Распределенные информационные системы

Технология DDS – 2

Вопросы

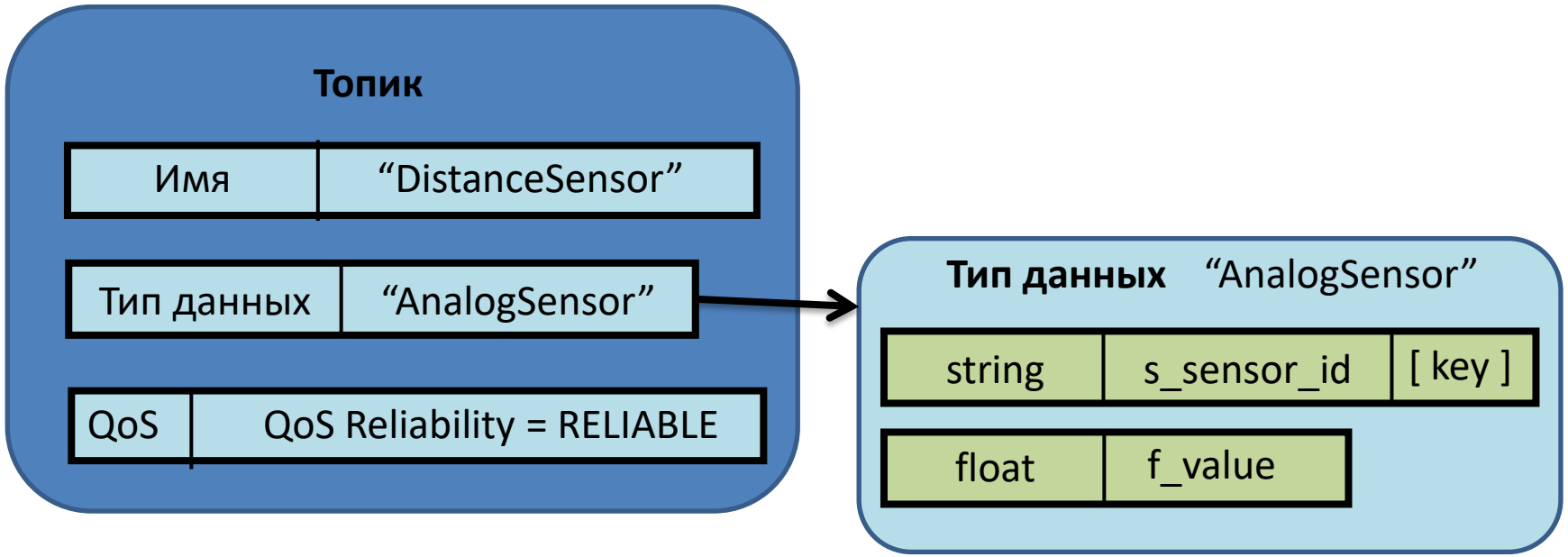
- Основные понятия:
  - Типы данных, ключи
  - Регистрация топиков
- Разграничение данных
- Получение и отправка данных

Типы данных, ключи

Регистрация топиков

ОСНОВНЫЕ ПОНЯТИЯ

Основные понятия



Типы данных – 1/6

- Тип данных, на который ссылается топик DDS, описывается структурой IDL, содержащей произвольное число полей. Типами данных поля могут быть:
  - Примитивные типы IDL. Например: octet, short, long, float, string (bound/unbound), и т.д.
  - Перечислимый тип (enumeration)
  - Объединение (union)
  - Последовательность (sequence – bounded or unbounded)
  - Массив
  - Структура (вложенная)

Типы данных – 2/6

```
struct HelloTopicType {
   string message;
};
```

```
struct Counter {
   long cID;
   long count;
};
```

```
struct PingType {
   long counter;
   string<32> vendor;
};
```

```
struct ShapeType {
   long x;
   long y;
   long shapesize;
   string color;
};
```

```
enum TemperatureScale {
   CELSIUS,
   FAHRENHEIT,
   KELVIN
};
struct TempSensorType {
   short id;
   float temp;
   float hum;
   TemperatureScale scale;
};
```

Типы данных: ключи – 3/6

- Каждый тип данных (для топика DDS) дополнительно описывается ключом (key-set) – набором полей
- На количество полей, входящих в ключ, ограничений не накладывается (например, набор полей может быть пуст)
- В ключ могут входить атрибуты верхнего уровня и атрибуты из вложенных структур

Типы данных: ключи – 4/6

```
struct HelloTopicType {
    string message;
};
```

```
struct PingType {
    long counter;
    string<32> vendor;
};
```

```
struct Counter {
    long cID;
    long count;
};
#pragma keylist Counter cID
```

```
struct ShapeType {
    long x;
    long y;
    long shapesize;
    string color;
};
#pragma keylist ShapeType color
```

```
enum TemperatureScale {
    CELSIUS,
    FAHRENHEIT,
    KELVIN
};
struct TempSensorType {
    short id;
    float temp;
    float hum;
    TemperatureScale scale;
};
#pragma keylist TempSensorType id
```

Типы данных: ключи – 5/6

- Ключи используются для идентификации «экземпляров»
- Если провести параллель с ООП, то:
  - Типы данных без ключа – синглетоны (экземпляр строго единственный)
  - Типы данных с ключом аналогичны «обычным» классам с экземплярами, идентифицируемыми значением ключевых полей
  - Можно считать, что новое значение ключа порождает новый «объект» в РИС DDS

Типы данных: ключи – 6/6

```
struct Counter {
   long cID;
   long count;
};
#pragma keylist Counter count
```

```
struct HelloTopicType {
   string message;
};
#pragma keylist HelloTopicType message
```

```
struct PingType {
   long counter;
   string<32> vendor;
};
#pragma keylist PingType vendor
```

```
struct PingType {
   long counter;
   string<32> vendor;
};
#pragma keylist PingType counter
```

Имена топиков и регистрация – 1/2

- Регистрация топика идемпотентна (sic!), т.е. может быть повторена без ошибки различными (или одним и тем же приложением)
- Попытка зарегистрировать топик, отличающийся только типом приведет к ошибке

Имена топиков и регистрация – 2/2

Приложение 1	Приложение 2

Успешная последовательность

```
dds::Topic<ShapeType> shape("Circle");
```

```
dds::Topic<ShapeType> shape("Circle");
```

Ошибочная последовательность

```
dds::Topic<ShapeType> shape("Circle");
```

```
dds::Topic<AntotherType> shape("Circle");
```

Успешная последовательность

```
dds::Topic<ShapeType> shape("Circle");
```

```
dds::Topic<ShapeType> shape("Cube");
```

Домены

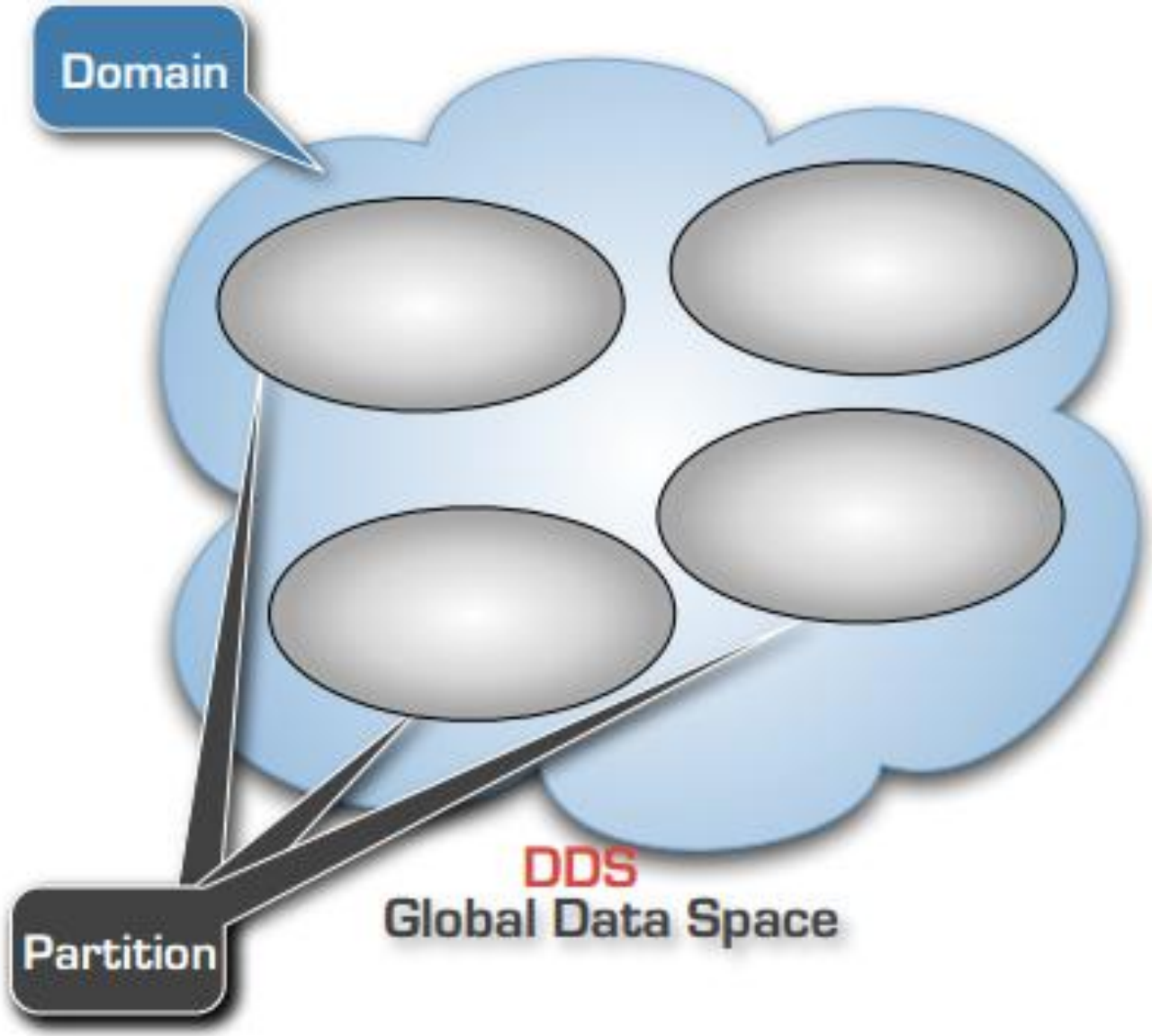
Области

РАЗГРАНИЧЕНИЕ ДАННЫХ

Домены и области – 1/2

- Домен
  - Под доменом понимается экземпляр глобального пространства данных (DDS Global Data Space)
  - Приложения DDS (и более низкие элементы) всегда относятся только к одному домену
- Область (partition)
  - Область – это механизм, позволяющий выделять подмножества взаимодействующих компонентов DDS (в том числе приложений)
  - Приложения DDS (и более низкие элементы) могут относиться сразу к нескольким областям

Домены и области – 2/2



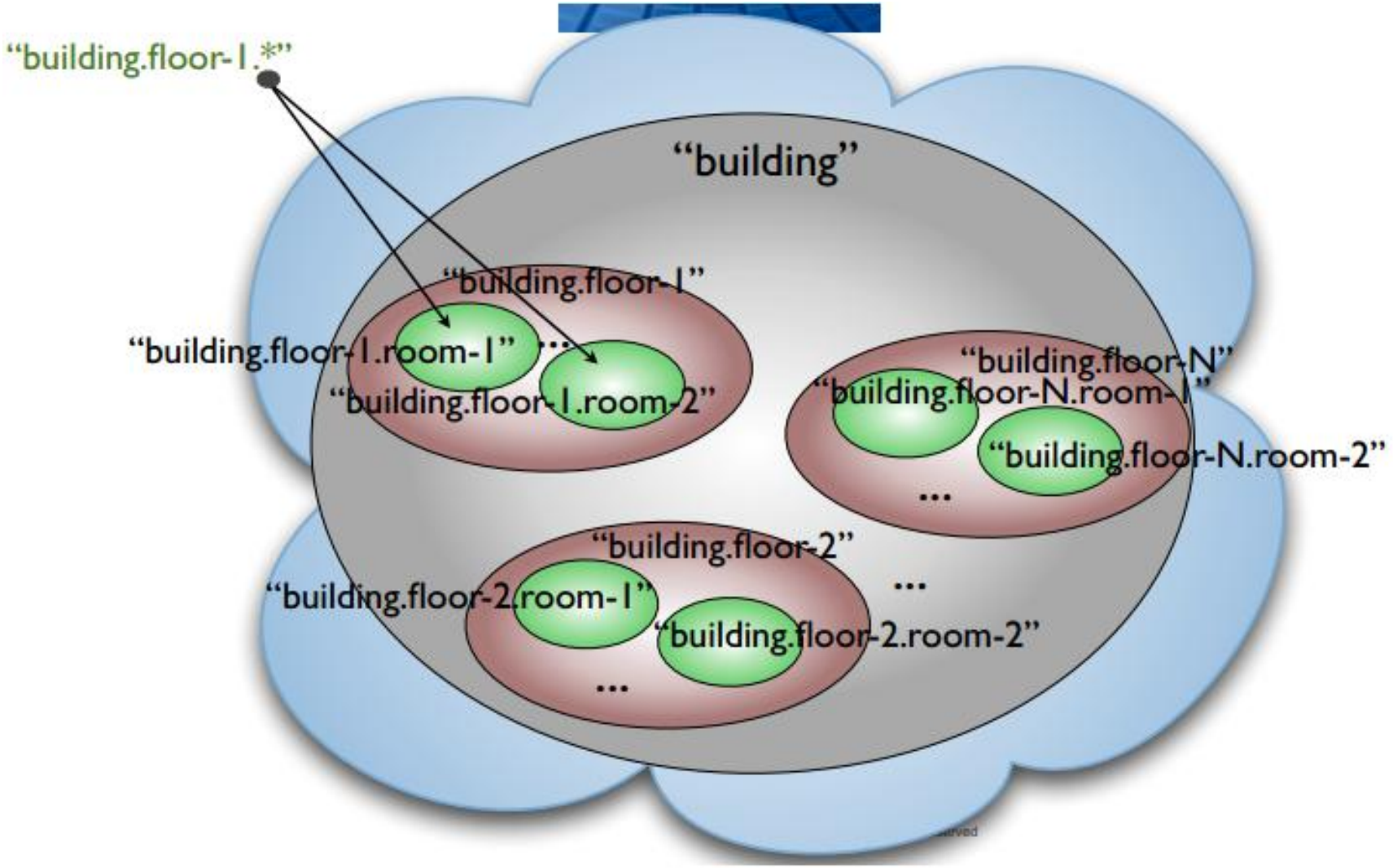
Области – 1/3

- Каждая область идентифицируется строкой. Например, «sensor-data», «log-data» и т.д.
- Доступ к области по чтению/записи обеспечивается через компоненты DDS Publisher/Subscriber
- Каждому такому компоненту (Publisher или Subscriber) можно передать список имен областей. Каждое имя может включать групповые символы (wildcards) или регулярное выражение, например «*-data»

Области – 2/3

- Хотя иерархичность областей не поддерживается в DDS напрямую, ее легко организовать при разработке РИС, используя «точечную» нотацию
- Например, для здания, в котором размещаются датчики температуры, можно использовать нотацию «здание.этаж.номер-комнаты»
  - building.floor-2.room-10
  - building.floor-3.room-15
- Тогда, для получения данных по этажу используется маска названия области «building.floor-2.*», а для всего здания маска «building.*»

Области – 3/3


"building.floor-1.*"
"building"
"building.floor-1"
"building.floor-1.room-1"
"building.floor-1.room-2"
"building.floor-N"
"building.floor-N.room-1"
"building.floor-N.room-2"
"building.floor-2"
"building.floor-2.room-1"
"building.floor-2.room-2"

Отправка данных

Чтение данных

ПОЛУЧЕНИЕ И ОТПРАВКА ДАННЫХ

Отправка данных – 1/2

- Отправка данных производится в два шага:
  - Создается компонент DataWriter с помощью нужного конструктора (выбор завит от требующейся функциональности – отличной от стандартной)
  - Потом данные передаются в произвольный момент с использованием этого компонента

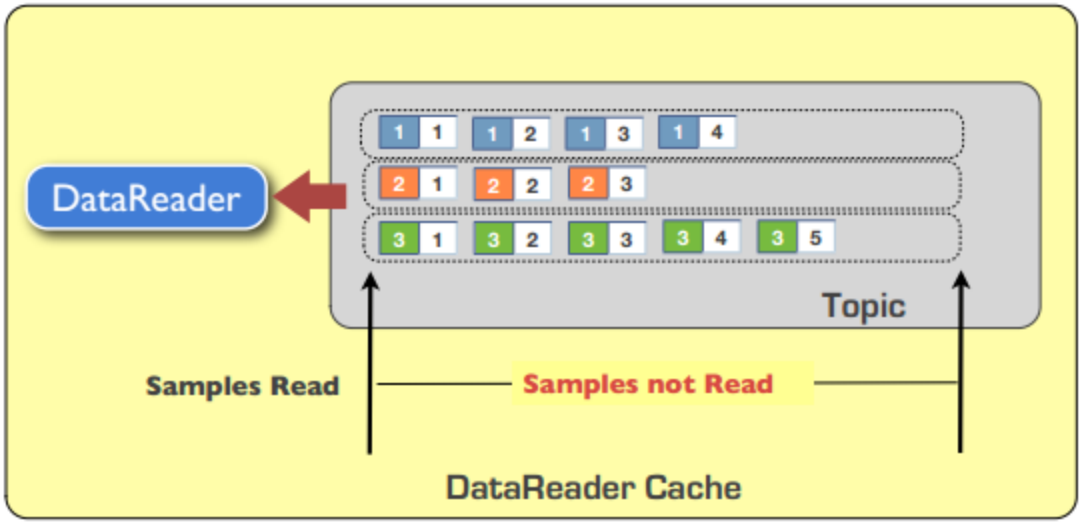
Отправка данных – 2/2

```
template <typename T>
class dds::pub::DataWriter : public dds::core:: Entity {
public:
 DataWriter();
 DataWriter(Topic<T> topic)
 DataWriter(Topic<T> topic, const DataWriterQos& qos)
 DataWriter(Topic<T> topic, const DataWriterQos& qos,
Publisher pub);
// ...
};
```

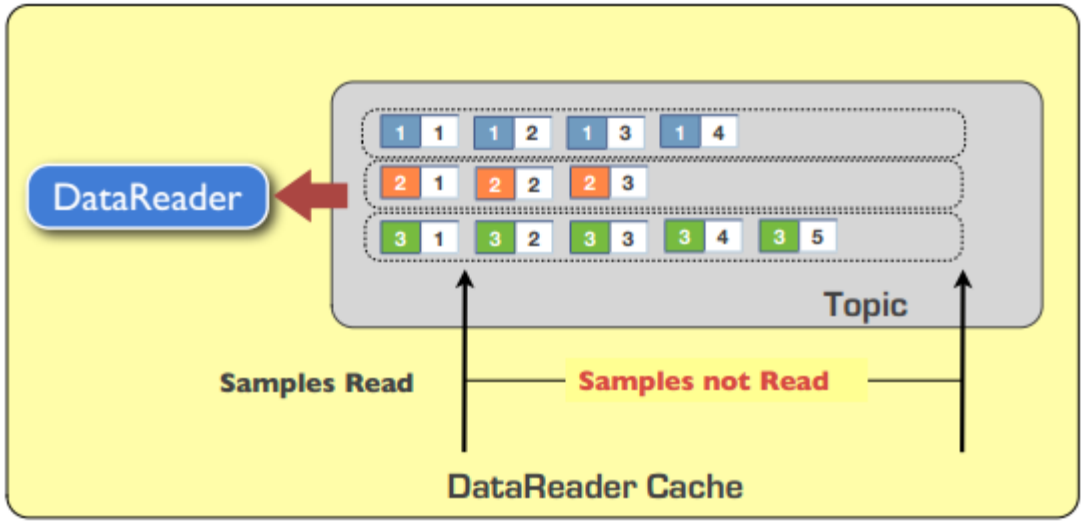
Получение данных – 1/7

- Операция чтения (read) «проходит» по всем доступным семплам (sample) экземпляра
- Семплы не удаляются из локального кеша DDS в результате операции чтения (read)
- Прочтенные могут быть прочтены из локального кеша DDS снова при указании специальных параметров операции чтения

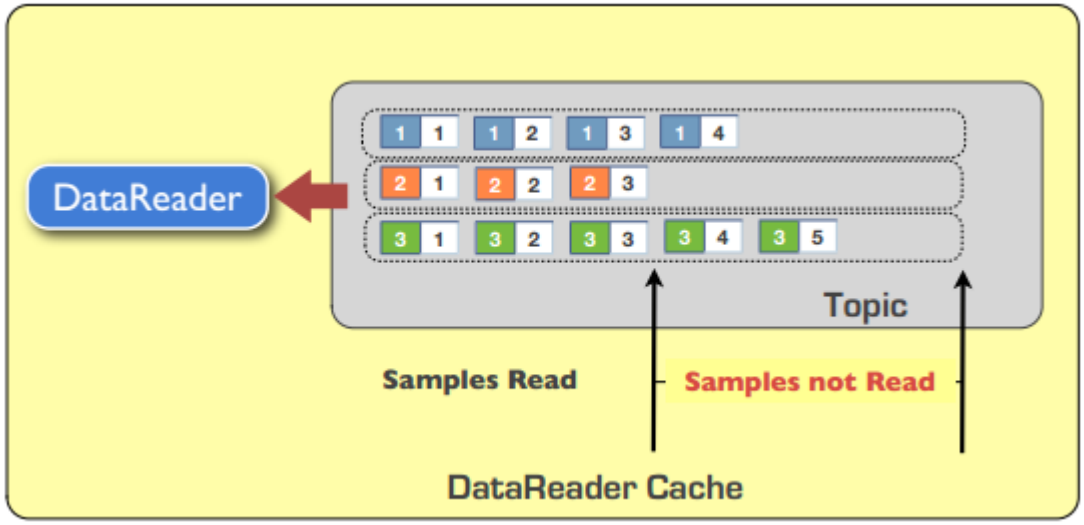
Получение данных – 2/7



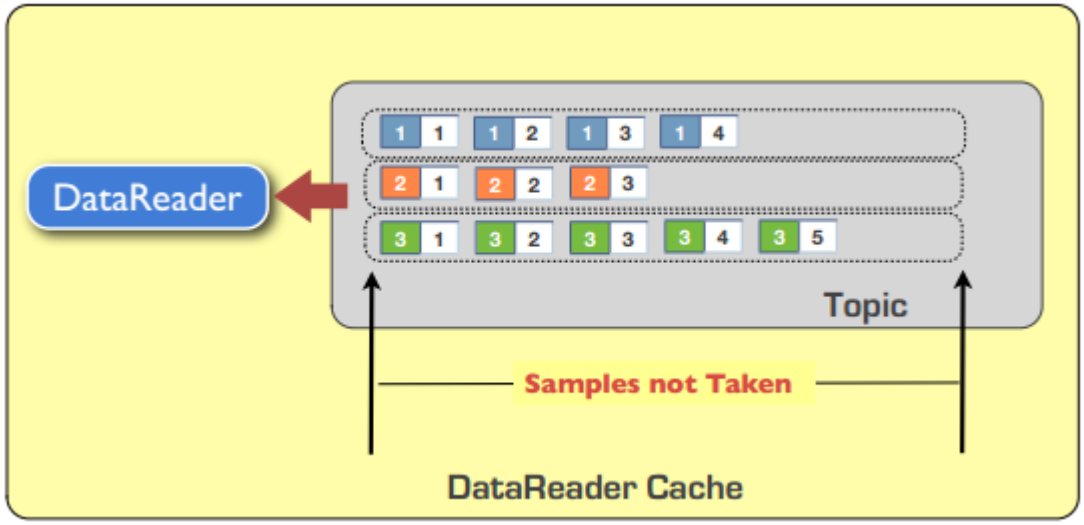
Получение данных – 3/7



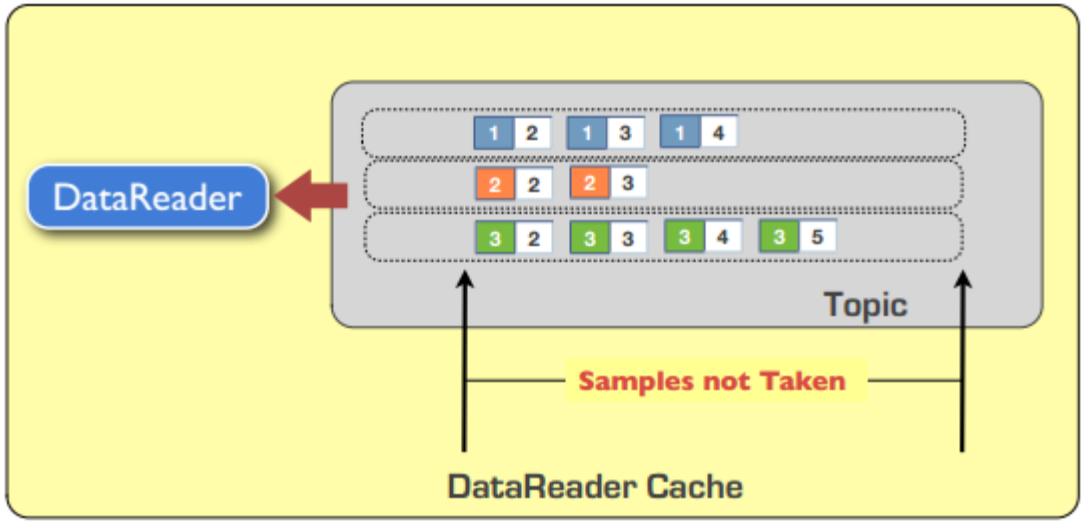
Получение данных – 4/7



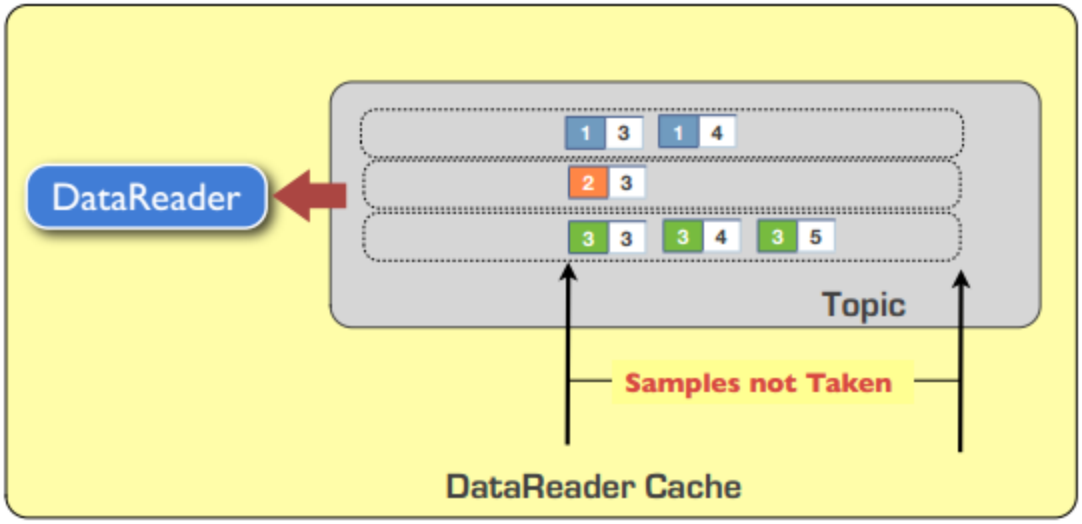
Получение данных – 5/7



Получение данных – 6/7



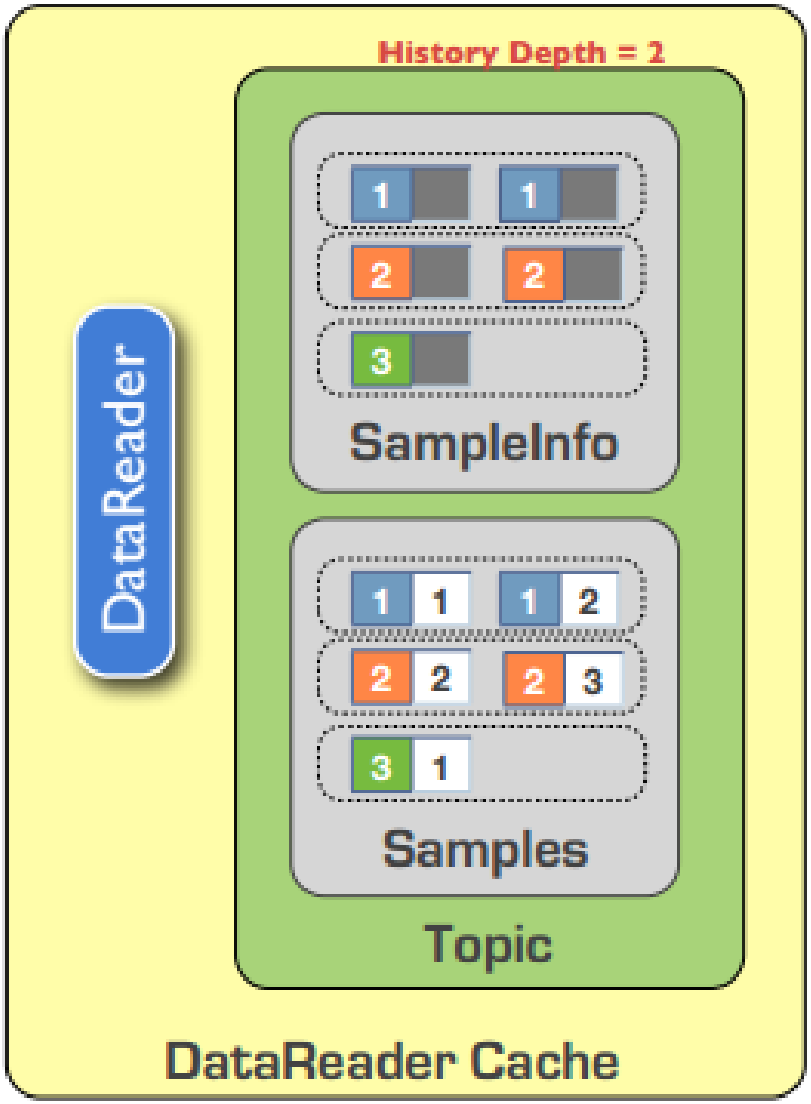
Получение данных – 7/7



Состояния при чтении – 1/2

- Кроме содержащих непосредственно данных семплов DataReaders предоставляют информацию о состоянии принятых данных и источниках данных (DataWriters)
- Состояние семпла: {READ | NOT_READ}. Определяет, был ли данный семпл уже хотя бы раз прочитан (read)
- Состояние экземпляра: {ALIVE | NOT_ALIVE_NO_WRITERS | NOT_ALIVE_DISPOSED}. Определяет:
  - Существует ли хоть один писатель для данного конкретного экземпляра
  - Нет ни одного писателя для экземпляра
  - Экземпляр был сброшен (disposed)
- Состояние view: {NEW | NOT_NEW}. Определяет, первый ли это семпл для нового (или повторно появившегося – re-born) экземпляра

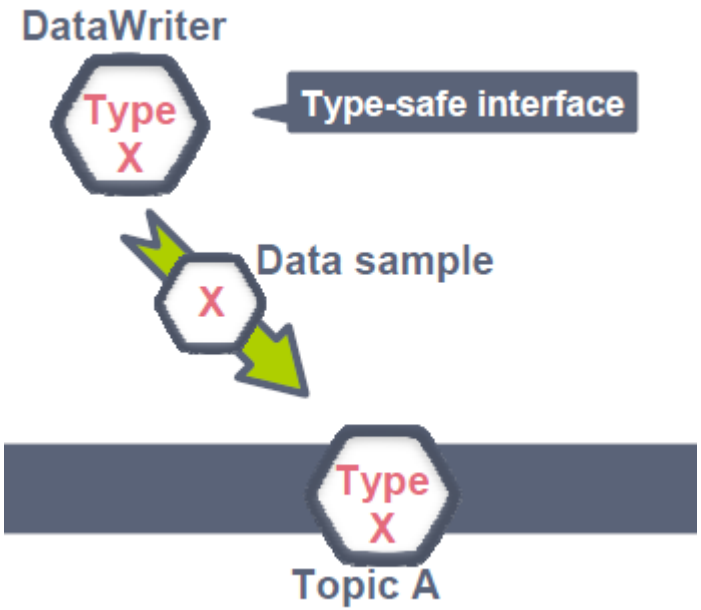
Состояния при чтении – 2/2



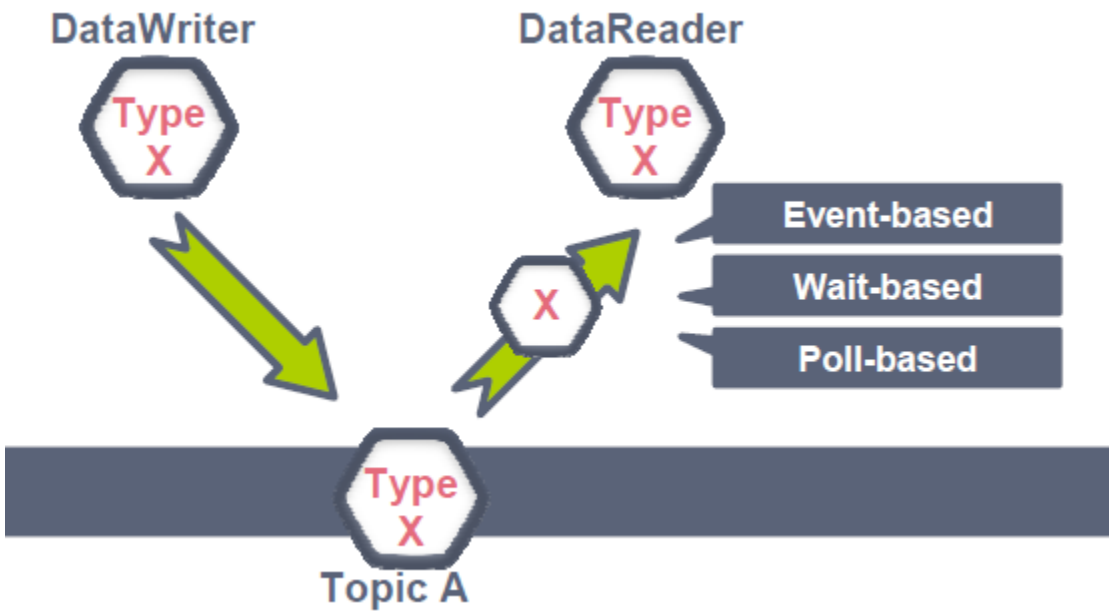
Получение данных от DDS – 1/3

- Доступны три способы взаимодействия (обмена данными между) DDS и приложения:
  - Опрос (polling)
    - Приложение (по своей инициативе, обычно, периодически) запрашивает DDS для получения новых данных или информирования о смене состояния. Интервал опроса может зависеть от приложения и/или от данных и их значений
  - Списки ожидания (WaitSets)
    - Приложение регистрирует в DDS списки ожидания и ждет (например, в режиме suspended) до тех пор, пока одно из переданных событий не произойдет
  - «Слушатели» (listeners)
    - Приложение регистрирует в DDS (в классах, где события могут произойти) специальные классы-слушатели, которые будут информированы при наступлении этих событий

Получение данных от DDS – 2/3



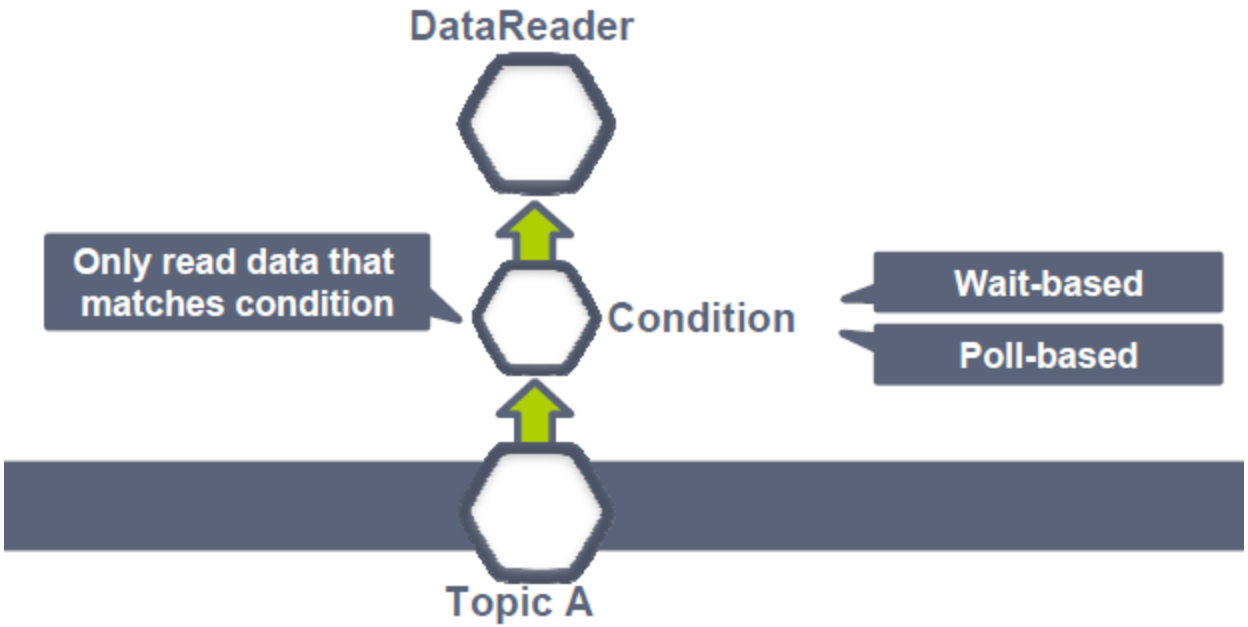
Получение данных от DDS – 3/3



Обработка полученных данных – 1/3

- Основанная на жизненном цикле (lifecycle):
  - Чтение данных только по «живым» экземплярам
  - Чтение данных только по новым экземплярам
  - Чтение только не прочитанных ранее семплов
- SQL-подобная фильтрация:
  - Фильтрация по значениям атрибутов
  - Агрегация экземпляров по ключу
  - Условия могут быть изменены во время работы

Обработка полученных данных – 2/3



Обработка полученных данных – 3/3