Список використаних джерел:

1. O. Security. Kali linux. https://www.kali.org
2. G. Lyon. Nmap. https://nmap.org/
3. L. Gordon. Nping. https://nmap.org/nping/

Брицун А.В.

студент,

Национальный технический университет Украины

«Киевский политехнический институт имени Игоря Сикорского»

ГРАДИЕНТНАЯ ВРЕМЕННАЯ СИНХРОНИЗАЦИЯ В БЕСПРОВОДНЫХ СЕНСОРНЫХ СЕТЯХ

Беспроводная сенсорная сеть (БСС) – многообещающий новый инструмент для наблюдения природных явлений в высоком разрешении. Точное время – очень важный компонент для правильного и корректного функционирования БСС, без него (а также без местоположения) обнаруженные данные могут терять актуальность. Хотя можно представить случаи применения БСС, в которых аспекты «когда и где» воспринимаемых данных не вызывают большой озабоченности, большинство приложений предпочитают помечать измеренные данные меткой времени. Такая временная метка будет иметь смысл только в том случае, если узлы в сети беспроводных датчиков будут иметь достаточное временное согласование [5]. Для управления доступом к среде, используя TDMA, нужна точная информация о времени, для того, чтобы сеансы передачи не смешивались [1]. Также точное время помогает сохранять энергию, сокращая необходимые защитные времена пробуждения.

Хотя каждый сенсорный узел оснащен аппаратными часами, эти аппаратные часы обычно не могут использоваться напрямую, поскольку страдают от серьезного дрейфа. Независимо от того, насколько хорошо они будут откалиброваны при развертывании, часы, в конечном счете, проявят большой перекос. Чтобы обеспечить точное время, узлы должны время от времени обмениваться сообщениями, постоянно корректируя значения своих часов.

В данной статье рассматривается Gradient Time Synchronization Protocol (GTSP), который предназначен для обеспечения точной синхронизации синхронизирующих импульсов между соседями. GTSP работает полностью децентрализованным способом: каждый узел периодически широковещательно передает свою информацию о времени. Сообщения синхронизации, полученные от прямых соседей, используются, чтобы калибровать логические часы. Алгоритм не требует ни древовидной топологии, ни ссылочного узла, который делает его устойчивым против отказов узла и ссылки.

Современные современные многочастотные тактовые синхронизирующие протоколы, такие как FTSP [2], предназначены для оптимизации глобального перекоса. Однако существуют возможности для улучшения локального

перекоса. Это не удивительно, так как FTSP и аналогичные протоколы работают со связующим деревом, синхронизируя узлы в дереве со своими родителями и, в конечном счете, с корнем дерева. [3] Соседние узлы, которые не являются тесно связанными в дереве не будут синхронизированы хорошо, потому что ошибки распространяются вниз по разным путям дерева, см. рис. 1. Каждый прыжок будет представлять собой неизбежную случайную ошибку δ. Поскольку случайные распределенные ошибки суммируются в соответствии с квадратной корневой функцией на каждом скачке, ожидаемая ошибка между головкой и хвостом цепи из k узлов имеет порядок $\delta\sqrt{k}$.

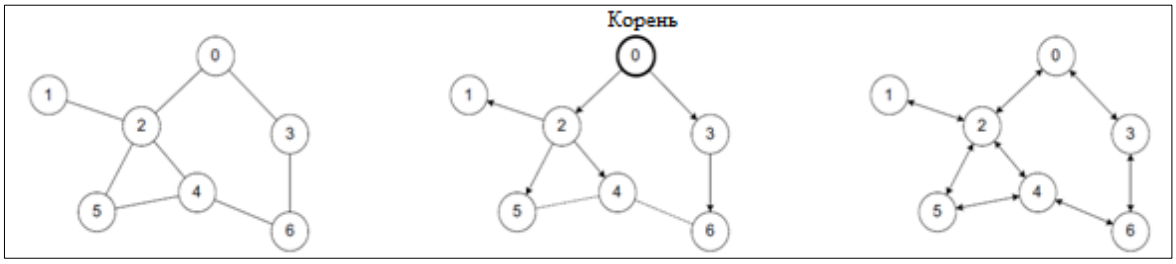

Рис. 1.

На рисунке 1 слева видна типичная сеть датчиков; ребра между узлами датчиков показывают двунаправленную линию связи. Центральная фигура представляет собой протокол синхронизации с деревом с узлом 0 в качестве опорного тактового генератора (корень), где каждый узел в дереве синхронизируется с его родительским; в этом примере ожидается, что узлы 4 и 6 синхронизируются субоптимально, даже если они являются непосредственными соседями, потому что они являются частью разных поддеревьев. Наконец, справа изображено идею градиентного протокола синхронизации: каждый узел синхронизируется со всеми своими соседями в графе связи. Корневой узел не требуется.

Предположим, что сеть состоит из нескольких узлов, оборудованных аппаратными часами, подверженными дрейфу часов. Кроме того, узлы могут преобразовывать текущее аппаратное чтение часов в логическое значение часов и наоборот. Каждый сенсорный узел i снабжен аппаратными часами $H_i(\cdot)$. Значение времени в момент времени t определяется как

$$H_i(t) = \int_{t0}^{t} h_i(\tau)d\tau + \Phi_i(t_0),$$

где $h_i(\tau)$ – аппаратная тактовая частота в момент времени τ и $\Phi_i(t_0)$ – смещение аппаратных часов в момент t_0.

Предполагается, что аппаратные часы имеют ограниченный дрейф, т. е. существует постоянная 0 <ρ <1 такая, что

$$1 - \rho \le h\,(t) \le 1 + \rho$$

для всех времен t. Это означает, что аппаратные часы никогда не останавливаются и всегда прогрессируют, по крайней мере, с частотой 1 – ρ. Это – разумное предположение, так как общие узлы датчика оборудованы внешними кварцевыми генераторами, которые используются в качестве генератора тактовых импульсов для встречного регистра микроконтроллера. На

этих генераторах проявляется дрейф, который постепенно изменяется только в зависимости от условий окружающей среды, таких как температура окружающей среды или напряжение батареи, а также от старения осциллятора.

Логическое значение часов $L_i(\cdot)$ вычисляется как функция текущего аппаратного тактового сигнала. Значение логической синхронизации L_i(t) представляет собой синхронизированное время узла i. Рассчитывается следующим образом:

$$L_i(t) = \int_{t0}^{t} h_i(\tau) l_i(\tau) d\tau + \theta_i(t_0),$$

где $l_i(\tau)$ – относительная логическая тактовая частота, а $\theta_i(t_0)$ – смещение тактовой частоты между аппаратным синхронизирующим сигналом и логическим тактовым сигналом в опорное время t_0. Логические часы поддерживаются как программные функции и рассчитываются только по запросу, основанному на данном показании аппаратных часов.

Основная идея алгоритма распределенной синхронизации часов – обеспечить точную синхронизацию часов между прямыми соседями, в то время как каждый узел может быть более свободно синхронизирован с узлами, более удаленными.

Абсолютная логическая тактовая скорость xi (t) узла i в момент времени t определяется следующим образом:

$$x_i(t) = h_i(\tau) \cdot l_i(\tau)$$

Каждый узел i периодически передает широковещательный маячок синхронизации, содержащий его текущее логическое время Li (t) и относительную логическую частоту синхронизации li (t). Получив маяки от всех соседних узлов в течение периода синхронизации, узел i использует эту информацию для обновления своей абсолютной логической тактовой частоты следующим образом:

$$x_i(t_{k+1}) = \frac{(\sum_{j \in N_1} x_j(t_k)) + x_i(t_k)}{|N_i|+1} \quad (1)$$

где Ni – множество соседей узла i.

Важно отметить, что на практике узел i не может сам настроить xi, так как у него нет возможности измерить собственную тактовую частоту аппаратного обеспечения hi. Вместо этого он может только обновить свою относительную логическую частоту синхронизации $l_i = \frac{x_i}{h_i}$ следующим образом:

$$l_i(t_{k+1}) = \frac{\left(\sum_{j \in N_1} \frac{x_j(t_k)}{h_i(t_k)}\right) + l_i(t_k)}{|N_i|+1} \quad (2)$$

Помимо того, что все узлы согласовали скорость передачи логических часов, также необходимо синхронизировать фактические значения часов. Опять же, узлы должны согласовать общее значение тактовой частоты, которое может быть получено путем вычисления среднего значения тактовых импульсов, как и для компенсации дрейфа. Узел i обновляет свое смещение логического такта θi следующим образом:

$$\theta_i(t_{k+1}) = \theta_i(t_k) + \frac{\sum_{j \in N_1} L_j(t_k) - L_i(t_k)}{|N_i|+1} \quad (3)$$

Однако использование среднего значения всех соседей в качестве нового значения синхронизации является проблематичным, если смещения являются большими. Во время запуска узла аппаратный тактовый регистр инициализируется нулем, что, возможно, приводит к огромному смещению узлов, которые уже синхронизированы с сетью. Такое огромное смещение заставит все остальные узлы повернуть свои часы, что нарушает принцип причинности. Вместо этого, если узел узнает, что часы соседа находятся дальше, чем определенное пороговое значение, он переходит на значение часов соседей [4]. Используя этот механизм начальной загрузки, узел, присоединяющийся к сети, быстро синхронизируется с остальной частью сети. В худшем случае может потребоваться до O(D) времени, чтобы все узлы свободно синхронизировались, где D – диаметр сети.

Список использованных источников:

1. N. Burri, P. von Rickenbach, and R. Wattenhofer. Ultra-low Power Data Gathering in Sensor Networks, 2007.
2. M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol, 2004.
3. J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks, 2004.
4. R. Fan and N. Lynch. Gradient Clock Synchronization, 2004.
5. T. K. Srikanth and S. Toueg. Optimal Clock Synchronization, 1987.

Гонтаренко І.В.
студент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МОДЕЛЬ БАЛАСУВАННЯ НАВАНТАЖЕННЯ У ВІРТУАЛЬНОМУ ОБЧИСЛЮВАЛЬНОМУ КЛАСТЕРІ НА ОСНОВІ КОНТЕЙНЕРІВ ДОДАТКІВ

В даній статті розглянуто практичне використання розподіленого обчислювального середовища, яке дозволить конфігурувати надані обчислювальні ресурси у відповідності до вимог програми, а також систематизація та порівняння поширених методів масштабування додатків та алгоритмів балансування навантаження.

У сучасному світі людина з усіх сторін оточена обчислювальними пристроями. Потужності обчислювальної техніки протягом всього періоду її розвитку постійно зростають, але навіть незважаючи на закон Мура [1], завжди існували та існують завдання, яким існуючих потужностей поодиноких комп'ютерів виявляється недостатньо. Саме тому останнім часом велику популярність та важливу роль відіграють високонавантажені системи