMS: стойкий и нестойкий. Стойкое сообщение должно быть доставлено раз и только один раз-это означает, что сообщение не теряется, если поставщик JMS сбойнул, он будет доставлен после того как сервер восстанавливается. Не стойкое сообщение доставляется более одного раза, что означает, что оно может быть потеряно и никогда не доставлено, если поставщик JMS засбоит. В обоих стойких и нестойких режимах доставки сервер сообщений не должен посылать сообщение к тому же потребитель не один раз, но это возможно. В целом, нестойкие сообщения лучше, чем стойкие сообщения. Они поставляются быстрее и требуют меньше системных ресурсов на сервере сообщений. Тем не менее, не стойкие сообщения должны быть использованы только тогда, когда потеря сообщений из-за сбоя провайдера JMS не является проблемой.
- JMSPriorityPurpose - сообщениям может быть назначен приоритет производителя сообщения, когда они передаются. Серверы сообщений могут использовать приоритет сообщения по заказу доставки сообщений для потребителей; Сообщения с более высоким приоритетом поставляются впереди нижних приоритетных сообщений

AMQP сообщения имеют аналогичные свойства QoS, как и в MQTT. Это включает в себя поддержку очереди сообщений и семантику доставки ag-most-once, at-list-once и once-and-only-once (надежный обмен сообщениями).

QoS REST обеспечивается основой транспорта. HTTP является протоколом прикладного уровня. Его определение предполагает в основной и надежный протокол транспортного уровня и наиболее часто здесь используется TCP/IP. Однако HTTP может использовать и ненадежные протоколы, такие как протокол дейтаграмм пользователя (UDP), например, в простом сервисе протокола обнаружения (SSDP)(Discovery).

COAP предусматривает только рудиментарные свойства QoS доставки сообщений. Запросы COAP и ответные сообщения могут быть помечены как "confirmable" или "nonconfirmable". Confirmable сообщения должны быть признаны приемником ACK пакета. Nonconfirmable сообщения являются в режиме "fire and forget".

XMPP не предоставляет никаких явных поддержек качества обслуживания и надежной доставки сообщений. Это может быть сделано с помощью простой порядковый номер атрибуты строф.

VIII. ПРОИЗВОДИТЕЛЬНОСТЬ

В очень простой конфигурации точка-точка между узлами AMQP, MQTT, JMS rest / HTTP, COAP и DDS могут иметь сопоставимые характеристики, хотя брокер-маршрутизация добавляет дополнительные накладные расходы по сравнению с безброкерской инфраструктурой, таких как DDS, HTTP или COAP.

При развертывании нескольких экземпляров брокера и федераций очереди через брокеров, может быть достигнуто увеличение пропускной способности и масштабируемости сообщений.

Запросы к CoAP ресурсам от клиента HTTP имеют дополнительные накладные расходы от того, чтобы быть направлены через CoAP/ HTTP прокси-сервер. CooAP следует рассматривать только тогда, когда низкая задержка и производительность в реальном времени не является обязательным требованием. Поддержка CoAP для IP multicast означает, что один клиент CoAP может выдать тот же запрос к нескольким серверам одновременно (COAP например один запрос может быть выдан в нескольких беспроводных устройств, чтобы выключить все огни на улице сразу). CoAP также имеет возможность опрашивать ресурс. Когда соответствующий флаг наблюдения установлен, запрос GET сервер может продолжать отвечать после первоначальной передачи документа, позволяя передавать изменения состояния на сторону клиента, как только они происходят.

Когда DDS развертывается в локальной сети, коммуникации между издателями и подписчиками происходят через UDP multicast и в сочетании с богатым и гибким набором политик QoS позволяет исключительную масштабируемость. Реализация DDS может надежно масштабироваться до десятков 1000е сообщений/сек на одноранговых сетях, состоящих из тысяч устройств (см. http://www.prismtech.com/vortex/vortex-opensplice/performance). В режиме реального времени, где задержка системы измеряется в микросекундах и предсказуемость доставки данных является ключевым требованием только DDS из технологий передачи сообщений, рассмотренных в данном документе, может обеспечить управление синхронизацией, необходимой для этих типов систем.

В некоторых типах (не в системах реального времени), таких как бизнес для бизнеса (B2B), бизнес для потребителя (B2C) или финансовых приложений, пропускная способность сообщения не может быть главной проблемой по сравнению с необходимостью надежной транзакционной доставки сообщения. В этих случаях AMQP и JMS реализации имеют явные преимущества.

XMPP не считается протоколом высокой производительности по сравнению с другими протоколами, обсуждаемыми в статье. XML является текстовой базой, так что XMPP имеет более высокую загрузку сети и затраты на обработку по сравнению с бинарными протоколами. Как и в других брокерских протоколах, маршрутизация сообщений через сервер XMPP также несёт дополнительные накладные расходы

по сравнению с протоколом одноранговой сети, например, DDS. В высокопроизводительных IIoT, где низкая задержка сообщений и высокая пропускная способность, в сочетании с надежной и предсказуемой доставкой данных требуются, XMPP не подходит.

IX. БЕЗОПАСНОСТЬ

Для того, чтобы построить доверенную и отказоустойчивую систему вопросы безопасности должны быть рассмотрены в первую очередь. Они включают в себя, как защитить коммуникации, как управлять аутентификацией и контролем доступа к ресурсам в системах, которые могут состоять из тысяч устройств и как обеспечить целостность и конфиденциальность данных в системе.

JMS не предоставляет API для управления конфиденциальностью и интеграцией сообщений. Он также не определяет, как цифровые подписи или ключи распределяются между клиентами. Безопасность считается проблемой для спецификации каждого провайдера JMS. Основные провайдеры JMS обеспечивают различные уровни функциональности собственной безопасности в рамках их реализации. Как правило, это включает в себя обеспечение возможности для обеспечения аутентификации клиента и контроль доступа к очередям JMS и топикам. Transport Layer Security (TLS) и его предшественник, Secure Sockets Layer (SSL), которые являются криптографическими протоколами часто используются для обеспечения безопасности связи. Они используют асимметричную криптографию для проверки подлинности обмена ключами, симметричное шифрование для конфиденциальности и аутентификации сообщений для целостности сообщения. В ряде ведущих реализаций JMS, служба Java аутентификации и авторизации (JAAS) используется для обеспечения подключаемой аутентификации и авторизации поддержки провайдера.
В MQTT v3.1, имя пользователя и пароль могут быть переданы в пакете MQTT. Это может упростить идентификацию отдельных клиентов в системе за счет уменьшения числа ключей, которые должны быть распределены и управляться по сравнению с эксклюзивной ключевой системой. Шифрование данных при обмене по сети может осуществляться независимо от протокола SSL с помощью MQTT или TLS.

С AMQP, уровни безопасности, как ожидается, должны быть определены с внешней стороны AMQP спецификации, например использование TLS для шифрования данных. Исключением является использование простого слоя аутентификации (SASL), который указан в стандарте и может быть использован для того, чтобы AMQP имел согласованный механизм аутентификации.

Группа OMG приняла новую DDS Спецификацию безопасности [14] в дополнение к существующим проприетарным решениям безопасности, используемых ведущими производителями DDS в процессе внедрения стандарта. Новая OMG спецификация безопасности DDS определяет модель безопасности и сервис плагин интерфейс (SPI) для совместимых реализаций DDS. DDS модель безопасности вызывается SPI, по реализации DDS. Спецификация также определяет набор встроенных реализаций этих SPI.

- Специфицированные встроенные SPI реализации позволит использовать и коробочную безопасность и совместимости между совместимыми приложениями DDS.
- Использование SPI позволяет пользователям DDS настроить поведение и технологии, чтобы реализация DDS использовалась для обеспечения безопасности информации, в частности, позволяет настраивать аутентификацию, управление доступом, шифрование, аутентификацию сообщений, ЭЦП, логирование и разметку данных.

В RESTful системе обеспечивается обмен сообщениями по протоколу HTTP, как правило, через SSL. Именуемый как HTTPS, HTTP, был первый протокол, чтобы использовать протокол SSL. HTTP также может быть использован с новыми реализациями TLS. Оба SSL и TLS обеспечивают HTTP клиентам и серверам с асимметричной криптографией для аутентификации обмена ключами и симмертричное шифрование для конфиденциальности.

CoAP построен на UDP и как таковой он не может полагаться на SSL/TLS (доступен с TCP/IP), чтобы обеспечить возможности безопасности. В случае UDP, Datagram Transport Layer Security (DTLS) предоставляет те же гарантии, как TCP, но для обмена сообщениями через UDP.

Ряд функций безопасности были построены в ядре спецификаций XMPP. В частности, соединение идентифицируется с простой аутентификации и безопасности Layer (SASL) и шифруются с Transport Layer Security (TLS).

X. ЗАКЛЮЧЕНИЕ

Ряд ключевых протоколов обмена сообщениями появились для поддержки нового поколения приложений IoT. Протоколы обмена сообщениями, рассмотренные в этом документе, включают в себя DDS, MQTT, AMQP, JMS, REST, CoAP и XMPP, каждый из которых может быть использован для подключения устройств в распределенной сети. Тем не менее, их пригодность для поддержки различных операционных сценариев, рассмотренных, в том числе Интер и Интра устройства связи, устройство для Cloud связи и Интер ЦОД связи изменяется, особенно когда приняты во внимание ключевые системные требования, такие как производительность, качество обслуживания, интероперабельность, обеспечение отказоустойчивости и безопасности.

Для приложений "устройство - устройство", которые требуют высокой производительности, режима реального времени, многие-ко-многим DDS имеет явные преимущества перед другими протоколами обмена сообщениями. DDS также становится ключевым фактором для подключения сетей устройств в режиме реального времени в облачном ЦОДе.

Выбор наиболее подходящего решения для обмена сообщениями должен базироваться на понимании как архитектуры и требований сообщение/данные

совместного использования данных для каждой целевой системы.

Благодарности

Автор выращает благодарность Дмитрию Намиоту за поддержку и участие при подготовке статьи к публикации.

Библиография

[1] Data Distribution Service for Real-Time Systems Version 1.2, OMG Available specification formal/07-01-01
[2] The Real-Time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification Version 2.1, OMG Document Number: formal/2009-01-05
[3] MQ Telemetry Transport (MQTT) Specification v3.1, IBM, Eurotech
[4] Advanced Message Queuing Protocol (AMQP) Version 1.0 Specification, OASIS
[5] Java Message Service, Nigel Deakin, Oracle, Version 2.0, March 2013
[6] Constrained Application Protocol (CoAP) draft-ietf-core-CoAP-18, June 28, 2013
[7] MQTT For Sensor Networks (MQTT-S) Protocol Specification, Version 1.2, June 6, 2011
[8] Extensible and Dynamic Topic Types for DDS, Version 1.0 OMG Document Number: formal/2012-11-10
[9] Extensible Messaging and Presence Protocol (XMPP): Core, RFC 6120, March 2011
[10] Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, RFC 6121, March 2011
[11] Extensible Messaging and Presence Protocol (XMPP): XEP-0060: Publish-Subscribe (draft specification), July 2010
[12] Extensible Messaging and Presence Protocol (XMPP): XEP-0124: Bidirectional-streams Over Synchronous HTTP (draft specification), April 2014
[13] Extensible Messaging and Presence Protocol (XMPP): XEP-0030: Service Discovery, June 2008
[14] DDS Security Specification , FTF Beta 1, OMG Document Number: ptc/2014-06-01
[15] QoS Reference Guide https://community.rti.com/rti-doc/500/ndds.5.0.0/doc/pdf/RTI_CoreLibrariesAndUtilities_QoS_Reference_Guide.pdf
[16] CDR https://megapredmet.ru/1-710.html

Industrial Application Architecture IoT and protocols AMQP, MQTT, JMS, REST, CoAP, XMPP, DDS

S. Seleznev, V. Yakovlev

***Abstract* — The most important messaging protocols proposed as the basis for the new generation of the Internet of Things (IOT) and, more specifically, industrial Internet applications (IIoT) are discussed in this article. Understanding the architecture and message separation requirements in each target system influences the choice of the most appropriate solution for operational technologies (OT) or process control systems.**

If the target system needs to process thousands or more of other network devices, the choice of AMQP, MQTT, JMS, REST, or XMPP protocols can result in poor performance and particular complexity. CoAP supports IP Multicast, allowing one request to publish a message to multiple devices simultaneously.

Unlike DDS, which provides support for dynamic discovery (Discovery), setting up a system that uses AMQP, MQTT or JMS through a broker is required. Access to the broker is usually implemented using a well-known network address or search services, such as JNDI.

DDS can provide real-time, many-to-many, managed connections required by high-performance applications (device-to-device). DDS is also becoming the key messaging protocol for connecting real-time device networks to cloud-based data centers (Cloud - Fog - Cloudlet). The specific implementation of DDS (there are about a dozen independent manufacturers in the world today) can offer a unique way to exchange high performance data.

Ensuring that the system is fault tolerant and secure is a key factor in the IoT world, consisting potentially of many thousands of messaging devices. Lead implementations typically provide their own solutions based on SSL or TLS, as well as s / Mime. AMQP and XMPP specify the use of SASL to provide a pluggable message authentication interface, and a comprehensive security system is standardized in the accepted OMG DDS specification.

***Keywords* — Internet of Things, IoT, IIoT, protocols, messages.**

REFERENCES

[1] Data Distribution Service for Real-Time Systems Version 1.2, OMG Available specification formal/07-01-01

[2] The Real-Time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification Version 2.1, OMG Document Number: formal/2009-01-05

[3] MQ Telemetry Transport (MQTT) Specification v3.1, IBM, Eurotech

[4] Advanced Message Queuing Protocol (AMQP) Version 1.0 Specification, OASIS

[5] Java Message Service, Nigel Deakin, Oracle, Version 2.0, March 2013

[6] Constrained Application Protocol (CoAP) draft-ietf-core-CoAP-18, June 28, 2013

[7] MQTT For Sensor Networks (MQTT-S) Protocol Specification, Version 1.2, June 6, 2011

[8] Extensible and Dynamic Topic Types for DDS, Version 1.0 OMG Document Number: formal/2012-11-10

[9] Extensible Messaging and Presence Protocol (XMPP): Core, RFC 6120, March 2011

[10] Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, RFC 6121, March 2011

[11] Extensible Messaging and Presence Protocol (XMPP): XEP-0060: Publish-Subscribe (draft specification), July 2010

[12] Extensible Messaging and Presence Protocol (XMPP): XEP-0124: Bidirectional-streams Over Synchronous HTTP (draft specification), April 2014

[13] Extensible Messaging and Presence Protocol (XMPP): XEP-0030: Service Discovery, June 2008

[14] DDS Security Specification, FTF Beta 1, OMG Document Number: ptc/2014-06-01

[15] QoS Reference Guide https://community.rti.com/rti-doc/500/ndds.5.0.0/doc/pdf/RTI_CoreLibrariesAndUtilities_QoS_Reference_Guide.pdf

[16] CDR https://megapredmet.ru/1-710.html