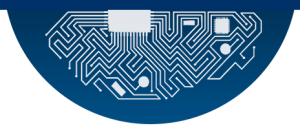
СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ



А. Ю. Филатов

Data Distribution Service

Фрагмент конспекта

СПбГЭТУ «ЛЭТИ», 2022 г.

1 Data Distribution Service

1.1 Что такое Data Distribution Service?

Введение

Data Distribution service - это протокол передачи сообщений между узлами программного обеспечения. Стандартизацией этого протокола глобально занимается OMG (Object Management Group https://www.omg.org/)

Основная цель DDS - обеспечить передачу правильных сообщений между правильными адресатами в правильное время. Это означает, что существует три задачи, решаемые в рамках DDS:

- Определить адресата, которому следует передать сообщение. DDS динамически определяет публикатора, подписчика и структуру сообщения. DDS формирует логический канал связи между публикатором и подписчикам и берёт на себя задачу разделения между ними памяти, в которой записано сообщение. Подписчику должен быть уверен, что публикатор может что-то прислать.
- Доставить сообщение в нужный промежуток времени. В случае работы с системой реального времени необходимо поддерживать расходы на передачу сообщений на таком уровне, чтобы доставка сообщений укладывалась в ограничения системы.

DDS использует политики QoS (Quality of Service) для определения баланса между скоростью передачи сообщений и надёжностью. QoS - это способность системы обеспечить необходимый сервис заданному трафику в определенных технологических рамках.

Для большинства систем качество связи определяется четырьмя параметрами:

- Скорость передачи информации (Bitrate). Измеряется в Мб/сек
- Задержка при передаче сообщений (Delay). Измеряется в мс
- Колебание задержки при передаче сообщений (Jitter). Измеряется в мс.
- Потеря сообщений (Packet Loss). Измеряется в процентах
- Существует 3 модели QoS:

- Негарантированная доставка (Best Effort Service). Увеличение скорости работы осуществляется за счёт расширения канала связи
- Интегрированный сервис (integrated Service). Необходимая пропускная способность гарантируется протоколом резервирования сетевых ресурсов RSVP, который обеспечивает выполнение требований ко всем промежуточным узлам
- Дифференцированное обслуживание (Differentiated Service). Для передачи трафика вводится несколько классов, для каждого из которых вводится свой уровень QoS

DDS - это централизованная система передачи информации. На сайте OMG этот факт указан как достоинство для применения к Интернету вещей. Централизованность данных позволяет включать в сообщения информацию контекста, что упрощает восприятие получаемой информации.

Концептуально DDS как будто владеет локальным хранилищем данных, называемым «глобальным пространством данных». Для приложения глобальное пространство данных выглядит как локальная память, доступ к которой осуществляется через API. С точки зрения разработчика передача сообщений выглядит, как работа с собственной памятью. На самом деле DDS отправляет сообщения для обновления соответствующих хранилищ на удаленных узлах.

Основные достоинства DDS:

- Простота интегрирования. Поскольку система обмена сообщениями является централизованной, она позволяет инкапсулировать от пользователя абстракцию разделения памяти, предоставив ему вместо этого API, аналогичное с работой с локальной памятью
- Производительность и масштабируемость. При правильной настройки DDS можно добиться передачи сообщений между разными узлами локальной сети в 30 мкс. Также правильно настроенная DDS достигает почти линейную масштабируемость - при добавлении в локальную сеть нового узла средняя скорость обмена сообщения увеличивается не более, чем на константное время
- Безопасность. DDS реализует стандартизованные механизмы аутентификации, шифрования данных, логирования.

Открытый стандарт. Поддержка QoS. Реализация end-to-end протокола обмена данными, а также device-to-device, cloud-to-device и device-to-cloud.

Существует множество реализаций DDS, как открытых и бесплатных, так и закрытых. К наиболее распространённым бесплатным реализациям можно отнести OpenDDS, OpenSplice, ROS2.

Их все объединяет общий подход к реализации обмена сообщений. Ниже представлен псевдокод этого похода.

Пример действий для различных сущностей в DDS:

Публикатор:

```
публикатор = Зарегистрировать_публикатора_в_системе(имя_топика)
сообщение = Создать_сообщение_для_топика(имя_топика)
Заполнить_сообщение_данными(сообщение)
публикатор.Отправить_сообщение(сообщение)
```

Подписчик:

```
подписчик = Зарегистрировать_подписчика_в_системе(имя_топика, Колбэк)
подписчик.Быть_готовым_к_приёму

функция Колбэк(сообщение):
    Обработать_сообщение(сообщение)
```

Создание протоколов обмена для узлов робототехнической системы

Вызов сервиса из программы и из консоли. Связь сообщений, передаваемых через Topic и через Service. Реализация собственного типа сообщений для сервиса. Обработка ответа на Service. Одновременное применение передачи информации через Topic и Service.

OpenDDS

OpenDDS - это C++ реализация спецификация OMG DDS с открытым исходным кодом. Приложения Java могут использовать OpenDDS через привязки JNI. OpenDDS был разработан и представлен с открытым исходным кодом от Object Computing.

OpenDDS использует интерфейсы Interface Definition Language(IDL), определяемые спецификацией DDS, для инициализации и управления сервисом. Передача данных осуществляется через специфическую транспортную среду.

Транспортные протоколы, которые поддерживает OpenDDS:

- TCP/IP
- UDP/IP
- многоадресная рассылка IP
- sharedmemory
- RTPS_UDP

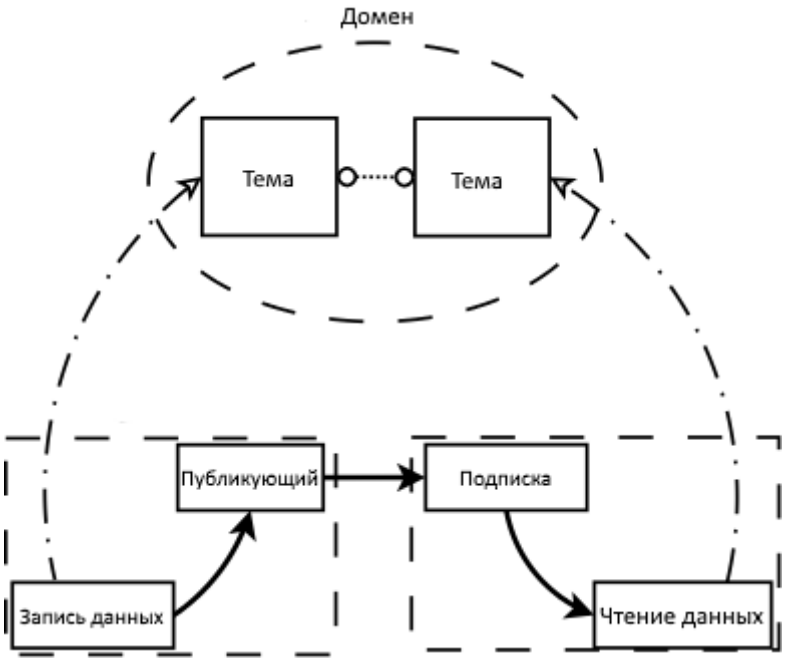


Рис.1 - Схема сущностей OpenDDS

Передача информации осуществляется в рамках одного пространства(domain). Между двумя пространствами передача информации по обычным каналам не предусматривается. На количество топиков, участников в рамках одного домена не накладывается ограничение. Участники общаются друг с другом через каналы. Но делают это не напрямую, а используя дочерние объекты. Так, у участника могут быть издатель (publisher) и подписчик (subscriber). У одного участника может быть несколько издателей и несколько подписчиков, в то время как один издатель или один подписчик может принадлежать только одному участнику. Таким образом, связь между участником и издателем/подписчиком оказывается “один к многим”. Непосредственно передачей или приемом сообщений через канал занимается писатель

(datawriter) и читатель (datareader) - атрибуты издателя и подписчика соответственно. Один писатель (читатель) привязан только к одному каналу, однако в один канал может писать (читать) несколько писателей (читателей) даже из разных издателей (подписчиков) и даже из разных участников, делая связь между ними типа “многие к одному”. Один издатель (подписчик) может иметь несколько писателей (читателей), делая связь между этими объектами типа “один к многим”. Согласно концепции, издатель не может иметь читателя, а подписчик - писателя.

При помощи OpenDDS приложения системы могут быть настроены для обнаружения DCPSInfoRepo или распределённой RTP. Спецификация OMG DDS описывает подробности обнаружения для реализаций. OpenDDS предоставляет две опции для обнаружения:

1. Информационный репозиторий: стиль централизованного репозитория, который исполняется как отдельный процесс, позволяющий издателям и подписчикам обнаруживать друг друга централизованно.
2. Обнаружение RTPS: одноранговый стиль обнаружения, который использует протокол RTPS для объявления о доступности и местоположении в сети.

Open Splice

Vortex OpenSplice - это одна из реализаций стандарта Object Management Group Data Distribution Service (OMG DDS). Цель - предоставить инфраструктуру и программное обеспечение для распределенных систем реального времени. OpenSplice поддерживает кросс-платформенную разработку на таких языках программирования, как C, C++, C#, Java, Python, а также в среде Node.js.

OpenSplice реализует федеративную модель распределенной архитектуры. То есть отдельно взятый процесс демона поддержки DCPS(Data-Centric Publish and Subscribe) для каждого сетевого интерфейса должен быть доступен и запускаться на каждом узле до того, как участники смогут общаться. Таким образом обеспечивается оптимизация объединения и пакетной обработки данных, которые OpenSplice использует по умолчанию.

OpenSplice DCPS поддерживает RTPS и RTNetworking протоколы доставки данных.

Структуры хранения данных

Позволяет объединить приложения DDS и администрирование Vortex OpenSplice в рамках одного процесса операционной системы. Нужен в средах, где общая память недоступна или нежелательна. Каждое приложение DDS на узле обработки реализовано как отдельный автономный процесс операционной системы, то есть приложение с одним процессом. Связь между несколькими такими приложениями, размещенными на одном и том же машинном узле, осуществляется через (кольцевую) сеть, из-за того, что между ними нет общей памяти. Существует возможность расширения архитектуры одного процесса путём создания библиотек приложений которые можно «связать» в один процесс.

Shared Memory Architecture

В архитектуре с разделяемой памятью данные физически присутствуют только один раз на любой машине, но интеллектуальное администрирование предоставляет каждому подписчику его особое личное представление об этих данных. Приложения DDS и интерфейс администрирования Vortex OpenSplice напрямую связаны с общей памятью, которая создается Vortex OpenSplice демоном при запуске. Любой процесс может присоединиться к своему собственному виртуальному пространству после того, как будет создан разделяемый сегмент памяти, и затем может работать с ним как с обычным сегментом памяти. Эта архитектура позволяет рассматривать кэш данных подписчика как отдельную базу данных, а её содержимое можно фильтровать, запрашивать и т. д. с помощью профиля подписки.

Сущности и примеры их реализации на C

Сущность - это базовый блок DCPS. Является либо производителем информации (Publisher или DataWriter), либо потребителем информации (Subscriber или DataReader), либо соединителем приложения и сетевого домена, также является фабрикой для сущностей (DomainParticipant), либо

каналом связи производителей и потребителей информации (Topic). На поведение каждого объекта в OpenSplice можно повлиять с помощью политик QoS, которые должны быть связаны с ним во время создания. Чтобы отслеживать статус связи сущности, из нее можно получить объект StatusCondition или к ней можно прикрепить объект Listener. Сущность может быть создана или удалена только с использованием соответствующей фабрики.

Domain Participant

Глобальное пространство данных может быть организовано в домены(Domains), которые, в свою очередь, могут иметь разделы(Partitions). DDS сущности всегда должны принадлежать определенному разделу домена. Каждый раздел сопоставлен с IP-адресом многоадресной рассылки. Объект Domain Participant является фабрикой для сущностей.

Пример создания :

```
// идентификация участника домена
DDS_DomainId_t domain = DDS_DOMAIN_ID_DEFAULT;
// получаем фабрику для сущностей
DDS_DomainParticipantFactory dpf = DDS_DomainParticipantFactory_get_instance();
// создаём участника домена, используя фабрику
DDS_DomainParticipant dp =DDS_DomainParticipantFactory_create_participant (dpf, domain, DDS_PARTICIPANT_QOS_DEFAULT, NULL, DDS_STATUS_MASK_NONE);
```

QoS

QoS(Qaulity of Service) регулирует нефункциональные свойства информации, такие как надёжность, доступность, своевременности и другие, в глобальном пространстве данных. QoS у писателей и читателей должны быть совместимы, например, невозможно сопоставить издателя, который ненадежно доставляет данные, с подписчиком, который требует надежности.

Topic

Задается структурой, объединяющей тип, уникальное имя и установленную QoS.

Пример создания :

```
#define Chat_MsgTypeSupport DDS_TypeSupport
//выделение объекта Chat_MsgTypeSupport в куче
Chat_MsgTypeSupport ts = Chat_MsgTypeSupport__alloc (void);
// регистрация типа данных в DDS_DomainParticipant
DDS_ReturnCode_t status = Chat_MsgTypeSupport_register_type (Chat_MsgTypeSupport ts, DDS_DomainParticipant domain, DDS_string name );
// создание топика
DDS_Topic myTopic = DDS_DomainParticipant_create_topic( dp, "Chat_Msg", chatMessageTypeName, DDS_TOPIC_QOS_DEFAULT, NULL, DDS_STATUS_MASK_NONE);
```

Pub/Sub

Несут ответственность за управлением данными с помощью DataWriter и DataReader, каждый из которых связан только с одним писателем/читателем и с одним топиком.

Пример создания писателя и публикации им сообщения:

```
DDS_PublisherQos *pub_qos = DDS_PublisherQos__alloc();
status = DDS_DomainParticipant_get_default_publisher_qos (dp, pub_qos);
// создание писателя
DDS_Publisher chatPublisher = DDS_DomainParticipant_create_publisher(dp, pub_qos, NULL, DDS_STATUS_MASK_NONE);
// создание писателя данных для топика
Chat_MsgDataWriter talker = DDS_Publisher_create_datawriter(chatPublisher, myTopic, DDS_DATAWRITER_QOS_USE_TOPIC_QOS, NULL, DDS_STATUS_MASK_NONE);
// зарегистрируем сообщение для этого пользователя, предварительно нужно выделить ресурсы
Chat_Msg *msg = Chat_Msg__alloc();
DDS_InstanceHandle_t userHandle = Chat_MsgDataWriter_register_instance(talker, msg);
// публикуем сообщение
status = Chat_MsgDataWriter_write(talker, msg, userHandle);
```

Пример создания читателя и чтения им сообщения:

```
DDS_SubscriberQos *sub_qos = DDS_SubscriberQos__alloc();
status = DDS_DomainParticipant_get_default_subscriber_qos (dp, sub_qos);
// создание читателя
DDS_Subscriber chatSubscriber = DDS_DomainParticipant_create_subscriber(dp, sub_qos, NULL, DDS_STATUS_MASK_NONE);
// создание читателя данных для топика
Chat_MsgDataReader mbReader = DDS_Subscriber_create_datareader( chatSubscriber, myTopic, DDS_DATAREADER_QOS_USE_TOPIC_QOS, NULL, DDS_STATUS_MASK_NONE);
// чтение данных в msgSeg
DDS_sequence_Chat_Msg *msgSeq = DDS_sequence_Chat_Msg__alloc();
DDS_SampleInfoSeq *infoSeq = DDS_SampleInfoSeq__alloc();
```

```
status = Chat_MsgDataReader_take(mbReader, msgSeq, infoSeq, DDS_LENGTH_UNLIMITED,
DDS_ANY_SAMPLE_STATE, DDS_ANY_VIEW_STATE, DDS_ALIVE_INSTANCE_STATE);
```

Структура взаимодействия основных сущностей

Структуру взаимодействия основных сущностей можно рассмотреть на рис. 1

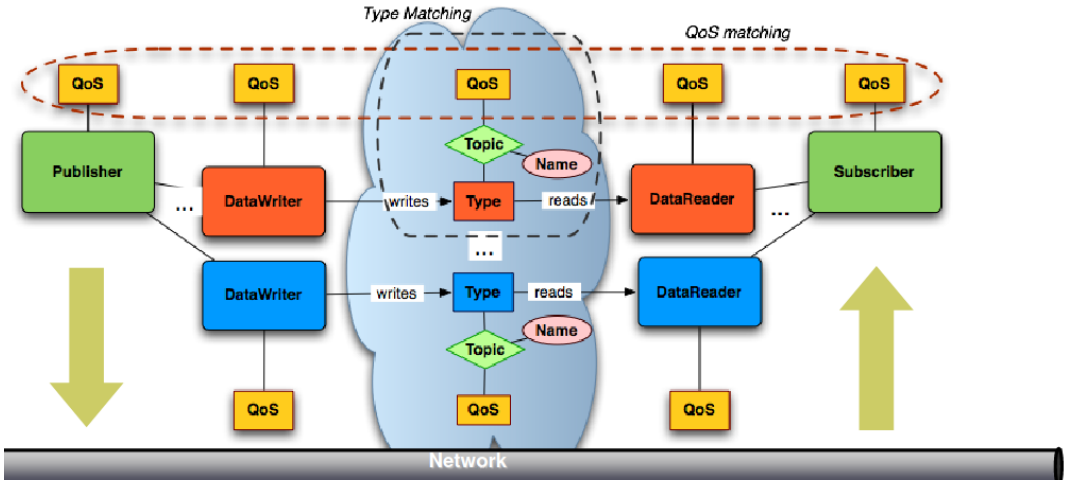


Рис. 1

Список источников

1. Инструкция по openSplice DDS [Электронный ресурс] URL: http://download.ist.adlinktech.com/docs/Vortex/pdfs/OpenSplice_DDSTutorial.pdf (Дата обращения: 02.05.2022)

2. Начальное руководство по openSplice [электронный ресурс] http://download.prismtech.com/docs/Vortex/pdfs/OpenSplice_GettingStartedGuide.pdf (Дата обращения: 02.05.2022)
3. Руководство по Vortex OpenSplice с примерами кода на C [Электронный ресурс]URL:https://github.com/ADLINK-IST/opensplice/blob/master/docs/pdf/OpenSplice_Tutorial_C.pdf (Дата обращения: 02.05.2022)