УДК 004.71:004.72

МІНІМАЛЬНИЙ TCP/IP СТЕК ДЛЯ EMBEDDED ЗАСТОСУВАНЬ

Котунов В.О.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Розглянуто можливості оптимізації стеку TCP/IP для Embedded застосувань з обмеженими обчислювальними можливостями. Проаналізовано мінімально необхідний набір протоколів для забезпечення функціонування. Наведено опис роботи протоколів та можливості їх оптимальної реалізації для Embedded систем, зокрема, інтеграція з апаратними засобами мікроконтролера. За результатами дослідження було створено та протестовано експериментальну систему в складі мережі. За результатами тестування було підтверджено працездатність розробленої системи, її сумісність з існуючими стандартами та протоколами та високу швидкодію.
Ключові слова: TCP/IP стек, Ethernet, Embedded, вбудовувані системи, STM32, ARM, IoT.

Постановка проблеми. Сучасний рівень розвитку техніки призводить до впровадження мікрокомп'ютерів у всі сфери життя. Все більше пристроїв отримують можливість приєднуватись до локальних комп'ютерних мереж або мережі Інтернет. З'явилась нова галузь – Інтернет речей (англ. Internet of Things – IoT). Окрім цього, для вирішення сучасних потреб комунікації між вузлами та пристроями все частіше використовують Ethernet або Wi-Fi та TCP/IP стек.

Описані фактори зумовлюють актуальність створення реалізації TCP/IP стеку оптимізованої під Embedded-рішення.

Аналіз останніх досліджень та публікацій. Сьогодні існує декілька аналогічних програмних стеків, наприклад, lwIP [1, 2], uIP [3]. Недоліком цих рішень можна вважати необхідність операційної системи, а це може бути неприйнятним для деяких критичних Embedded застосувань реального часу.

Виклад основного матеріалу. Для реалізації описаної системи було використано мікроконтролер загального призначення STM32F107. Це 32-бітні мікроконтролери архітектури ARM Cortex-M3, що мають вбудовану апаратну підтримку 10/100 Мб/с Ethernet на рівні MAC [4]. Також була успішно протестована робота з мікроконтролерами серії STM32F4 [5].

Для роботи з лініями 10Base-T та 100Base-T використано трансивер фізичного рівня DP83848 [6], що приєднується до мікроконтролера інтерфейсом MII або RMII.

В згаданих мікроконтролерах прийом та передача даних через Ethernet здійснюється через окремий контролер прямого доступу до пам'яті (англ. Direct Memory Access – DMA). Вбудований контролер Ethernet має розгалужену систему апаратних подій та переривань, це дає змогу ефективно використовувати процесорний час. Керування режимами роботи та інші налаштування здійснюються через регістри, що інкапсульовані у вигляді звичайних змінних.

Як було сказано раніше, на канальному рівні, практично вся обробка здійснюється апаратними засобами мікроконтролера. Так вбудований Ethernet-контролер має апаратну підтримку наступних функцій, що можуть економити процесорний час:

- фільтрація до 4 MAC-адрес отримувача
- фільтрація до 3 адрес відправника

· можливість фільтрації пакетів за маскою MAC-адрес

· окремі правила фільтрації широкомовних пакетів

· автоматична перевірка контрольної суми

· автоматичне відкидання при отриманні та додавання при передачі службової інформації (преамбула, контрольна сума та вирівнювання) до Ethernet-пакету.

Після того як Ethernet-контролер отримає пакет даних що пройде всі активовані фільтри та перевірки контрольних сум генерується переривання ETH_IRQ. Недолік даного Ethernet-контролера в тому, що на всі події генерується одне переривання, тому необхідно перевірити регістр прапорів DMASR на стан біта RS, логічна 1 означає що як мінімум один пакет був успішно отриманий. Обробка переривання полягає у послідовному виклику функцій ETH_DMAClearITPendingBit(ETH_DMA_IT_R | ETH_DMA_IT_NIS), що очищає прапори переривань та ETH_HandleRxPkt(), що в якості аргументи приймає посилання на буфер, куди будуть переписані отримані дані за допомогою DMA [7]. Подальший розбір (англ. parsing) пакету можна виконувати як в обробнику переривання ETH_IRQ, так і в основному циклі програми. Перший варіант забезпечить найбільшу швидкодію Ethernet, другий варіант використовується коли більш критичним є час виконання основної програми.

Протокол визначення адрес (англ. Adress Resolution Protocol – ARP) – забезпечує встановлення відповідності між логічною адресою (IP-адреса) та фізичною (MAC-адреса) [8]. В більшості випадків для встановлення зв'язку з віддаленим вузлом з використанням стеку протоколів TCP/IP користувачу необхідно знати лише логічну адресу віддаленого вузла. В той же час, як було сказано в попередньому розділі, пакети канального рівня мають містити фізичні адреси отримувача та відправника. Власна фізична адреса для вузла відправника відома, а от для визначення фізичної адреси віддаленого вузла за відомою логічною слугує протокол ARP.

ARP містить два типи повідомлень: запит та відповідь. Нижче наведено приклад ARP-запиту (рисунок 1).

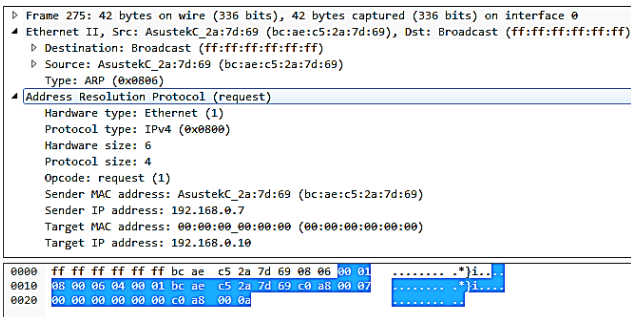

Рис. 1. Приклад ARP-запиту

В наведеному прикладі показано весь пакет, включно з протоколом Ethernet, а виділено лише ту частину пакету, що містить ARP. В даному випадку пристрій з логічною адресою 192.168.0.7 (c0:a8:00:07) та фізичною адресою bc:ae:c5:2a:7d:69 надсилає запит для встановлення фізичної адреси пристрою, якому належить логічна адреса 192.168.0.10 (c0:a8:00:0a). Також ARP пакет містить інформацію про апаратний рівень (00:01 – Ethernet), тип протоколу (08:00 – IPv4), довжину фізичної (06) та логічної (04) адрес та тип ARP пакету (00:01 – запит). Зверніть увагу на заголовок Ethernet, оскільки фізична адреса приймача в даний момент невідома, його адреса вказана як ff:ff:ff:ff:ff:ff, це – адреса широкомовних (англ. Broadcast) пакетів. Тобто, цей запит оброблятиметься на всіх пристроях, де дозволений прийом таких пакетів. Адреса відправника в Ethernet-заголовку така ж, як і в ARP, код протоколу ARP – 08:06.

Якщо пристрій одержує ARP запит зі своєю логічною адресою він повинен надіслати відповідь (рисунок 2).

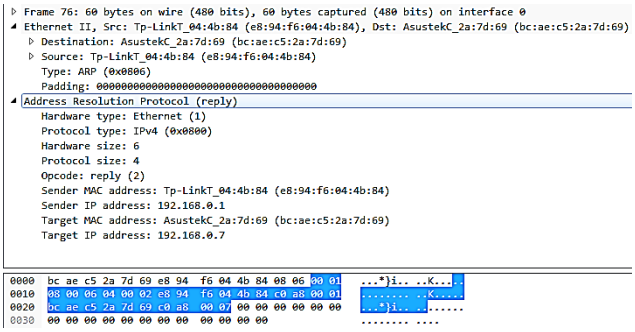

Рис. 2. Приклад ARP-відповіді

Від запиту ARP-відповідь відрізняється тим, що в цьому повідомленні вже прописані конкретні фізичні адреси, як у Ethernet заголовку, так і у самому ARP. Також змінився код типу ARP повідомлення на 00:02 – відповідь.

Також можуть існувати добровільні (англ. Gratuitous) ARP-повідомлення (рисунок 3).

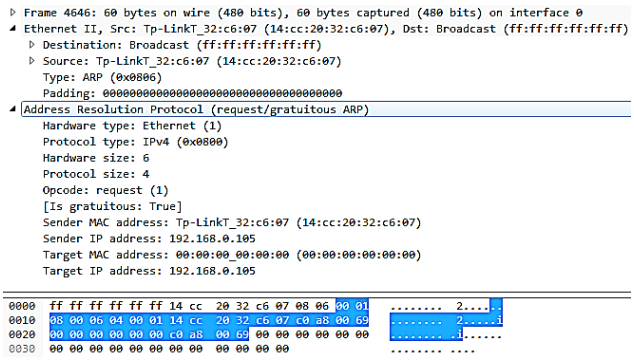

Рис. 3. Приклад Gratuitous ARP-повідомлення

Добровільні повідомлення можуть періодично надсилати пристрої для того щоб сповістити інші вузли мережі про свою фізичну адресу. Таке повідомлення не відрізняється від звичайного ARP запиту за винятком того, що логічна адреса відправника (Sender IP address) така ж як і у запитуваного пристрою (Target IP address) [9].

Результати роботи ARP зберігаються до ARP-таблиці, кожен запис в ній містить логічну адресу певного віддаленого вузла, відповідну йому фізичну адресу та термін дії запису. Якщо треба надіслати пакет до віддаленого вузла логічна адреса якого вже є в ARP-таблиці, то фізична адреса береться з таблиці, ARP запит вже не надсилається.

Термін дії записів може бути довільно обраний користувачем виходячи з характеристик мережі та зазвичай становить від декількох хвилин до години. По завершенню терміна дії запис необхідно актуалізувати, для цього надсилається повторний ARP запит. З метою зменшення навантаження на мережу, цей запит може бути вже не широкомовним, а з фізичною адресою, що була збережена в таблиці. У разі одержання відповіді на ARP запит термін дії запису в таблиці подовжується. В іншому випадку запис з таблиці видаляється.

Якщо розроблювана система самостійно не ініціює надсилання пакетів, а лише надсилає відповіді на запити інших вузлів – достатньо реалізувати лише обробку ARP запитів та надсилання відповіді. В іншому випадку необхідно також реалізувати роботу ARP-таблиці.

У випадку відповіді на запит, вся інформація, необхідна для формування пакета (фізична та логічна адреси запитуючого вузла), береться з вхідного запиту. Враховуючи те, що окрім адрес інші біти ARP-відповіді мають фіксовані значення та зміщення, зберігання та видачу адрес відповідаючого вузла можна організувати таким чином, щоб заповнити змінні частини пакету в одному циклі на 6 ітерацій (розмір фізичної адреси) – для випадку з використанням IPv4.

Міжмережевий протокол (англ. Internet Protocol – IP) – протокол мережевого рівня стека TCP/IP, основний протокол для інкапсулювання в себе протоколів передачі даних, забезпечує логічну адресацію в мережі [10].

Сьогодні активно використовується 4 версія протоколу (IPv4), але поступово вона замінюється 6 версією (IPv6). В даній роботі розглядалась 4 версія протоколу через її широке застосування та сумісність з великою кількістю існуючих систем. IPv6 в описаній системі має апаратну підтримку (як і IPv4) та може бути реалізований в рамках подальшого розвитку. Розглянемо приклад заголовку IPv4 (рисунок 4).

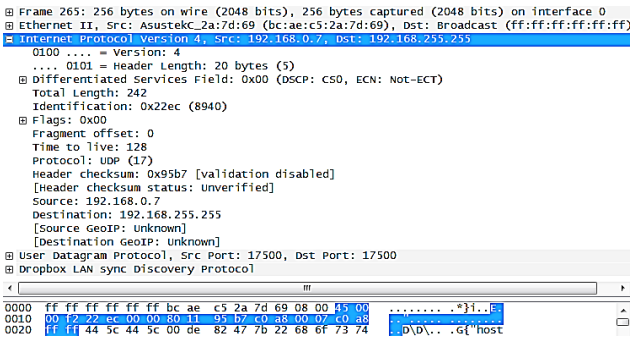

Рис. 4. Приклад IP-заголовку

Перший байт визначає версію протоколу (старші 4 біти 0x4 – IPv4) та розмір заголовку в 32-бітних словах (молодші 4 біти – 0x5 – 20 байт). Другий байт використовується для розподілу трафіку на типи обслуговування – в рамках даної роботи ігнорується. Третій та четвертий біти визначають повний розмір пакету, включно із заголовком (0x00f2 – 242 байти), разом з розміром заголовка використовуються для визначення зміщень та відокремлення пакету наступного рівня. П'ятий та шостий байти – ідентифікатор пакету (0x22ec), ігнорується для вхідних пакетів, для вихідних пакетів в якості ідентифікатора було використано статистичну інформацію про кількість надісланих пакетів, що апаратно генерується і зберігається в регістрі MMCTGFCR [7]. Сьомий та восьмий байт містять прапори та зміщення фрагменту. В рамках даної роботи прийом обробка фрагментованих пакетів не реалізовувалась, заборона фрагментації пакетів задається відповідним бітом прапором (0x4000). Дев'ятий та десятий байти встановлюють час життя пакету (англ. Time To Live – TTL), для вхідних пакетів ігнорується, для вихідних встановлюється в 0x40. Одинадцятий байт – дані якого протоколу містить пакет (0x11 – UDP) повний список номерів протоколів наведено в [11]. Дванадцятий та тринадцятий байти – контрольна сума заголовку, може бути проігнорована для вхідних пакетів, оскільки цілісність даних забезпечується контрольною сумою протоколу вищого рівня (UDP, ICMP), але мусить бути згенерована для вихідних пакетів. Методика обчислення контрольних сум наведена далі. Вісім наступних байт визначають логічні адреси відправника та адресата, відповідно. Якщо функціональність системи передбачає лише надсилання відповідей на вхідні запити, логічна адреса віддаленого вузла-приймача, як правило, береться з вхідного запиту.

Протокол міжмережевих повідомлень керування (англ. Internet Control Message Protocol – ICMP) використовується для передачі інформації про помилки та інші виключні ситуації, описаний в [12]. Даний протокол не є обов'язковим для реалізації мінімального TCP/IP стека, буде розглянуто лише два типи повідомлень, що забезпечують роботу утиліти ping. Ця утиліта дуже широко застовується під час налаштування та налагоджування мереж.

Перший тип повідомлення – ехо-запит (рисунок 5).

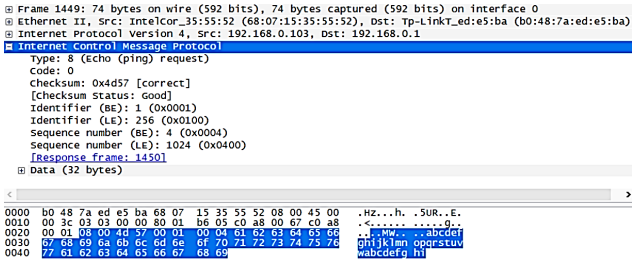

Рис. 5. Приклад ICMP ехо-запиту

Не дивлячись на те що ICMP є окремим протоколом мережевого рівня, ці пакети інкапсулюються в IP-заголовки. Сам ICMP ехо-запит містить інформацію про тип та код ICMP-повідомлення (08:00 – ехо-запит), контрольну суму ICMP-пакету (4d:57), ідентифікатор (00:01), номер запиту (00:04) та тестові дані. Оскільки формат ICMP ехо-запиту завжди сталий – об'єм тестових даних явно не вказується, він обчислюється на основі довжини пакету, що вказана в IP-заголовку. Номер запиту використовується для ідентифікації пакетів. Тестові дані можуть бути довільними.

Після одержання ехо-запиту пристрій має надіслати ехо-відовідь (рисунок 6).

Від запиту відповідь відрізняється лише типом та кодом повідомлення (0x0000) і, як наслідок, контрольною сумою.

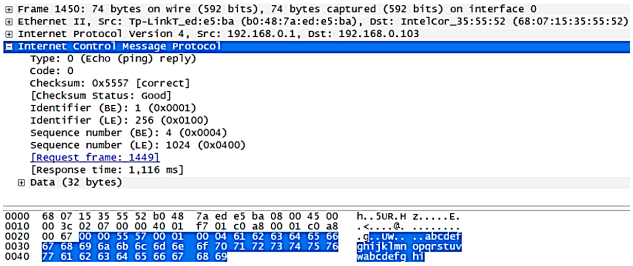

Рис. 6. Приклад ICMP ехо-відповіді

Як правило, на Embedded-системах немає потреби в реалізації всього функціоналу утиліти ping, необхідно лише реалізувати обробку ехо-запита та надсилання відповідної йому ехо-відповіді. Прийнятий пакет необхідно перевірити на відповідність контрольної суми ICMP. Оскільки відповідь від запита відрізняється лише трьома байтам – з метою оптимізації машинного часу доцільно використовувати один і той же буфер даних для формування відповіді, що й використовувався для прийому запиту.

Протокол дейтатаграм користувача (англ. User Datagram Protocol – UDP) – протокол транспортного рівня, забезпечує передачу даних користувача, що представлені протоколами прикладного рівня, описаний в [13]. Особливістю даного протоколу є відсутність засобів забезпечення цілісності, впорядкованості, та встановлення зв'язку – так званих «рукостискань» (англ. Handshake). Ця особливість забезпечує значну простоту реалізації протоколу, саме тому в описаній системі UDP був вибраний основним транспортним протоколом. При цьому передбачається що цілісність та впорядкованість даних не є критичними або забезпечуються протоколами прикладного рівня.

Розглянемо приклад UDP-заголовку (рисунок 7).

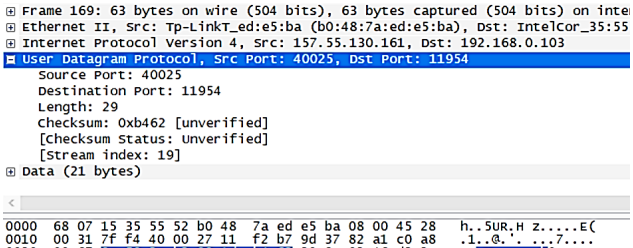

Рис. 7. Приклад UDP-заголовку.

Заголовок містить наступну інформацію: надсилаючий порт (0x9c59), порт-приймач (0x2eb2), довжина – дані та заголовок (0x001d – 29 байт), контрольна сума (0x001d). Якщо пакет запаковано в IPv4 або IPv6 то для обчислення контрольної суми, окрім даних та заголовку UDP, використовується так званий псевдозаголовок. Псевдозаголовок містить в собі частину інформації з IP-заголовку. Допускається не обчислювати контрольну суму записавши в її значення нулі. Також способи оптимізації UDP наведені у [14].

Спосіб обчислення контрольних сум для протоколів, що описані в даній роботі, визначено в [15].

Описану вище систему розбору вхідного пакету можна зобразити у вигляді дерева (рисунок 8).

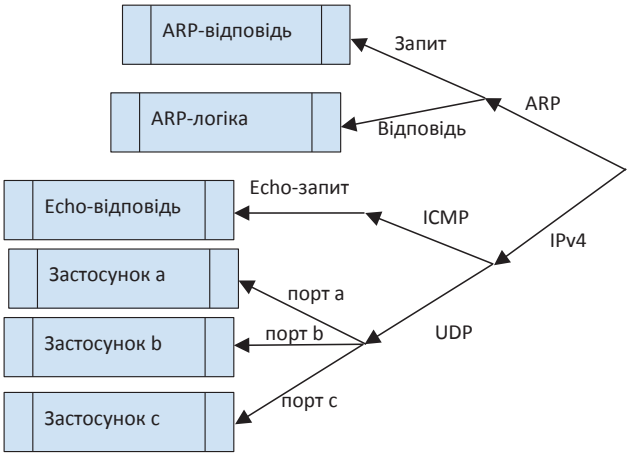

Рис. 8. Розбір вхідного пакету

Як вже було сказано, на рівні MAC (фізичних адрес) пакети фільтруються апаратно, розбір починається з мережевого рівня. Зверніть увагу, не дивлячись на те що за функціональним призначенням ICMP є протоколом мережевого рівня, під час розбору він виділяється на вищому рівні, разом з UDP. Це пов'язано з тим, що ICMP має заголовок, що аналогічний IPv4. Обробка пакетів з UDP розгалуджується на декілька гілок – портів. Кількість портів та їх номери визначаються користувачем. Зазвичай, кожному застосунку (протоколу прикладного рівня) виділяють окремий порт. Наприклад, порт 123 використовується для налаштувань системи через протокол Modbus, порт 124 – для передачі навігаційних даних у форматі NMEA, порт 125 – для обміну текстовими повідомленнями. У відповідь на отриманий пакет застосунок, як правило, генерує та надсилає пакет-відповідь.

Пакет-відповідь це – підготовлена послідовність байт, що визначають заголовки одразу всіх рівнів починаючи з канального. До заголовків додається інформація, яку необхідно передати. Після цього обчислюються контрольні суми, їх значення підставляються на відповідні місця в пакеті. Буфер зі сформованим пакетом та його довжина передається у функцію ETH_HandleTxPkt(), подальша передача пакету відбувається автоматично.

Результати. Для оцінки ефективності розробленої системи варто використовувати лише функціонал самого стеку, уникаючи впливу прикладних застосунків. Таким оціночним критерієм можна вважати час відгуку на ICMP Echo-запит. Як було описано раніше, обробка даного запиту та генерування відповіді відбуваються повністю засобами розробленого стеку, а з рисунку 8 видно, що складність розбору такого запиту аналогічна до складності розбору вхідного UDP-пакету.

Система була запущена на мікроконтролері STM32F107RBT6 з тактовою частотою 70МГц. Через Ethernet-комутатор до даної системи був також приєднаний персональний комп'ютер з процесором Intel Core i7–6500U з тактовою частотою 2,6ГГц та операційною системою Ubuntu 14.04 LTS – тестова система 1, цей комп'ютер надсилав ICMP echo-запити та вимірював час відгуку. Тестова система 2 є персональним комп'ютером на основі процесора AMD

Phenom II X6 1100T та під керуванням операційної системи Windows 7. Для порівняння з іншими Embedded системами, в цю ж мережу також увімкнено точку доступу Ubiquiti Rocket M2 на основі мікроконтролера Qualcomm Atheros AR7241 з тактовою частотою 400МГц та набором утиліт BusyBox (тестова система 3). Ще однією системою що використовувалась для порівняння був маршрутизатор TP-Link (тестова система 4).

Розглянемо результати тестування системи 2 (рисунок 9).

```
PING 192.168.0.7 (192.168.0.7) 56(84) bytes of data.
64 bytes from 192.168.0.7: icmp_seq=1 ttl=128 time=1.96 ms
64 bytes from 192.168.0.7: icmp_seq=2 ttl=128 time=0.523 ms
64 bytes from 192.168.0.7: icmp_seq=3 ttl=128 time=0.527 ms
64 bytes from 192.168.0.7: icmp_seq=4 ttl=128 time=0.594 ms
64 bytes from 192.168.0.7: icmp_seq=5 ttl=128 time=0.541 ms
64 bytes from 192.168.0.7: icmp_seq=6 ttl=128 time=0.520 ms
64 bytes from 192.168.0.7: icmp_seq=7 ttl=128 time=0.557 ms
64 bytes from 192.168.0.7: icmp_seq=8 ttl=128 time=0.525 ms
64 bytes from 192.168.0.7: icmp_seq=9 ttl=128 time=0.505 ms
64 bytes from 192.168.0.7: icmp_seq=10 ttl=128 time=0.556 ms
^C
--- 192.168.0.7 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.505/0.680/1.960/0.428 ms
```

Рис. 9. Тестова система 2

Зверніть увагу, час першого відгуку значно більший за інші (1,966 мс). Це зумовлено тим, що тестування проводилися з очищеною ARP-таблицею і потрібен час на виконання та обробку ARP запиту. Таким чином, даний тест охоплює обробку не лише ICMP, а й ARP.

На вибірці у 10 запитів тестова система 1 мала середній час відгуку 0,68 мс, мінімальний – 0,505 мс, та максимальний – 1,96 мс.

Результати тестування систем 3 (рисунок 10) та 4 (рисунок 11) подано нижче.

```
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.887 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.525 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.555 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=64 time=0.528 ms
64 bytes from 192.168.0.1: icmp_seq=5 ttl=64 time=0.583 ms
64 bytes from 192.168.0.1: icmp_seq=6 ttl=64 time=0.526 ms
64 bytes from 192.168.0.1: icmp_seq=7 ttl=64 time=2.14 ms
64 bytes from 192.168.0.1: icmp_seq=8 ttl=64 time=0.529 ms
64 bytes from 192.168.0.1: icmp_seq=9 ttl=64 time=0.555 ms
64 bytes from 192.168.0.1: icmp_seq=10 ttl=64 time=0.528 ms
^C
--- 192.168.0.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8999ms
rtt min/avg/max/mdev = 0.525/0.735/2.140/0.480 ms
```

Рис. 10. Тестова система 3

```
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=0.677 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=0.549 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=0.528 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=0.569 ms
64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=0.427 ms
64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=11.1 ms
64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=0.527 ms
64 bytes from 192.168.1.20: icmp_seq=8 ttl=64 time=0.517 ms
64 bytes from 192.168.1.20: icmp_seq=9 ttl=64 time=0.527 ms
64 bytes from 192.168.1.20: icmp_seq=10 ttl=64 time=4.32 ms
^C
--- 192.168.1.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.427/1.979/11.141/3.256 ms
```

Рис. 11. Тестова система 4

Для визначення часу відгуку тестової системи 1 використовувалась тестова система 4 та встановлений на ній набір утиліт BusyBox. В цей час керування системою 4 здійснювалося через протокол SSH з системи 1 (рисунок 12).

Мінімальний час відгуку розробленої системи склав 0,467 мс, а середній – 0,574 мс. Час обробки першого пакету склав 0,935 мс, цей результат дещо поступається більшості протестованих систем (за винятком системи 2), але середній час відгуку менший, ніж в усіх інших системах.

```
PING 192.168.1.116 (192.168.1.116): 56 data bytes
64 bytes from 192.168.1.116: seq=0 ttl=64 time=0.874 ms
64 bytes from 192.168.1.116: seq=1 ttl=64 time=0.690 ms
64 bytes from 192.168.1.116: seq=2 ttl=64 time=0.697 ms
64 bytes from 192.168.1.116: seq=3 ttl=64 time=0.726 ms
64 bytes from 192.168.1.116: seq=4 ttl=64 time=0.686 ms
64 bytes from 192.168.1.116: seq=5 ttl=64 time=0.689 ms
64 bytes from 192.168.1.116: seq=6 ttl=64 time=0.474 ms
64 bytes from 192.168.1.116: seq=7 ttl=64 time=0.796 ms
64 bytes from 192.168.1.116: seq=8 ttl=64 time=0.689 ms
64 bytes from 192.168.1.116: seq=9 ttl=64 time=0.435 ms
64 bytes from 192.168.1.116: seq=10 ttl=64 time=0.696 ms
^C
--- 192.168.1.116 ping statistics ---
11 packets transmitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 0.435/0.677/0.874 ms
```

Рис. 12. Тестова система 1

```
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.
64 bytes from 192.168.0.15: icmp_seq=1 ttl=64 time=0.935 ms
64 bytes from 192.168.0.15: icmp_seq=2 ttl=64 time=0.527 ms
64 bytes from 192.168.0.15: icmp_seq=3 ttl=64 time=0.528 ms
64 bytes from 192.168.0.15: icmp_seq=4 ttl=64 time=0.529 ms
64 bytes from 192.168.0.15: icmp_seq=5 ttl=64 time=0.535 ms
64 bytes from 192.168.0.15: icmp_seq=6 ttl=64 time=0.547 ms
64 bytes from 192.168.0.15: icmp_seq=7 ttl=64 time=0.613 ms
64 bytes from 192.168.0.15: icmp_seq=8 ttl=64 time=0.529 ms
64 bytes from 192.168.0.15: icmp_seq=9 ttl=64 time=0.532 ms
64 bytes from 192.168.0.15: icmp_seq=10 ttl=64 time=0.467 ms
^C
--- 192.168.0.15 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.467/0.574/0.935/0.125 ms
```

Рис. 13. Тестування розробленої системи

Не можна сказати що такий тест є абсолютно об'єктивним, але він показує що розроблена система, за рівнем швидкодії, знаходиться на одному рівні з сучасними рішеннями, що використовують значно більші обчислювальні ресурси.

Висновки. У ході виконання описаної роботи було проведено дослідження протоколів стеку TCP/IP. За результатами дослідження було виділено мінімальний набір протоколів необхідний для функціонування стеку.

Описана реалізація протоколу забезпечує основні функції для встановлення та адміністрування мережевого з'єднання та забезпечує передачу даних протоколом UDP. Дане рішення широко використовує апаратні можливості обладнання та не потребує операційної системи. Як наслідок, розроблений стек може бути використаний у критичних застосуваннях реального часу, навіть на пристроях з обмеженою обчислювальною потужністю.

Для підтвердження результатів дослідження була створена та успішно протестована дослідна система. Не дивлячись на те, що описана система не використовує операційну систем та має обмежені обчислювальні ресурси (тактова частота 70 МГц), вона, за результатами тестування, знаходиться на одному рівні з сучасними рішеннями, що використовують значно більші потужності.

Таке рішення може бути корисним для високошвидкісної передачі даних, для IoT-систем, для об'єднання Embedded рішень в мережі, їх інтеграції з існуючими локальними комп'ютерними мережами та мережею Інтернет, для створення користувацьких веб-інтерфейсів та ін.

Подальший розвиток розробленої системи може полягає у розширенні переліку протоколів

що підтримуються, збільшенні надійності та навантажувальної спроможності.

Ймовірно, одним з найширших застосувань описаної системи, що орієнтоване на широке коло споживачів, може бути забезпечення користувацького веб-інтерфейсу. У зв'язку з цим, розвиток підтримки HTTP та функціоналу веб-сервера може стати ще одним напрямком майбутнього розвитку.

Список літератури:

1. Dunkels A. Design and Implementation of the lwIP TCP/IP Stack / A. Dunkels. – Swedish Institute of Computer Science, 2001 – Т. 2. – С. 77.
2. Kong D. Transplant and Application of LWIP in ARM Platform [J] / D. Kong, J. Zheng – Communications Technology. – 2008. – Т. 6. – С. 38–40.
3. Dunkels A. uIP-A free small TCP/IP stack / A. Dunkels – The uIP. – 2002.
4. STM32F107RB Mainstream Connectivity line [Електронний ресурс] Режим доступу: http://www.st.com/en/microcontrollers/stm32f107rb.html
5. STM32F4 Series [Електронний ресурс] Режим доступу: http://www.st.com/en/microcontrollers/stm32f4-series.html
6. DP83848-HT [Електронний ресурс] Режим доступу: http://www.ti.com/product/DP83848-HT
7. RM0008 Reference manual [Електронний ресурс] Режим доступу: http://www.st.com/resource/en/reference_manual/cd00171190.pdf
8. Plummer D. An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware / D. Plummer – Network Working Group. – 1982.
9. Gratuitous ARP [Електронний ресурс] Режим доступу: https://wiki.wireshark.org/Gratuitous_ARP
10. RFC 791. Internet protocol / – Internet standard. – 1981.
11. Protocol Numbers [Електронний ресурс] Режим доступу: http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
12. Postel J. RFC 792. Internet Control Message Protocol / J. Postel – Internet standard. – 1981.
13. Postel J. RFC 768. User Datagram Protocol / J. Postel – Internet standard. – 1980.
14. Larzon L. UDP lite for real time multimedia applications / L. A. Larzon, M. Degermark, S. Pink – Hewlett-Packard Laboratories, 1999.
15. Braden R. RFC 1071. Computing the Internet Checksum / R. Braden, D. Borman, C. Partridge – Internet standard. – 1988.

Котунов В.О.
Национальный технический университет Украины
«Киевский политехнический институт имени Игоря Сикорского»

МИНИМАЛЬНЫЙ TCP/IP СТЕК ДЛЯ EMBEDDED ПРИЛОЖЕНИЙ

Аннотация
Рассмотрены возможности оптимизации стека TCP/IP для Embedded приложений с ограниченными вычислительными возможностями. Проанализирован минимально необходимый набор протоколов для обеспечения функционирования. Дано описание работы протоколов и возможности их оптимальной реализации для Embedded систем, в частности, интеграция с аппаратными средствами микроконтроллера. По результатам исследования было создано и протестировано экспериментальную систему в составе сети. По результатам тестирования была подтверждена работоспособность разработанной системы, ее совместимость с существующими стандартами и протоколами и высокое быстродействие.
Ключевые слова: TCP/IP стек, Ethernet, Embedded, встраиваемые системы, STM32, ARM, IoT.

Kotunov V.O.
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

MINIMAL TCP/IP STACK FOR EMBEDDED APPLICATIONS

Summary
The possibilities of optimizing the TCP/IP stack for Embedded applications with limited computing capabilities are considered. The minimum necessary set of protocols for functioning is analyzed. The description of work of protocols and possibilities of their optimal realization for embedded systems is given, in particular, integration with hardware of microcontroller. According to the results of the research, an experimental system was created and tested in the network. According to the results of the testing, the efficiency of the developed system, its compatibility with existing standards and protocols and high performance were confirmed.
Keywords: TCP/IP stack, Ethernet, Embedded, Embedded Systems, STM32, ARM, IoT.