УДК 004.42

Богдан А.П.
студент магистратуры
2 курс, факультет информатики
Самарский национальный исследовательский университет имени академика С.П. Королева
Россия, г. Самара

Bogdan AP
graduate student
2 year, Faculty of Information Technology
Samara National Research University
Russia, Samara

АЛГОРИТМЫ ПАРАЛЛЕЛЬНОГО МОДЕЛИРОВАНИЯ СЕТЕВЫХ ГРАФОВ.
ALGORITHMS FOR PARALLEL MODELING OF NETWORK GRAPHS.

Аннотация:

Статья посвящена актуальной на сегодняшний день задаче моделирования компьютерных сетей. Рассматриваются особенности реализации алгоритма параллельного моделирования.

Ключевые слова: *дискретно-событийное моделирование, компьютерные сети, модель Барабаши-Альберт, параллельное моделирование, разбиение графов.*

Abstract:

The article is devoted to the current problem of modeling computer networks. The features of the implementation of the parallel modeling algorithm are considered.

***Keywords**: discrete-event modeling, computer networks, Barabashi-Albert model, parallel modeling, graph splitting.*

На сегодняшний день существует несколько моделей безмасштабных сетей. Они отличаются друг от друга похожестью на реальные сети, а также входными параметрами.

В 2002 году Л. Барабаши и Р. Альберт была предложена простая модель [1] безмасштабной сети, основанная на двух идеях:

- постоянный рост сети;

- принцип предпочтительного присоединения.

Постоянный рост сети означал, что в начале сеть состоит из произвольного числа элементов, соединённый друг с другом произвольным образом, а затем через определённые моменты времени в неё добавляются новые узлы. Принцип предпочтительного присоединения – принцип, согласно которому каждый новый узел присоединяется к каждому уже существующему с определённой вероятностью.

Ряд работ [2], [3], [4] посвящён представлению графовых сетевых структур в памяти ЭВМ, а также моделированию сетей.

Одна из проблем заключается в том, что используемый фреймворк INET Framework не готов к работе в параллельном режиме. Для решения этой проблемы была использована его модификация [5]. Однако она основана на старой версии фреймворка, которая не совместима с текущей версией OMNeT++, поэтому в работе использовался OMNeT++ версии 4.5.

Также, в этой модификации неправильно работал механизм автоматического конфигурирования IP адресов в сети. В связи с этим, был разработан простой алгоритм генерации IP адресов.

Другая проблема заключается в том, что используемый по умолчанию алгоритм синхронизации cNullMessageProtocol существенно замедляет моделирование сети. В связи с этим были изучены дополнительные возможности [6] OMNeT++ и было принято решение о разработке собственного алгоритма синхронизации, основанного на cNoSynchronization.

Для создания собственного алгоритма синхронизации необходимо создать класс, унаследованный от cParsimProtocolBase.

Синхронизация по времени реализована следующим образом: пусть есть множество вычислительных узлов. У каждого узла есть своя очередь моделируемых событий. В условиях поставленной задачи сбалансировать нагрузку на вычислительные узлы не представляется возможным. Отсюда возникает ситуация, когда некоторые из вычислительных узлов обгоняют другие.

```
cMessage *msg = sim->msgQueue.peekFirst();
if (comm->getProcId() == 0)
{
    if (deltaT++ % 1000 == 0)
    {
        cCommBuffer *buffer = comm->createCommBuffer();
        simtime_t time = msg->getArrivalTime();
        buffer->pack(time);
        comm->broadcast(buffer, TAG_TIME);
        comm->recycleCommBuffer(buffer);
    }
    return msg;
}
else
{
    if (msg->getArrivalTime() < this->syncTime || !hasEvent[0])
        return msg;
}
```

Рисунок 1 – Алгоритм синхронизации по времени

Для недопущения этого был добавлен механизм синхронизации. Согласно нему, первый узел (на нём должна быть самая большая нагрузка) объявляется главным. Раз в определённое количество времени Δt он отправляет всем остальным узлам значение своих часов симуляции. Остальные узлы, получив такое сообщение, сохраняют полученное время и, при обработке событий, сравнивают время на своих часах с этим. В случае, если собственное время оказывается больше, симуляция на данном узле приостанавливается до тех пор, пока главный узел не отправит новое время. Код, соответствующий данному алгоритму, приведён на рисунке 1.

Синхронизация по нагрузке обусловлена следующим фактором. Так

как синхронизация по времени основывается только на главном узле, неизбежно возникает ситуация, когда он достигает конца симуляции (закончились события). В этом случае, он отправляет всем остальным сообщение об окончании симуляции. Остальные узлы, получая такое известие, так же вынуждены завершить симуляцию. Однако на них симуляция ещё не закончилась. Такое поведение приводит к ухудшению качества моделирования.

Решением этой проблемы стал механизм синхронизации нагрузки. Он реализован следующим образом: узел, у которого закончились события, смотрит на состояние остальных узлов. В случае, если у них тоже закончились события, он отправляет им сообщение о завершении симуляции. В противном случае, он отправляет остальным узлам сообщение о том, что у него закончились события. Код соответствующего алгоритма приведён на рисунке 2. Другие узлы, получив такое сообщение, делают соответствующую пометку у себя. В случае, если узел, у которого ранее закончились события, получает новые события, то он отправляет соответствующее сообщение остальным узлам.

```
if (sim->msgQueue.isEmpty())
{
    ev.printf("no local events, waiting for something to arrive from other partitions\n");
    if (hasEvent[comm->getProcId()])
    {
        hasEvent[comm->getProcId()] = false;
        sendHasEvent(false);
        int i = 0;
        while (i < comm->getNumPartitions() && !hasEvent[i])
            i++;
        if (i == comm->getNumPartitions())
            throw cTerminationException(eENDEDOK);
    }
    if (!receiveBlocking())
        return NULL;
}
```

Рисунок 2 – Алгоритм синхронизации нагрузки

Таким образом, был реализован алгоритм, позволяющий моделировать компьютерные сети с использованием параллельных ветвей и инструкций.

Использованные источники:

1. Albert, R. Statistical mechanics of complex networks [Text] / R. Albert, A.L. Barabási // Reviews of modern physics. – 2002. – Vol. 74. – No. 1. – P. 47.
2. Стуликова К.А., Полукаров Д.Ю. Проблемы отображения автономных систем с помощью графов //Известия Самарского научного центра Российской академии наук. – 2014. – Т. 16. – №. 4-2.
3. Капустин И.В., Полукаров Д.Ю. Реализация графовых структур данных с помощью библиотек JAVASCRIPT //IT & Transport/ИТ & Транспорт: сб. науч. статей – Самара, 2016. – С. 81-88.
4. Никулин С. А., Полукаров Д. Ю. НЕКОТОРЫЕ ОСОБЕННОСТИ МОДЕЛИРОВАНИЯ СЕТЕЙ С ЯЧЕИСТОЙ ТОПОЛОГИЕЙ //Перспективные информационные технологии (ПИТ 2019). – 2019. – С. 79-81.
5. Stoffers, M. Enabling distributed simulation of OMNeT++ INET models [Text] / M. Stoffers // arXiv preprint arXiv:1409.0994. – 2014
6. OMNeT++ - Simulation Manual [Электронный ресурс]. – URL: https://doc.omnetpp.org/omnetpp/manual/ (дата обращения: 10.05.2021)