

Communication Networks (0368-3030) / Spring 2011

The Blavatnik School of Computer Science,
Tel-Aviv University

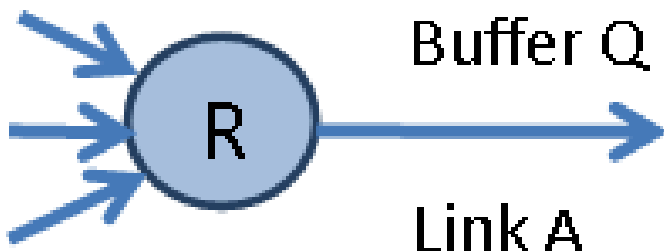
Allon Wagner

A decorative graphic consisting of several horizontal lines of varying lengths and colors (teal, light blue, white) extending from the right side of the slide towards the center.

TCP Recap

שאלה ממבחן (2011/12)

- בשאלה זו נדון בשיטת Random Early Detection (RED) לטיפול בעומס ע"י הנתבים (routers) ברשת.
- נתון נתב R עם מספר כניסות. נניח שכל הלקוחות העוברים דרכו משתמשים ב-TCP Reno. כל הלקוחות שולחים מידע היוצא מ-link A ולכולם יש כל הזמן מידע חדש לשלוח.
- הקצב של הלינק מוגבל כמובן, ולכן יש תור (Buffer) שנסמן Q ובו מאוחסנות פקטות הממתינות לתורן להשלח.
- אם הלינק לא מספיק לשלוח את הפקטות החוצה בקצב שבו הן מצטברות בתור, התור ילך ויגדל עד שיתמלא.



שאלה ממבחן (2011/12)

פעולת הנתב ללא RED

- נניח שהתור מלא. כשמגיעה הפקטה הבאה, הנתב זורק אותה (tail-drop policy). תארו את התהליך שיוביל להקלה בעומס ברשת כתוצאה מכך. התייחסו בתשובתכם לפעולות של TCP אצל ה-host אליו מיועדת הפקטה שנזרקה, ואצל ה-host ששלח אותה
- כשהפקטות הבאות אחרי זו שנזרקה יגיעו ליעד יוצרו duplicate acks שיביאו להפעלת מנגנון ה-congestion control אצל השולח: הקטנת ה-threshold לחצי מ-cwnd בעת ה-loss event ומעבר ל-FR ואח"כ אל CA.
- לחילופין, אם נזרקות הרבה פקטות של השולח ולא יוצרו מספיק dup-acks יקרה אצל השולח timeout והוא יוריד את cwnd ל-1 ואת thresh כמתואר קודם ויעבור ל-SS.

שאלה ממבחן (2011/12)

פעולת הנתב ללא RED

- תוך כמה זמן מרגע שמתחילות להזרק פקטות צפוי הראוטר להרגיש תחילת הקלה בעומס המגיע אליו? (רמז: נסחו את התשובה במונחי RTT).
- לפי הנחות השאלה, מייד אחרי הפקטה שנזרקה ממשיכות להשלח עוד פקטות שעוברות דרך הראוטר. מהרגע שבו הפקטות הללו עוברות דרך R, עד שהן מגיעות ליעד (ויוצרות dup acks), ה – acks חוזרים מהיעד אל המקור, המקור מוריד את הקצב שלו ולראוטר מתחילות להגיע פקטות בקצב נמוך יותר עובר בערך RTT אחד.
- לחילופין, אם אצל השולח קורה timeout הוא גם קורה בערך אחרי RTT אחד (מרגע שהפקטה נשלחה) ולכן השינוי מורגש אצל הראוטר כעבור RTT מהזמן שבו הראוטר זרק את החבילה.

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

RED מנסה לאתר עומס עוד לפני שהוא נוצר ולמנוע אותו מראש. נסמן ב- $Q.Count$ את מספר הפקטות הממתכות כרגע בתור Q . יהיו MAX_THRESH ו- MIN_THRESH שני קבועים שחוסמים את גודל התור.

אלגוריתם RED הוא:

- כשמגיעה פקטה חדשה שרוצה להכנס לתור:

- אם $Q.Count < MIN_THRESH$, קבל את הפקטה לתור.
- אחרת, אם $Q.Count > MAX_THRESH$, זרוק את הפקטה.
- אחרת, זרוק את הפקטה בהסתברות p וקבל אותה לתור בהסתברות $1 - p$.

לשם פשטות, נניח ש- $p \in (0,1)$ הוא קבוע (במציאות זה לא כך).

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

- כיצד RED מסייע להקל על העומס ברשת?
- אבדן הפקטות קורה עוד לפני שמגיעים למצב של קיבולת מלאה. באופן הזה הרשת מאותתת ל – hosts שיש עומס ומאפשרת להם להוריד את הקצב לפני שהעומס אינו מאפשר יותר לרשת לתפקד.

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

- בניח שחלק מהלקוחות שולחים תעבורה שהיא bursty – כלומר, יש פרקי זמן קצרים שבהם הם שולחים כמות גדולה של מידע, ובשאר הזמן הם משדרים בקצב נמוך. מה הבעיה המתעוררת?
- במקרה כזה יכול להיות שנקבל פרץ (burst) של מידע שיעלה אותנו מעבר ל – MIN_THRESH ולכן נזרוק חלק מהפקטות. לאחר מכן הרשת תהיה שקטה לזמן מסויים ובזמן הזה היינו יכולים לרוקן את התור בשקט – מסתבר שזרקנו את המידע לחינם. הבעיה היא בעצם שאומדן רגעי של גודל התור אינו מספק באמת מדד טוב לעומס ברשת בנוכחות bursts.

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

- על מנת להתמודד עם הבעיה הוצע להשתמש במשתנה חדש WA . האלגוריתם החדש הוא:
 - $WA \leftarrow 0$
 - כשמגיעה פקטה חדשה שרוצה להכנס לתור:
 - אם $WA < MIN_THRESH$, קבל את הפקטה לתור.
 - אחרת, אם $WA > MAX_THRESH$, זרוק את הפקטה.
 - אחרת, זרוק את הפקטה בהסתברות p וקבל אותה לתור בהסתברות $1 - p$.
 - $WA \leftarrow (1 - \alpha)WA + \alpha \cdot (Q.Count)$
- כאשר $\alpha \in (0,1)$ הוא קבוע כלשהו, וכמו קודם $p \in (0,1)$ קבוע.

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

- נניח שכרגע התחלנו את האלגוריתם. מגיעות n פקטות. נסמן את גדלי התור בעת הגעתן x_1, \dots, x_n . תנו נוסחה סגורה (כלומר, ללא רקורסיה) לערך של WA . הסבירו כיצד השינוי פותר את הבעיה.
- $WA = \alpha(1 - \alpha)^{n-1}x_1 + \alpha(1 - \alpha)^{n-2}x_2 + \dots + \alpha x_n$
 WA הוא exponentially weighted moving average כפי שראינו בהרצאה עבור TCP RTT estimation. הוא ממצע ומחליק לאורך הזמן את הערך של Q.Count וככה מאפשר לנו להעריך יותר טוב מתי הרשת באמת עמוסה.

שאלה ממבחן (2011/12)

אלגוריתם RED (Random Early Detection)

- האם היה עדיף להחליף את שורה (iv) בשורה:
 $WA \leftarrow \frac{1}{n} ((n - 1) \cdot WA + Q.Count)$? נמקו.
 לא.

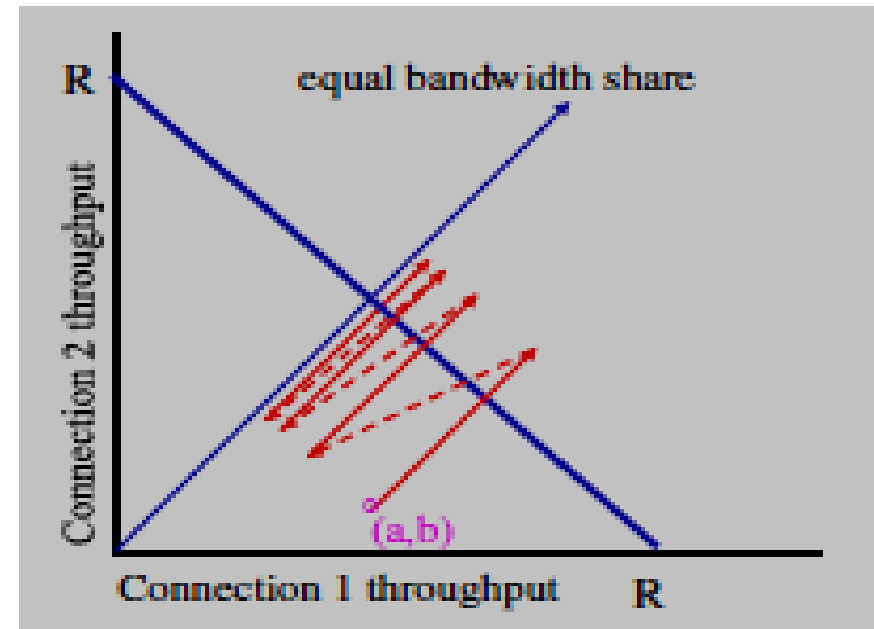
- נשים לב שמשמעות הנוסחה החדשה היא: $WA = \frac{1}{n} \sum_{i=1}^n x_i$
- בנוסחה המקורית הערכים הישנים של x_i דועכים אקספוננציאלית כך שהם מאבדים את משמעותם מהר. משמעות השינוי המוצע היא ש- WA הוא ממוצע חשבוני רגיל של כל הערכים שנמדדו אי פעם x_i . כלומר, ממוצע שנותן משקל שווה לערכים הישנים ולחדשים ולכן משקף פחות טוב את גודל התור הממוצע ברגע הנוכחי.

Exam question (2011/12)

- We learnt that the congestion control of TCP is based on the AIMD (Additive Increase, Multiplicative Decrease) principle. In which phase of TCP's congestion control is this principle implemented precisely?
- In the CA phase when there are only dup acks and not timeouts.
- Every RTT cwnd grows in 1MSS (approximately) → additive increase
- When there is a dup ack, cwnd drops to half its former value (± 3 because of the fast retransmit phase) → multiplicative decrease.

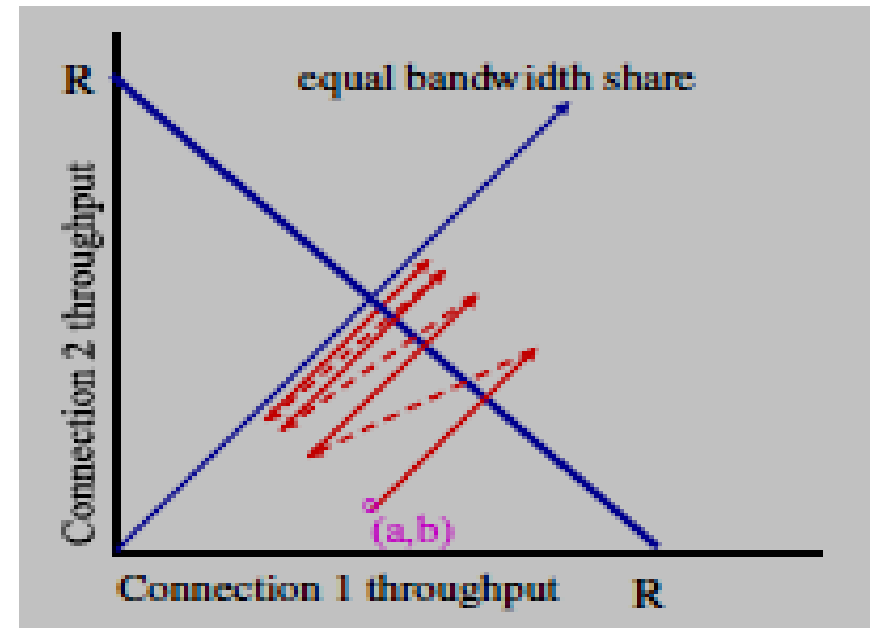
Exam question (2011/12)

- We saw that the AIMD principle allocates network resources optimally.
- Explain what does the figure represents, and how does it supports AIMD fairness
- Each axis: the throughput of one of the clients.
- Left-to-right diagonal: fair allocation. Each client gets the same share



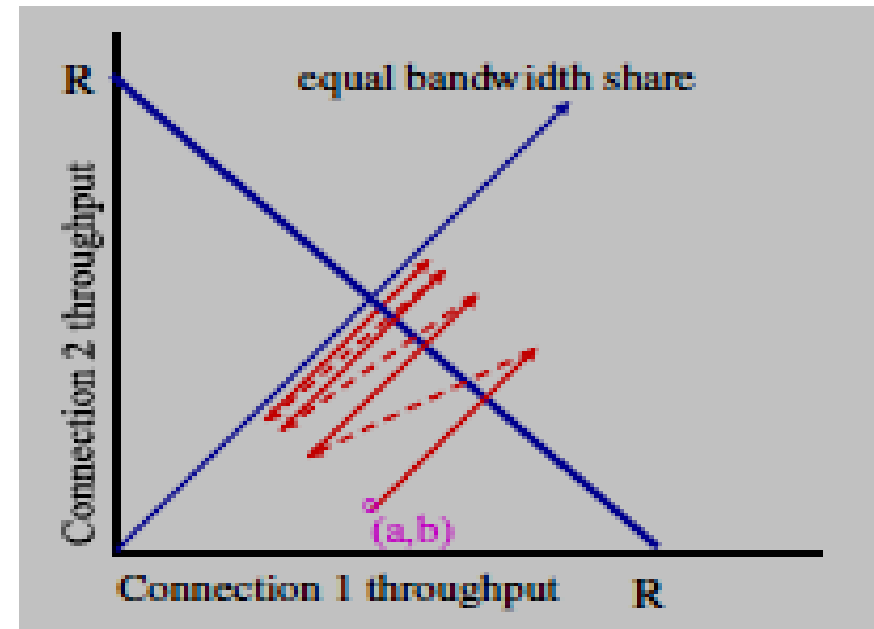
Exam question (2011/12)

- Right-to-left diagonal: efficient allocation. The bandwidth is fully utilized.
- Each axis: the throughput of one of the clients.
- Their intersection: optimal allocation (fair and efficient).
- (a,b): arbitrary starting point for the two connections



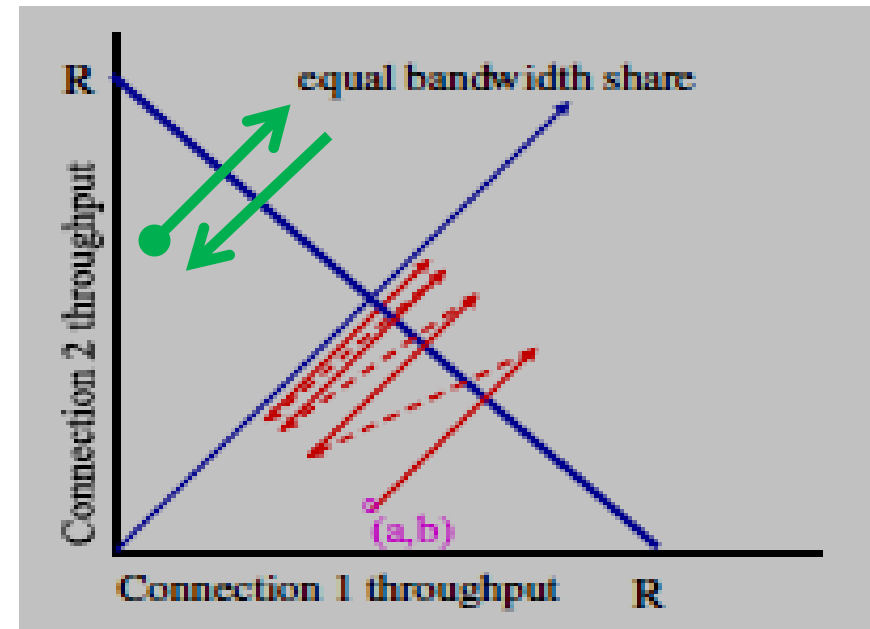
Exam question (2011/12)

- Solid lines: additive increase phases
- Dashed lines: multiplicative decrease events
- We can see the connections converge to the optimal point
- This happens no matter where the starting point (a,b) was.



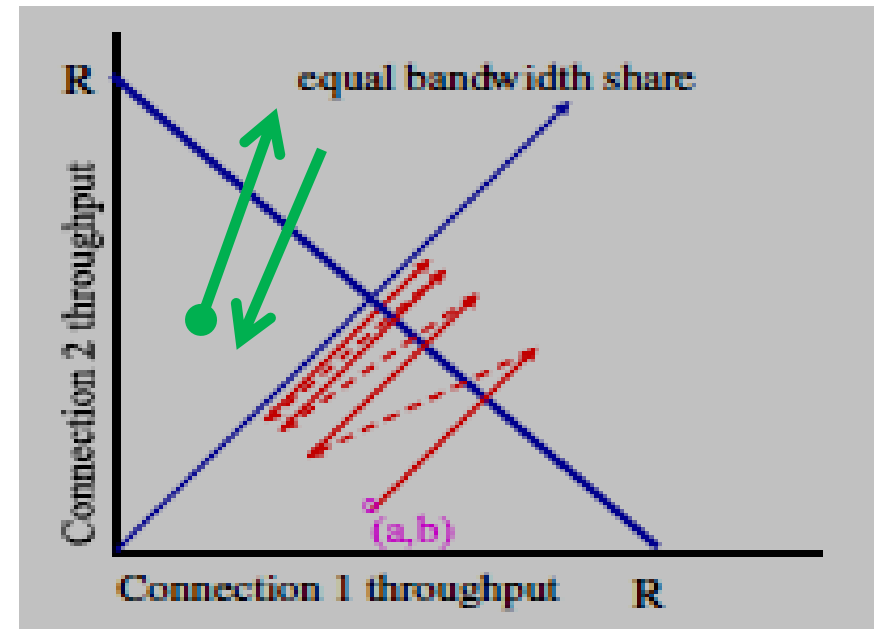
Exam question (2011/12)

- What will happen if on any loss event we decrease cwnd by 2 MSS?
- (i.e., additive decrease instead of multiplicative decrease)
- Oscillation along an additive line (45° angle, i.e., slope = 1 and intersects the y-axis according to the additive factor)



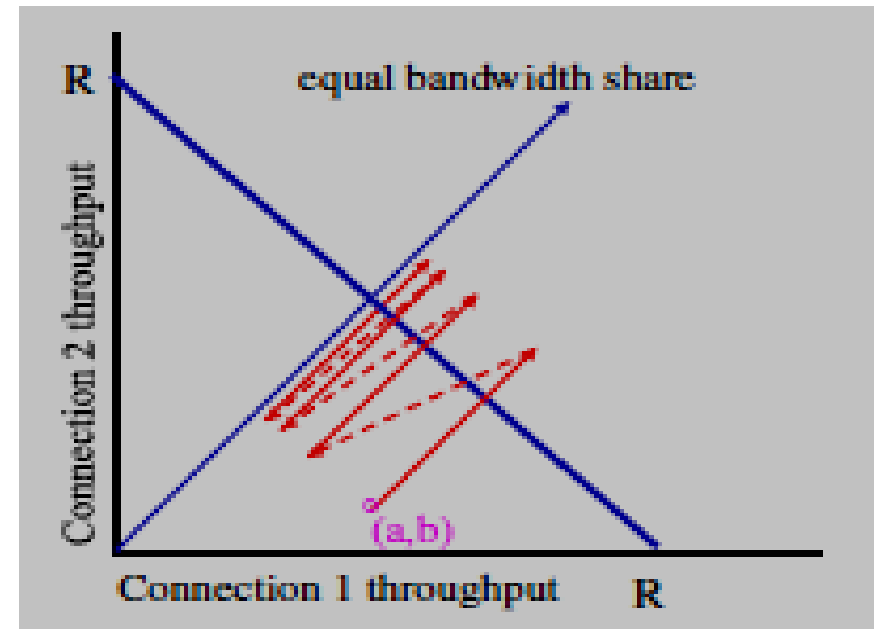
Exam question (2011/12)

- What will happen if in every RTT we increase cwnd by 10%?
- (i.e., multiplicative increase instead of additive increase)
- Oscillation along a multiplicative line (points to the origin, slope determined by the multiplicative factor).



Exam question (2011/12)

- What will happen if we do multiplicative increase additive decrease?!
- Divergence from optimality by exactly the same plot as the original
- Simply traverse the lines backwards.



Exam question (2011/12)

- Does AIMD ensures fairness “in the wild”?
- No. For example:
 - Applications that use UDP do not reduce their sending rate on loss, so they will eventually choke TCP connections.
 - An application that initiates multiple parallel TCP sessions will receive an unfair allocation.
 - TCP does additive increase on every RTT (and not on every global time unit), and so the allocation is biased in favor of short-distance sessions.