Карпова И.П. Хранение данных системой автономных устройств //
Информационные технологии, 2013, №5. – с. 29-34.

УДК 004.75

И.П. Карпова, к.т.н., доцент МИЭМ НИУ ВШЭ

Хранение данных системой автономных устройств

Аннотация

Рассматривается вопрос организации хранения данных в памяти системы автономных устройств, оснащённых датчиками и собирающими информацию о состоянии окружающей среды. Предлагается метод организации записи данных в память, который обеспечит рациональное использование памяти в условиях ограничений на ресурсы.

Ключевые слова: система сбора и обработки данных, хранение данных, кольцевой буфер.

Введение

В настоящее время активно развиваются беспроводные сети сбора и обработки данных. В качестве примеров таких сетей можно привести беспроводные сенсорные сети (БСС) [1] или системы автономных мобильных роботов [2]. В узлах сети располагаются устройства (мобильные или стационарные), оснащённые датчиками. Номенклатура датчиков зависит от той задачи, которая стоит перед сетью. Обычно такие сети предназначены для мониторинга состояния окружающей среды, в системах безопасности и проч. [3].

Показания датчиков сохраняются в памяти устройства для дальнейшей обработки и/или передачи данных на центральный компьютер. Центральный компьютер может запрашивать с узлов информацию о показаниях датчиков в определённый момент времени и в определённой точке пространства. В целом система должна уметь отвечать на вопросы, включающие пространственно-временные характеристики, например: "Что наблюдалось в тот или иной момент времени и окрестности определённой

точки пространства?" Таким образом, для получения целостной картины устройства должны иметь единую привязку по времени и координатам пространства и должны уметь определять своё местоположение (относительно маяков, меток или других ориентиров).

Работа сенсорной сети построена на передаче данных от одного узла другому, причём на передачу данных тратится значительно больше энергии, чем на сбор или обработку этих данных. В случае передачи собранных данных на центральный узел сразу после опроса датчиков устройство может не иметь собственной памяти для хранения данных. Но здесь возникают ограничения, связанные с энергопотреблением [1,3]. Для сенсорных сетей проблема энергосбережения стоит очень остро, т.к. сенсорные устройства работают автономно, а возможность замены элементов питания в них не предусмотрена. В системах мобильных роботов другая проблема: при перемещении робот может выйти из зоны приёма центрального компьютера, и не сможет передать данные в момент их получения с датчиков.

С другой стороны, далеко не все задачи, решаемые с помощью подобных систем, требуют немедленной передачи собранных данных. В качестве примера можно привести системы, центральный компьютер которых опрашивает устройства периодически (например, в целях мониторинга загрязнения окружающей среды) или при возникновении какого-то нештатного события (аварии на химическом производстве и т.п.). Более того, общая тенденция в развитии подобных систем такова: по возможности переложить обработку данных на сами узлы, т.е. передавать на центральный узел не "сырые" данные, а результаты их обработки. В этом случае инициатива о передаче данных вообще может возлагаться на устройство в узле. Устройство обрабатывает данные за некоторый период времени, определяет тенденцию развития ситуации или получает какую-то

статистику, после чего передаёт результаты на центральный узел. Это уменьшает трафик в сети и сокращает её энергопотребление.

Во всех рассмотренных ситуациях устройствам необходимо хранить собранные данные. Причём чем больше будет тот период, в течение которого данные будут храниться, тем выше будет ценность собранной информации: можно качественнее восстановить предшествующую событию ситуацию и точнее сделать прогноз на будущее.

Таким образом, возникает задача разработки такого способа управления собранными данными, который обеспечивал бы к ним доступ без потребности массовой передачи данных в центральные узлы [4].

Постановка задачи

Итак, пусть имеется несколько автономных устройств (мобильных или стационарных), оснащённых определённым количеством датчиков. С помощью этих датчиков они могут собирать информацию об окружающем мире, например, о температуре, освещённости, химическом составе воздуха и т.п. Условия работы системы таковы:

- Каждое устройство оснащёно произвольным набором датчиков.
- Все устройства работают асинхронно, но для системы в целом поддерживается единая служба времени.
- Время отсчитывается тактами, размер такта для всех устройств одинаков (например, 1 секунда).
- Устройство снимает показания с датчиков с периодичностью, которая определяется его программой и не зависит от других устройств.
- В каждый момент времени устройство может снимать показания с произвольного количества своих сенсоров.
- Данные необходимо сохранять в памяти устройства для дальнейшей обработки и/или передачи по заранее обусловленному адресу.
- Все показания снимаются с определенной погрешностью, для каждого датчика (показателя) величина этой погрешности своя. Она зависит от

качества того датчика, которым оснащено устройство, условий работы и проч.

- Нет никаких гарантий, что каждый датчик ответит на запрос: на датчике может произойти сбой, и тогда данные от него не поступят.
- Объём памяти устройства невелик; при исчерпании этого объёма запись должна производиться поверх самых старых по времени показаний.
- Важны не просто последние по времени значения данных, а их изменения. Если данные не изменились, то имеет смысл хранить период, в течение которого значение показателя не менялось. Это позволит в том же объёме памяти сохранить данные за более длительный промежуток времени.

Определим круг задач, которые стоят перед такой системой:

- Хранение данных в базе данных (БД) устройства. Показания датчиков привязаны ко времени и местоположению устройства.
- Обеспечение ответов на запросы о значениях показателей в окрестностях определённого момента времени и в окрестностях определённой точки пространства.

В данной статье будет рассмотрен вопрос организации хранения данных в такой системе в условиях ограниченного объёма памяти каждого устройства.

Организация хранения данных

Для начала требуется определить, в каком виде наиболее целесообразно хранить эти данные.

В нашем случае все данные можно разделить на две группы: справочные данные и показания датчиков. К первой группе относятся:

a. Настройки (интервал времени для определения временной окрестности; диапазон координат для определения окрестностей точки пространства и т.д.).

b. Перечень показателей, которые можно снять с датчиков (наименование показателя, погрешность измерения показателя и проч.).

Пространственно-временные настройки должны быть одинаковы для всех устройств, а перечень показателей и погрешности их измерений могут быть разным, т.к. зависят от конкретной номенклатуры датчиков. Интервал времени для определения временной окрестности относительно момента t_0 может быть несимметричным, например, $[t_0-1, t_0+5]$. А в качестве диапазона координат для определения окрестностей точки пространства имеет смысл взять радиус окрестности точки x_0, $O(x_0, r)$.

Что же касается хранения показаний датчиков, можно хранить данные в виде реляционной таблицы (или таблиц). Это хорошо известный механизм, применение которого позволило бы при передаче данных на центральную машину без проблем заливать эти данные в реляционную базу данных и использовать для обработки данных тот же язык SQL.

Но при ограничениях по объёму памяти использование для хранения данных реляционной модели является не самым удачным решением. Продемонстрируем это на конкретном примере. В реляционной модели возможны два варианта хранения данных о показания датчиков:

1) в виде кортежей: время плюс набор всех показателей;
2) в виде троек (время, номер показателя, значение показателя).

Рассмотрим две крайние ситуации:

1) в каждый момент времени происходит съём показателей со всех датчиков, и значения всех показателей меняются;
2) в каждый момент времени изменяется только один показатель или происходит опрос только одного датчика.

В первом случае оптимальным является хранение данных кортежами, т.к. при этом никакая информация не теряется, а время хранится только один раз на кортеж. Во втором случае целесообразно хранить данные тройками (время, номер, значение), чтобы не хранить

"пустоту". В реальности ни первой, ни второй ситуации в чистом виде не будет. В зависимости от условий, в которых будет функционировать устройство, и от задач, которые он будет решать, ситуация будет ближе либо к первому, либо ко второму варианту. Но и в том, и в другом случае память будет расходоваться неэкономно.

Приведём несложные расчёты для случая, когда значения всех показателей занимают одинаковое количество памяти (4 байта). 4 байта для хранения времени (количества тактов после начала работы) это свыше ста лет при размере такта 1 секунда, т.е. более чем достаточно. Пусть N – это количество датчиков на устройстве. При хранении данных кортежами на каждый кортеж уйдет (N+1)*4 байт. Обычно, количество датчиков на устройстве от 5 до 10. Таким образом, пропуск даже одного значения даёт потери примерно 10 ÷ 15%.

С другой стороны, если хранить данные тройками, то каждая из них будет занимать 9 байт: время (4 байта) + номер показателя (1 байт) + значение (4 байта). Такая схема хранения имеет смысл только в той ситуации, при которой в каждый момент времени собирается не более 45% от общего количества показателей:

$$9M < 4(N+1),$$
$$\frac{M}{N+1} < 0.44,$$

где M – это количество реально снятых показаний. При этом M может меняться динамически от 1 до N в зависимости от реальной ситуации. И если брать за основу реляционную модель (хранить данные в виде реляционных таблиц), то либо память будет расходоваться неэффективно, любо придётся усложнять программу и динамически менять способ хранения данных, записывая их в разные таблицы. Такой подход имеет и ещё один недостаток: усложнение реализации запросов к данным из-за того, что они хранятся в двух таблицах разной структуры.

Исходя из вышесказанного, предлагается другой вариант для организации хранения значений показателей: хранить данные кортежами, но включать в кортеж только реальные значения. Это можно организовать одним из двух способов:

1) Если значение каждого показателя имеет фиксированную длину, то эта длина хранится в таблице настроек. А каждому кортежу показаний датчиков приписывается в начало битовая строка признаков, в которой i-й бит соответствует i-му датчику:

   1 – есть значение,

   0 – нет значения.

2) Если значения показателей могут занимать разный объём памяти (например, фотографии разной чёткости или запись звука разной продолжительности), то размер значения хранится в самом кортеже непосредственно перед значением показателя.

Такой подход позволит хранить только реально полученные с датчиков показания.

Примечание: в реальной ситуации возможны и промежуточные варианты, когда часть показаний имеет фиксированную длину и хранится в начале кортежа (первый способ), а часть показаний – переменную длину (второй способ).

Важно также различать ситуации, когда показания не были сняты (по причине сбоя датчика, например) и когда показания не записывались в память, потому что не изменились. Для отражения этих фактов необходима ещё одна битовая строка, разряды которой будут интерпретироваться следующим образом:

1 – показания не сняты,

0 – показания не изменились.

Длина каждой из этих битовых строк при указанном диапазоне возможных значений N составит 2 байта. Таким образом, для показателей фиксированной длины каждый кортеж займёт (8+4N) байт. На рис. 1

приведён пример такой организации для N=6 и временной отметки t = 29. Первая битовая строка означает, что из шести показателей хранятся три (с номерами 1, 4, 6), при этом 2-3 показатели не сняты, а пятый не изменился.

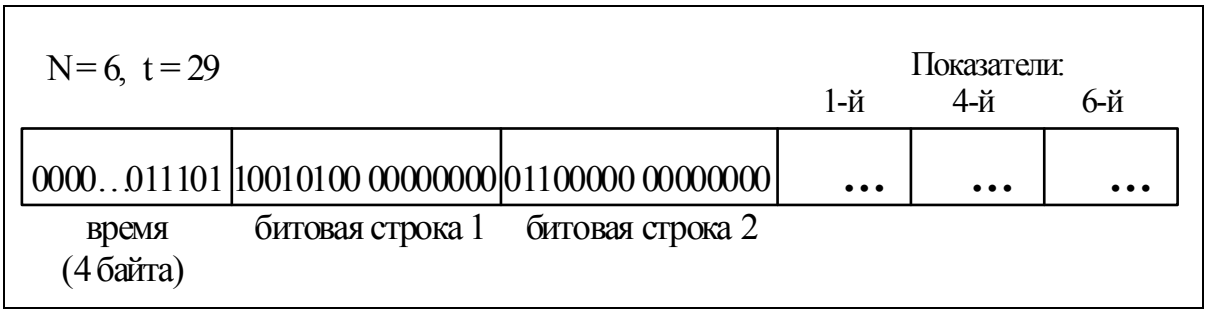

Рис. 1. Пример кортежа данных

При обработке данных во второй битовой строке будут анализироваться только те биты, для которых в первой битовой строке соответствующий бит установлен в 0.

Организация перезаписи данных

При поступлении данных (кортежей) они записываются в память до тех пор, пока её объём не исчерпан. При этом записываются только изменившиеся значения показателей. После исчерпания объёма памяти необходимо осуществлять запись новых кортежей поверх самых старых (устаревших, неактуальных) данных. Т.е. память должна быть организована в виде *кольцевого буфера*, который имеет два указателя: на первый и на последний кортеж, записанный в память.

Здесь возникают две сложности:

1) как определить, изменилось ли показание датчика по сравнению с предыдущим значением?
2) как определить, можно ли перезаписывать новый кортеж поверх старого, или в нём хранится неизменное значение показателя и его нельзя потерять?

Кэш. Для ответа на первый вопрос предлагается выделить специальную область памяти (*кэш*), в котором будут храниться последние значения

показаний датчиков (без привязки ко времени). После очередного опроса датчиков каждое полученное значение показателя сравнивается с тем, которое хранится в кэше. Если разность между новым значением показателя и тем, которое хранится в кэше, превышает погрешность, то это значение записывается в кэш и вносится в кортеж значений. Если же эта разность меньше погрешности для данного показателя, т.е. значение не изменилось, то оно не вносится в кортеж значений (причём i-е биты первой и второй битовых строк устанавливаются в 0). В кэше при этом также остаётся прежнее значение. Это необходимо для того, чтобы предотвратить "дрейф" значения показателя, т.к. отношение "примерно равно" не является транзитивным.

Теперь рассмотрим вопрос с актуальностью данных. Как определить, является ли перезаписываемое значение определённого показателя устаревшим или это неизменившееся значение? Сравнение с тем значением, которое хранится в кэше, ничего не даст. Перезаписываемое значение может быть равным тому, которое хранится в кэше, но это не значит, что оно не менялось за период между записью кортежа в память и получением того значения, которое хранится в кэше. Конечно, можно просмотреть всю память в поисках значения данного показателя с более поздней временной отметкой, и, если такого нет, не перезаписывать значение данного показателя. Но это потребует много времени, потому что кортежи в общем случае имеют разную длину, и их просмотр можно осуществить только последовательно.

Для решения этой задачи предлагается такой подход. Если в новом кортеже отсутствует значение определённого показателя, то можно считать, что это значение не изменяется и перезаписывать его нельзя. Но вместе с ним останется весь кортеж. Хранение таких кортежей – побочный эффект: будут храниться устаревшие показатели датчиков. Это серьёзный недостаток: эти данные уже неактуальны, но они занимают память. Потери

памяти в крайнем случае могут составлять до (N-1)-й записи, причём устаревшими в каждой записи будет (N-1) показание:

$$4(N-1)(N-1) = 4(N-1)^2$$

при условии, что каждый показатель занимает 4 байта и (N-1) показателей не меняет своего значения, но фиксация неизменного значения каждого показателя произошла в разные моменты времени.

Кроме того, периоды неизменности значений разных показателей в общем случае не совпадают друг с другом, и это может привести к искажению сведений о значениях изменяющихся во времени показателей: какие-то из них мы будем хранить (вместе с неизменными), а какие-то – удалять. Поэтому предлагается не хранить эти устаревшие значения, а "упаковывать" кортежи, оставляя только неизменные значения и время, когда это значение было получено (естественно, изменяя соответствующим образом битовые строки). Таких записей будет немного – не более (N–1). Они сохранят структуру обычного кортежа, но будут хранить только одно значение.

К сожалению, если оставить эти упакованные кортежи в основной памяти, то могут возникнуть другие проблемы. Упаковав кортеж с временной отметкой t_k и значением i-го показателя P[i], мы можем в следующий момент времени t_{k+n} получить кортеж, в котором i-й показатель будет присутствовать, и перезапишем кортеж с временной отметкой t_{k+1}. Таким образом, возникнет искажение данных: в соответствии с принятыми правилами такие данные будут означать, что в диапазоне $[t_k; t_{k+2}]$ значение P[i] не изменялось, что не соответствует действительности. Пример такого искажения приведен на рис. 2. Из него получается, что в период $t_k \div t_{k+2}$ значение P[2] оставалось неизменным (и было равно 6), а это не так: $P[2](t_{k+1}) = 5$, но это значение потеряно при записи кортежа (2, 3, 4) с временной отметкой t_n поверх кортежа с временной отметкой t_{k+1}.

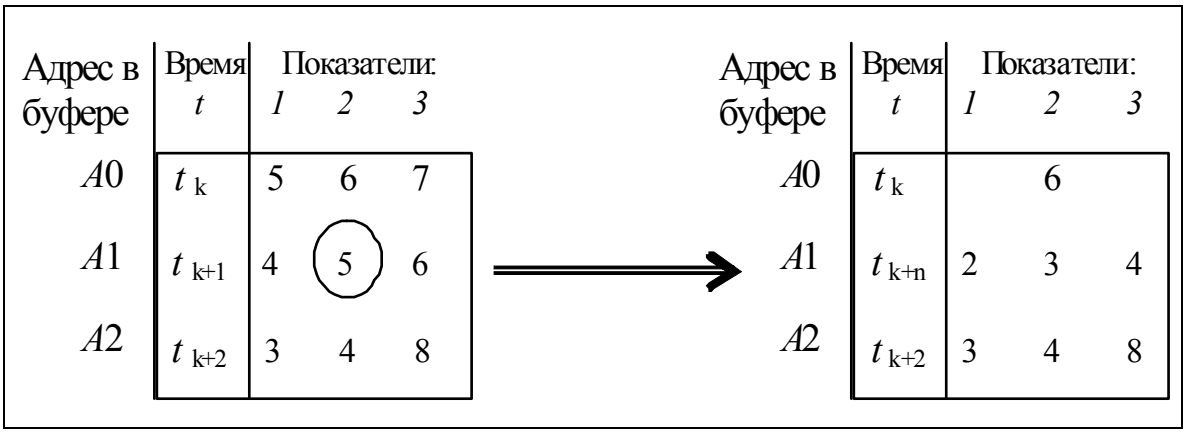

Рис. 2. Пример искажения данных

Буфер. Для того чтобы избежать таких искажений, эти упакованные кортежи необходимо хранить в отдельной области памяти. Назовем эту область ***буфером***. Правила работы с буфером будут следующими. Если в новом кортеже, который необходимо перезаписать поверх старого кортежа, отсутствует i-е значение, а в старом кортеже оно есть, то оно записывается в буфер на место i-го показателя вместе со своей временной отметкой. После чего новый кортеж записывается поверх старого. (Естественно, если в новом кортеже отсутствуют несколько показаний, все они из старого кортежа попадают в буфер.) Количество ячеек в буфере равно количеству показателей.

Начинать запись в буфер значения i-го показателя надо только тогда, когда значения этого показателя перестали появляться в новых данных. Если при пропуске значений помещать i-й показатель в буфер, а при появлении новых значений – очищать буфер, то может возникнуть ситуация, когда будет доступно только одно значение. Это может произойти, если в цикле перезаписи это значение появляется только один раз. Естественно, это недопустимо.

С другой стороны, если при наличии значения P[i] в буфере с появлением новых значений i-го показателя и записью их в память не очищать буфер от старых значений, то могут возникнуть искажения в данных, аналогичные рассмотренным ранее.

Для решения этих проблем можно помещать все перезаписываемые значения в буфер, и тогда не будет ни искажений, ни больших пропусков в данных. Но при этом ухудшается производительность за счёт двойной записи (в основную память и в буфер). Пожертвуем производительностью. Дело в том, что сбор данных происходит периодически и занимает больше времени, чем запись в память. Поэтому можно позволить в данной ситуации двойную запись: это не замедлит работу системы в целом.

Примечание: для других условий функционирования системы сбора данных или в случае использования устройства памяти с ограниченным количеством циклов перезаписи этот вариант реализации не подойдёт и может потребовать модификации.

Технические детали

При перезаписи данных возникает ещё одна небольшая чисто техническая задача: обеспечение корректности повторного использования памяти. Хранимые кортежи имеют переменную длину, поэтому при перезаписи нового кортежа поверх старых данных могут возникнуть разные ситуации:

1. Новый кортеж занимает столько же памяти, сколько старый, или меньше. Тогда никаких проблем не возникает: он просто записывается в память вслед за предыдущим и изменяется значение адреса первого кортежа *R1*.
2. Новый кортеж имеет длину, превышающую длину старого кортежа (в общем случае – превышающего суммарную длину k первых кортежей, подлежащих перезаписи). Тогда перед записью нового кортежа в память необходимо обработать все подлежащие перезаписи кортежи на предмет извлечения из них значений, подлежащих записи в буфер.

Окончательно последовательность записи данных в память будет выглядеть следующим образом:

1. Инициализировать переменные:
   – N – число датчиков;
   – NMAX – адрес конца области хранения данных;
   – B[N] – буфер, B[i]:=null, i=1,…,N;
   – K[N] – кэш, K[i]:=null, i=1,…,N;
   – E[N] – вектор погрешностей измерения показателей, i=1,…,N.
2. Установить указатели P1 на первый и PN на последний кортежи на начало области хранения данных (P1:=0, PN:=0).
3. Снять показания датчиков D[i] и сформировать текущий кортеж *C*:
   Если K[i]:=null,
   то установить K[i]:= D[i] и добавить D[i] в кортеж,
   иначе если | K[i] – E[i] | > D[i],
      то установить K[i]:= D[i] и добавить D[i] в кортеж;
      к.е.
   к.е.
   Вычислить LEN – длину кортежа *C*.
4. Если PN>P1 и PN+LEN <= NMAX,
   то записать кортеж *C* в память по адресу PN,
      установить PN:=PN+LEN;
   иначе
      Если PN+LEN>NMAX,
      то PN:=P1:
      к.е.
      Считать кортеж *R* по адресу P1.
      Вычислить LEN1 – длину кортежа *R* по адресу P1.
      Цикл пока PN+LEN>P1
         Цикл по i=1 до N
            Если кортеж *R* содержит показания i-го датчика,
            то если B[i]<>null или
                  кортеж *C* не содержит показания i-го датчика,
               то B[i]:=R[i];
               к.е
            к.е.
         к.ц.
         P1:=P1+LEN1;
         Считать кортеж *R* по адресу P1.
         Вычислить LEN1 – длину кортежа *R* по адресу P1.
      к.ц.
      Записать кортеж *C* в память по адресу PN.
      Установить PN:=PN+LEN.
   к.е.
5. Вернуться к п.3.

Заключение

В статье была рассмотрена задача организации хранения данных в памяти системы автономных устройств, оснащённых датчиками и собирающими информацию о состоянии окружающей среды. Был предложен метод организации памяти в виде кольцевого буфера и двух вспомогательных структур, который позволит обеспечить экономное хранение данных в узлах сети сбора информации в условиях жёстких ограничений на ресурсы.

К сожалению, невозможно получить однозначную оценку времени и количества энергии, которые система будет тратить на запись данных в память. В зависимости от того, когда начинают возникать пропуски в данных – в начале цикла перезаписи или в конце, – система будет тратить разное время и разное количество энергии на размещение данных в памяти. Поэтому возможны либо средние оценки, либо оценки в диапазоне.

Список литературы

1. Восков Л.С., Курпатов Р.О. Энергоэффективный комбинированный метод локализации в беспроводных сенсорных сетях. – Датчики и системы, 2011, № 4. – с. 42-45.
2. Карпов В.Э. Коллективное поведение роботов. Желаемое и действительное. // Современная мехатроника. Сборник научных трудов Всероссийской научной школы (г. Орехово-Зуево, 22-23 сентября 2011). – Орехово-Зуево, 2011. – 132 с. – с. 35-51.
3. Сергиевский М. Беспроводные сенсорные сети // КомпьютерПресс, 2007, №8. - http://www.compress.ru/article.aspx?id=17950
4. Кузнецов С.Д., Гринев М.Н. Ландшафт области управления данными: аналитический обзор. – http://citforum.ru/database/data_management_overview/5.shtml

I.P. Karpova

Data storage system of self-contained units

Introduce

The question of the organization of data storage in memory of system of the self-contained units equipped with sensors and collecting information on a state of environment is considered. The method of the organization of data recording in memory which will provide rational use of memory in the conditions of restrictions on resources is offered.

Keywords: collecting and data processing system, data storage, ring buffer.