УДК. 004.021, 519.2

Максимов В. В., канд. техн. наук, доцент (Тел. +380 68 810 40 47. E-mail: maksimov46@ukr.net)
Чмихун С. О., магістрант (E-mail: serega4sa@yandex.ru)
(Національний технічний університет України «КПІ», Інститут телекомунікаційних систем, м. Київ)

КЛАСИФІКАЦІЯ АЛГОРИТМІВ БОРОТЬБИ З ПЕРЕНАВАНТАЖЕННЯМИ

Максимов В. В., Чмихун С. О. Класифікація алгоритмів боротьби з перевантаженнями. Дана робота присвячена аналізу існуючих методів боротьби з перевантаженнями в мережах, що використовують протокол TCP (Transmission Control Protocol) в якості основного протоколу передачі даних, і подальшого синтезу їх класифікації. Показано, що більшість сучасних алгоритмів є більш універсальними і здатні однаково ефективно запобігати перевантаження в різних мережах.

Ключові слова: TCP/IP мережа, протокол TCP, пропускна здатність, перевантаження мережі, боротьба з перевантаженням, класифікація

Максимов В. В., Чмыхун С. А. Классификация алгоритмов борьбы с перегрузками. Данная работа посвящена анализу существующих методов борьбы с перегрузками в сетях, использующих протокол TCP (Transmission Control Protocol) в качестве основного протокола передачи данных, и последующему синтезу их классификации. Показано, что большинство современных алгоритмов являются более универсальными и способны одинаково эффективно предотвращать перегрузки в различных сетях.

Ключевые слова: TCP/IP сеть, протокол TCP, пропускная способность, перегрузка сети, борьба с перегрузкой, классификация

Maximov V. V., Chmykhun S. O. Classification of algorithms of controlling networks congestions. This work is devoted to the analysis of existing methods of controlling congestion in networks, which use the TCP (Transmission Control Protocol) as the main data transfer protocol and the subsequent synthesis of their classification. It is shown, that most of modern algorithms are more versatile and can equally effectively prevent overload in different networks.

Keywords: TCP/IP network, TCP protocol, carrying capacity, congestion network, controlling congestion, classification

1. Постановка задачі дослідження. Алгоритми боротьби з перевантаженнями відіграють значну роль у питанні збільшення ефективності використання мереж зв'язку, яке з кожним роком стає все більш актуальним. Сьогодні існує велика кількість даних алгоритмів, які задовольняють ті чи інші потреби користувачів. В зв'язку з цим є необхідність в класифікації цих алгоритмів з метою визначення переваг та недоліків їх застосування в певних умовах роботи мереж [1, 2].

Метою даної роботи є побудова класифікації алгоритмів боротьби з перевантаженням в мережах які використовують протокол TCP (Transmission Control Protocol). В основі пропонованої класифікації лежить існуюча класифікація систем RED [3], яку пропонується розширити та доповнити іншими реалізаціями алгоритмів боротьби з перевантаженнями, які використовуються на сьогоднішній день. При побудові пропонованої класифікації використовується ієрархічний метод, який передбачає послідовне ділення множини об'єктів на підпорядковані їм класи. Класи позначаються прописними латинськими літерами.

Класифікація проведена за наступними характеристиками:

– кількість модулів RED – *клас M (Module):*

– вид функції втрати пакетів, тобто у випадку коли відбувається втрата пакетів алгоритм вважає що система знаходиться у перевантаженому стані і починає виконувати певні дії для відновлення нормального стану системи – *клас L (Loss)*;

– вид функції черги затримки, тобто на основі даних про кількість пакетів, що знаходяться в буфері маршрутизатора (в черзі), алгоритм робить висновки про стан системи: якщо кількість пакетів перевищує певну верхню межу, яка встановлюється експериментальним шляхом, то фіксується перевантаження, яке треба усувати – *клас QD (Queue Delay)*;

– вид функції скидання – *клас D (Dropping);*

– вид функції втрат і затримок, тобто враховуються як дані про втрати пакетів, так і дані про чергу затримки – *клас B (Both)*.

Для ілюстрації побудованої класифікації наведемо опис найбільш характерних алгоритмів для кожного класу.

2. Огляд алгоритмів боротьби з перевантаженнями на базі протоколу TCP

2. 1. Алгоритм TCP Tahoe. Алгоритм розроблений у 1988 році і заснований на принципі "збереження пакетів", тобто, якщо з'єднання встановлено і працює на наявній пропускній здатності, то пакет не вводиться в мережу, поки попередній пакет не покине мережу. Даний алгоритм реалізує цей принцип за допомогою підтверджень, тому що отримання підтвердження означає, що пакет покинув мережу і був отриманий приймачем. Алгоритм роботи цього протоколу описується наступним чином:

$$if\ (cwnd < sstresh)$$
$$cwnd = cwnd + 1$$
$$else$$
$$cwnd = cwnd + 1/cwnd$$

Значення тайм-ауту обчислюється за формулою:

$$RTO = \overline{RTT} + 4\sigma_{\overline{RTT}},$$

де σ – середньоквадратичне відхилення среднього значення RTT (Round-trip delay time – час, витрачений на проходження пакета від клієнта до сервера і назад).

Важливим є те, що алгоритм Tahoe виявляє втрату пакетів, використовуючи таймаут. Таким чином, може пройти деякий час, перш ніж буде помічена втрата пакета, а потім знову буде відправлений загублений пакет [1].

Виходячи із вищесказаного, цей алгоритм можна віднести до класу ***L***.

До недоліків можна віднести те, що втрачений пакет і всі, надіслані після нього, пакети (незалежно від того, підтверджено їх отримання чи ні) пересилаються повторно. При високій ймовірності втрати це істотно знижує пропускну здатність і збільшує і без того високе завантаження каналу.

2.2. Алгоритм TCP Reno. Алгоритм розроблений у 1990 році і використовує алгоритми AIMD (Additive increase/multiplicative decrease – адитивне збільшення і мультиплікативне зменшення). Суть полягає в тому, що під час фази повільного старту розмір вікна збільшується для кожного отриманого ACK (Acknowledgment – підтвердження доставки пакету) до тих пір, поки не сталася втрата пакетів (відправник отримує тричі дубльований ACK). В момент втрати пакетів вікно зменшується вдвічі і починає працювати алгоритм збільшення, поки не сталася наступна втрата пакетів. Таким чином відбувається періодичне коливання вікна, яке дозволяє повністю не зупиняти передачу даних при перевантаженні, а лише зменшувати кількість пакетів, що передаються [4].

Алгоритм роботи описується наступними рівняннями:

$$cwnd(t + t_A) =$$
$$= \begin{cases} cwnd(t) + 1, & if\ cwnd(t) < ssth(t) - \text{фаза повільного старту}, \\ cwnd(t) + \dfrac{1}{cwnd(t)}, & if\ cwnd(t) \geq ssth(t) - \text{фаза виключення перевантаження}, \end{cases}$$

де *ssth*(*t*) – значення порогу, при якому TCP переходить із фази повільного старту у фазу виключення перевантаження;

cwnd(*t*) – розмір вікна перевантаження;

$t + t_A$ – момент часу, коли відправник отримує підтвердження доставки;

t – поточний час.

Коли в результаті тайм-ауту детектується втрата пакета значення *cwnd*(*t*) та *ssth*(*t*) оновлюється наступним чином:

$$cwnd(t) = 1; \qquad ssth(t) = (cwnd(t))/2.$$

Коли ж телекомунікаційна система фіксує втрату пакета згідно алгоритму швидкої повторної передачі, *cwnd*(*t*) і *ssth*(*t*) оновлюються інакше:

$$cwnd(t) = ssth(t); \qquad ssth(t) = (cwnd(t))/2.$$

Виходячи з вище сказаного, можна зробити висновки, що даний алгоритм працює на основі інформації про втрати пакетів, тому має бути віднесений до класу *L*.

Слід зазначити, що цей алгоритм ефективно працює при невеликій кількості втрат пакетів. Коли відбувається кілька втрат пакетів в одному вікні, то продуктивність TCP Reno знижується. Причина в тому, що якщо сталася втрата декількох пакетів, то перша інформація про втрату пакету надходить тоді, коли отримано дублікат ACK. Але інформація про втрату другого пакету, який був загублений, надійде тільки після того, як ACK для повторно відправленого першого сегменту досягне відправника після одного RTT. Також до недоліків можна віднести те, що даний алгоритм не відрізняє втрат, що сталися через затримки пакетів, тому може випадково прийняти мережу перевантаженою. Це призводить до того, що він малоефективний в мережах з великими затримками.

2.3. Алгоритм TCP Vegas. Алгоритм розроблений у 1994 році. Основна ідея TCP Vegas полягає у виявленні перевантаження на ранній стадії в маршрутизаторах між джерелом і одержувачем до втрати пакетів. Прогнозування цієї стадії зародження перевантаження відбувається завдяки контролю різниці між виміряною пропускною здатністю і очікуваною.

$$DIFF = (Expected - Actual)$$

$$DIFF \cdot BaseRTT = (\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT}) \cdot BaseRTT \approx 0$$

$$cwnd(t + t_A) = \begin{cases} cwnd(t) + 1, & if\ DIFF < \dfrac{\alpha}{BaseRTT} \\ cwnd(t), & if\ \dfrac{\alpha}{BaseRTT} \le DIFF \le \dfrac{\beta}{BaseRTT} \\ cwnd(t) - 1, & if\ \dfrac{\beta}{BaseRTT} < DIFF \end{cases},$$

де *RTT* – зареєстроване RTT;

BaseRTT – найменше у даному циклі RTT;

β – верхня межа;

α – нижня межа.

Як бачимо, алгоритм TCP Vegas використовує значення *DIFF* для визначення кількості пакетів, що перебувають в обмеженому буфері. Якщо кількість пакетів, що знаходяться в черзі, перевищує β, алгоритм TCP Vegas має на увазі, що з'єднання було залучено в перевантаженні, та лінійно зменшує вікно перевантаження (на 1 сегмент за кожне RTT). Якщо ж кількість пакетів в черзі не перевищує α, алгоритм збільшує вікно перевантаження лінійно. В іншому випадку, розмір вікна залишається незмінним [5].

Виходячи з вищеописаного принципу дії даного алгоритму, можна зробити висновки, що даний алгоритм працює на основі інформації про кількість пакетів, що знаходяться в черзі, тому має бути віднесений до класу *D*.

До недоліків даного методу можна віднести те, що він потребує високої роздільної здатності таймеру відправника.

2.4. Алгоритм TCP NewReno. Алгоритм розроблений у 1999 році і являється незначною модифікацією алгоритму TCP Reno. Він здатний виявляти багаторазові втрати пакетів і, отже, є набагато більш ефективним, ніж TCP Reno у разі втрат декількох пакетів.

Так само як і TCP Reno, TCP New Reno також входить у фазу швидкої повторної передачі, коли отримано кілька дублікатів підтверджень, однак його принцип роботи відрізняється від Reno тим, що алгоритм не виходить з фази швидкого відновлення до тих пір, поки всі дані, які були відправлені під час цієї фази, не будуть підтверджені.

$cwnd(t) = ssthresh + 3 \cdot SMSS$ – швидка повторна передача,

$cwnd(t) = SMSS$ – швидке відновлення,

де *SMSS* – максимальний розмір сегменту відправника.

Таким чином, даний алгоритм усуває проблеми, з якими стикається TCP Reno, який знижує вікно перевантаження в кілька разів. А також виключається проблема численних повторних пересилок пакетів [6].

Виходячи із вище сказаного, цей алгоритм треба віднести до класу *L*.

Недолік алгоритму TCP NewReno полягає в тому, що для виявлення втрати кожного пакета потрібен очікувати час RTT. Коли отримано ACK для першого повторно переданого сегменту, тільки тоді можна визначити, який ще інший сегмент був загублений.

2.5. Алгоритм TCP Veno. Алгоритм розроблений у 2001 році. TCP Veno намагається виділити втрати, що не пов'язані з перевантаженням, для того щоб не застосовувати механізм боротьби з перевантаженням там, де це не потрібно. Даний алгоритм використовує механізм, аналогічний тому, що використовується в TCP Vegas, але з метою визначення характеру втрат пакетів. Зокрема, якщо втрата пакетів виявляється в той час як оцінка показує, що мережа не перевантажена, встановлюється, що втрата є випадковою [7].

Тобто алгоритм роботи даного протоколу має наступний вигляд:

$if\ (N < \beta) -$ доступна смуга не повністю використана,

$set\ cwnd = cwnd + 1/cwnd\ when\ each\ new\ ACK\ is\ received$

$else\ if\ (N \geq \beta) -$ доступна смуга повністю використана,

$set\ cwnd = cwnd + 1/cwnd\ when\ every\ other\ new\ ACK\ is\ received$,

де *N* – кількість пакетів в буфері; β – верхня межа кількості пакетів в черзі.

Виходячи з вище сказаного, можна зробити висновки, що даний алгоритм працює не лише на основі інформації про втрати пакетів, а також аналізує характер втрат, тобто враховує можливість затримок. Тому даний алгоритм має бути віднесений до класу *B*.

2.6. Алгоритм TCP Westwood. Алгоритм розроблений у 2001 році і являє собою модифікацію алгоритму боротьби з перевантаженнями TCP Reno. Головною відміною є те, що TCP Westwood не дуже чутливий до випадкових втрат. Досягається це шляхом контролю швидкості повернення підтверджень ACK. На основі цих даних розраховується розмір вікна перевантаження та поріг повільного старту. Далі TCP Westwood намагається обрати такий розмір вікна перевантаження і поріг повільного старту, який би забезпечував ефективне використання смуги пропускання під час перевантаження мережі.

Тобто алгоритм роботи даного протоколу набуває вигляду:

$if\ (n\ DUPACKs\ are\ received)$

$ssthresh = (BWE * RTT_{min})/seg\_size;$

$if\ (cwnd > ssthresh) -$ уникнення перевантаження

$cwnd = ssthresh;$

$endif$

$endif$,

де *seg_size* – ідентифікатор довжини корисного навантаження сегмента TCP в бітах;

BWE – оцінка пропускної здатності;

ssthresh – поріг повільного старту.

Цей механізм є особливо ефективним в безпроводових мережах, де випадкові втрати через проблеми в радіоканалі часто невірно витлумачені як ознаки перевантаження мережі і приводять до зайвого зменшення розміру вікна [8].

Таким чином, як бачимо, даний алгоритм використовує інформацію про час повернення підтверджень і враховує можливість затримок. Тому віднесемо його до класу *B*.

2.7. Алгоритм TCP Hybla. Алгоритм розроблений у 2003-2004 роках. Основна ідея TCP Hybla є отримання протягом тривалого RTT з'єднання (наприклад в супутникових і безпрводових мережах) тієї ж самої миттєвої швидкості передачі $B(t)$, порівняно швидкого опорного TCP з'єднання (наприклад в дротових мережах). На відміну від багатьох інших рішень, це новий алгоритм заснований на аналітичному дослідженні еволюції вікна перевантаження в часі залежно від затримок з'єднань [9].

Алгоритм роботи даного протоколу має наступний вигляд:

$$cwnd(t) = \begin{cases} \rho \cdot 2^{\rho t/RTT}, 0 \le t \le t_{\gamma,0} \\ \rho\left[\rho \frac{t-t_{\gamma,0}}{RTT} + \gamma\right], t > t_{\gamma,0} \end{cases},$$

де ρ – нормалізований RTT;

$t_{\gamma,0}$ – час перемикання, що визначається за формулою $t_{\gamma,0} = RTT_0 \cdot \log_2 \gamma$;

γ – величина порогу *ssthres*h.

Тобто, як бачимо, даний алгоритм використовує параметр затримки пакетів під час визначення стану, в якому знаходиться система, тобто треба віднести його до класу *D*.

2.8. Алгоритм TCP CUBIC. Алгоритм розроблений у 2006 році. Головною особливістю TCP CUBIC є те, що функція зростання розміру вікна перевантаження визначається як:

$$W(t) = 3C(t-K) + W_{max},$$

де C – параметр CUBIC;

t – час з моменту останнього зменшення розміру вікна;

K – період часу, який необхідний для збільшення W до W_{max}, значення якого обчислюється з використанням виразу:

$$K = \sqrt[3]{\frac{W_{max} \cdot \beta}{C}}.$$

Тобто розмір вікна перевантаження визначається в режимі реального часу, що призводить до того, що його зростання не залежить від RTT. Період перевантаження CUBIC залежить від коефіцієнту втрат пакетів поодинці. Пропускна здатність TCP визначається як коефіцієнтом втрат пакетів, так і RTT, а пропускна здатність CUBIC визначається лише швидкістю втрат пакетів [10].

Виходячи з вище сказаного, можна зробити висновки, що даний алгоритм використовує як дані про втрати пакетів, так і дані про затримки. Тому він має бути віднесений до класу *B*.

2.9. Алгоритм TCP Illinois. Алгоритм розроблений у 2006 році. Ідея реалізації TCP Illinois полягає в наступному: коли середня черга затримки (d_a) мала, відправник припускає, що перевантаження не є неминучим і встановлює велике α (параметр збільшення вікна) й невеликие β (параметр зменшення вікна); коли d_a великий, відправник припускає, що перевантаження неминуче, і встановлює невелике α і велике β. Такий алгоритм має назву увігнутий-AIMD або C-AIMD і має вигляд:

$$\alpha = \begin{cases} \alpha_{max}, \; if \; \mathrm{d}_a \le \mathrm{d}_1 \\ \dfrac{k_1}{k_2 + \mathrm{d}_a}, \; otherwise \end{cases},$$

$$\beta = \begin{cases} \beta_{min}, & if \quad \mathrm{d}_a \le \mathrm{d}_2 \\ k_3 + k_4 \mathrm{d}_a, & if \quad \mathrm{d}_a < \mathrm{d}_a < \mathrm{d}_a \\ \beta_{max}, & otherwise \end{cases}.$$

Алгоритм C-AIMD використовує втрати, для визначення напрямку, а затримку для регулювання темпу зміни розміру вікна. Таким чином, втрати є основним сигналом о перевантаженні, а затримка вторинним [11]. Із вище сказаного, можна зробити висновки, що даний алгоритм має бути віднесений до класу *B*.

На основі проведеного аналізу алгоритмів боротьби з перевантаженнями побудована класифікація, яка представлена на Рис. 1.

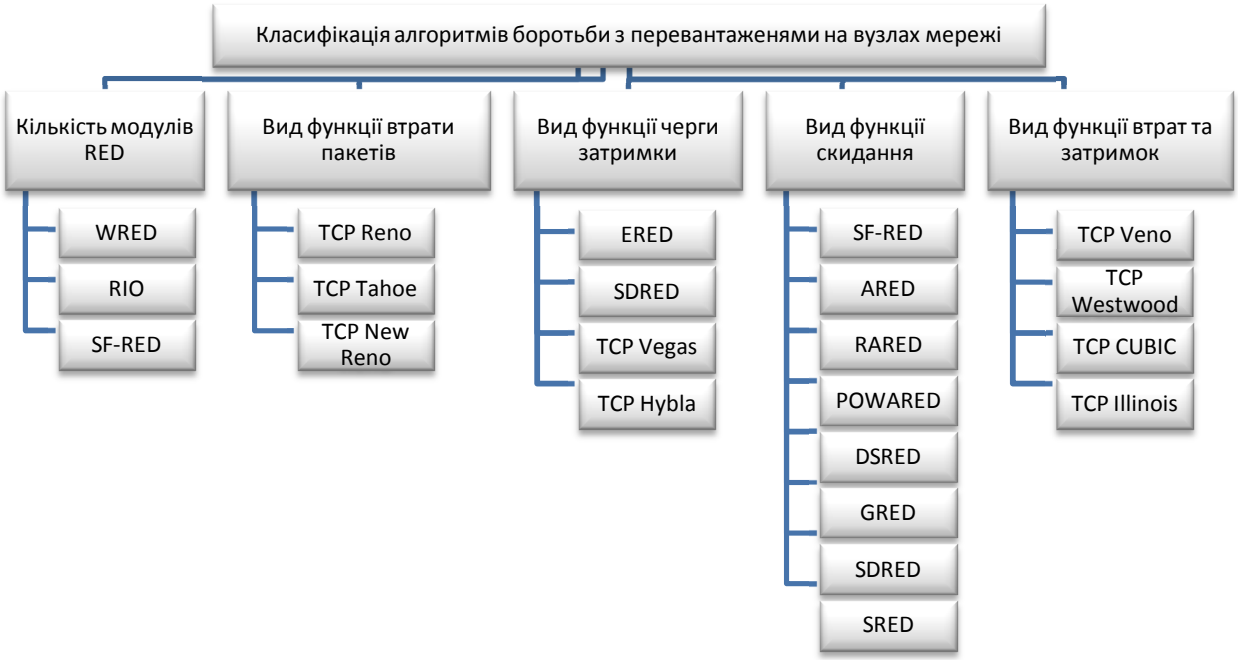

Рис. 1. Класифікація алгоритмів боротьби з перевантаженнями на основі оцінки методів визначення стану системи

В Табл. 1 наведено результуючий розподіл по класах розглянутих алгоритмів, що базуються на протоколі TCP.

Використання даної класифікації дозволить забезпечити більш ефективне використання конкретних реалізацій алгоритмів боротьби з перевантаженнями в TCP/IP мережах, в залежності від характеристик, що притаманні цим мережам

3. Висновки. Всі розглянуті модифікації алгоритмів боротьби з перевантаженнями протоколу TCP можна розділити на три підгрупи за ознаками, що беруться в якості основних при визначенні характеру втрати пакетів. Кожна з підгруп має свої особливості і в різних випадках використання цих алгоритмів вони по різному впливають на пропускну здатність мережі. Розроблена класифікація може бути корисною при побудові мереж, що використовують протокол TCP.

Розподіл модифікацій протоколу TCP по класах Табл. 1.

Клас / Алгоритм	*L*	*QD*	*B*
TCP Tahoe	+	-	-
TCP Reno	+	-	-
TCP Vegas	-	+	-
TCP NewReno	+	-	-
TCP Veno	-	-	+
TCP Westwood	-	-	+
TCP Hybla	-	+	-
TCP CUBIC	-	-	+
TCP Illinios	-	-	+

Література

1. Jacobson V. Congestion Avoidance and Control / Van Jacobson // SIGCOMM '88, ACM. –Aug., 1988.

2. Мухин В. Е. Методы и средства эффективного управления передачей данных в защищенных компьютерных сетях / В. Е. Мухин, Луай Дарвиш // Системні дослідження та інформаційні технології. – 2005. – №2. – С. 61-75.

3. Королькова А. В. К вопросу о классификации алгоритмов RED / А. В. Королькова, Д. С. Кулябов, А. И. Черноиванов // Вестник РУДН. – 2009. – №3. – С. 34-46.

4. Jacobson V. Berkeley TCP evolution from 4.3-tahoe to 4.3-reno / V. Jacobson // Proc. of the 18th Internet Engineering Task Force. – Vancouver. – August 1990.

5. Brakmo L. TCP Vegas: New techniques for congestion detection and avoidance / L. Brakmo, S. O'Malley, L. Peterson // SIGCOMM 94, ACM. – 1994.

6. Floyd S. NewReno Modification to Fast Recovery Algorithm/ S. Floyd, T. Henderson, A. Gurtov // RFC 3782.

7. Fu C. P. TCP Veno: End-to-End Congestion Control Over Heterogeneous Networks / C. P. Fu. // Ph.D. dissertation, The Chinese Univ. Hong Kong. – Hong Kong, 2001.

8. Casetti C. TCPWestwood: End-to-End Congestion Control for Wired/Wireless Networks / C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi // Wireless Networks. – 2002. – № 8. – P. 467-479.

9. Cainin C. TCP Hybla: a TCP enhancement for heterogeneous networks / Carlo Cainin, Rosario Firrincieli // Int. J. Satell. Commun. Network. – 2004. – № 22. – P. 547-566.

10. Injong Rhee. CUBIC: A New TCP-Friendly High-Speed TCP Variant" / Injong Rhee, Lisong Xu // Proceedings of the Third International Workshop on Protocols for Fast Long-Distance Networks, France. – February 2005.

11. Shao Liu. TCP-Illinois: A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks / Shao Liu, Tamer Basar, R. Srikant // Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, USA. – 2007.