~~19/11/19~~ 10/11/20

TCP - Vegas

* נסתכל על השהייה בקבלת ack כאינדיקציה לעומס ברשת.

$$RTT = \underbrace{T_p}_{\text{propagation delay}} + \underbrace{T_q}_{\text{Queue delay}}$$

השהייה בקו — propagation ~~time~~ delay

השהייה בתור — Queue ~~time~~ delay

נרצה לשערך את ההשהייה בקו בלבד.

בעזרת סדרה של פקטות ונקח את ה-RTT המינימלי כדי לשערך את (T_p)

(אם הפקטה לא הצטברה בתור אז T_q קטן מאוד ומתקבל $T_p \approx RTT_{min}$)

* נשערך את תעבורת הקצב הכי טובה בזמן t שניתן להשיג $e(t) = \frac{W(t)}{T_p(t)}$

* נשערך קצב מתקבל בפועל: נסמן חבילה כלשהי בזמן t ונחשב

$$a(t) = \frac{\text{מספר שהתקבלו עד שנקבל ack לחבילה המסומנת}}{\text{RTT לחבילה המסומנת}} = \frac{S}{T_p + T_q}$$

* ניקח שני קבועים $\alpha \leq \beta$

TCP - Vegas עובד כך:

① אם $\alpha \leq e(t) - a(t) \leq \beta$: קצב בפועל קרוב מספיק לקצב אידיאלי תחת $W(t)$. לכן <u>לא נשנה</u> את החלון.

② $e(t) - a(t) > \beta$: קצב בפועל <u>נמוך</u> בהרבה מקצב אידיאלי תחת $W(t)$, כלומר, T_q גדול מידי, <u>נוריד</u> את W ב-1 ב RTT **הבא.**

③ $e(t) - a(t) < \alpha$: קצב בפועל קרוב מידי לקצב אידיאלי, T_q קרוב לאפס, <u>נגדיל</u> את W ב-1

ניתוח מתמטי את הדינמיקה של TCP-Vegas

* נניח $\alpha \triangleq \alpha \approx \beta$ (קרובים לשיווי משקל)

* נניח שעדכון של T_{p_r} מדוייק לסמן $T_{p_r}(t) = T_{p_r}$

$\downarrow$ source/route r

* בשיווי משקל נקבל $e(t) - a(t) = \alpha$

$$e(t) = \frac{\hat{w}_r}{T_{p_r}} \qquad a(t) = \frac{\hat{w}_r}{T_{p_r} + \hat{T}_{q_r}}$$

נציב $\hat{x}_r = \frac{\hat{w}_r}{T_{p_r} + \hat{T}_{q_r}}$ ונקבל

$$\frac{\alpha T_{p_r}}{\hat{x}_r} = \hat{T}_{q_r} \qquad (*)$$

נרצה לנתח השהייה בתור $\hat{T}_{q_r}$

* **משפט ליטל** (תורת התורים)

$$\underbrace{\hat{b}_l}_{\text{אורך התור בלינק } l \text{ בש"מ}} = \underbrace{c_l}_{\text{קיבול הלינק}} \cdot \underbrace{\hat{T}^{(l)}_{q_r}}_{\text{השהייה בתור בלינק } l}$$

זהו קצב הגעת מנות
כשקרובים לקיבול

* תזכורת לניתוח, מכיוון ש- $\hat{T}^{(l)}_{q_r} = \frac{\hat{b}_l}{c_l}$ נקבל שסה"כ

השהייה בתור במסלול r:

$$\hat{T}_{q_r} = \sum_{l: l\in r} \hat{T}^{(l)}_{q_r} = \sum_{l: l\in r} \frac{\hat{b}_l}{c_l}$$

נציב ב (*) ונקבל

$$\frac{\alpha T_{p_r}}{\hat{x}_r} - \underbrace{\sum_{l: l\in r} \frac{\hat{b}_l}{c_l}}_{\text{אילוצי קיבול}} = 0 \qquad (\mathrm{I})$$

* נסתכל על בעיית האופטימיזציה הבאה:

$$\max_{\{x_r\}} \sum_r \alpha_r T_{p_r} \log(x_r)$$

(proportional fairness)

$$\text{s.t.} \quad \sum_{r: l\in r} x_r \le c_l \quad \forall l$$

$$x_r \ge 0 \quad \forall r$$

נרשום את הלגרנז'יאן

$$L(x,\lambda) = \sum_r \alpha_r T_{p_r} \log(x_r) - \sum_l \lambda_l \Big(\sum_{r: l\in r} x_r - c_l\Big)$$

$$\frac{\partial L}{\partial x_r} = \frac{\alpha_r T_{p_r}}{x_r} - \underbrace{\sum_{l: l\in r} \lambda_l}_{\text{כופלי לגרנז'}} = 0 \qquad (\mathrm{II})$$

19/11/19

המשך ניתוח TCP-Vegas

* בשינוי משקל אנו פותרים בעיית NUM עם קריטריון proportional fairness כאשר $\left(\frac{\hat{b}_l}{c_l}\right)$ אלו כופלי לגרנז'
שהם ההשהיות על הלינקים
של אילוצי קיבול על לינקים.

* ניתן לבחור α_r שונים לשקלול קצבי משתמשים.

הקשר בין TCP-Vegas לאלגוריתם הדואלי

תזכורת: באלגוריתם הדואלי קיבלנו בקרת צפיפות:

$$x_r = \underbrace{U_r'^{-1}}_{\text{מחיר}}(q_r) = U_r'^{-1}(\underbrace{cost_r}_{})$$

כופלי לגראנז שקיבלנו

$$= \sum_{l:\, l\in r} \lambda_l = \sum \frac{\hat{b}_l}{c_l} = T_{q_r}$$

sum Lagrange multipliers on route r

זמן השהי"ה.

eaual q_r,
not the best notation..

$$\dot{\lambda}_l = h_l(\underbrace{y_l}_{} - c_l)^+_{\lambda_l} \rightarrow \text{חיובי}$$

$$y_l = \sum_{r:\, l\in r} x_r \quad \text{עומס על לינק } l$$

אם נציב $U_r = \gamma_r \log(x_r)$ (משקל) נקבל

$$U_r'(x_r) = \frac{\gamma_r}{x_r} \Longrightarrow x_r = \frac{\gamma_r}{q_r}\ \left(= \frac{\alpha_r T_{p_r}}{T_{q_r}}\right)$$

מחיר

in our NUM

$$\dot{b}_l = (y_l - c_l)^+_{b_l}$$

שינוי באורך התור, קצב הגעת מנות, קצב עזיבת התור

כעת נשים לב כי b_l הוא אורך התור:

אז
$$\frac{\dot{b}_l}{c_l} = \frac{1}{c_l}(y_l - c_l)^+_{b_l}.$$

$\frac{b_l}{c_l}$ זמני השהי"ה ממוצע של חבילה בלינק l.

כלומר אם נקבע גודל צעד $h_l = \frac{1}{c_l}$ אז פונקציית המחיר של הלינק היא השהיית התור של הלינק.

נשווה את הקצבים המתקבלים להשוות לאלגוריתם הדואלי:

① אם $\underbrace{\frac{w_r}{T_{p_r}}}_{e(t)} - \underbrace{\frac{w_r}{T_{p_r}+T_{q_r}}}_{a(t)} < \alpha \Leftarrow$ נגדיל את W ב-1

נגדיר $x_r = \frac{w_r}{T_{p_r}+T_{q_r}}$ ונגדיר $T_{q_r} = \sum_{l:\, l\in r} \frac{b_l}{c_l} = \sum_{l:\, l\in r} \lambda_l$

ונקבל $x_r \sum_{l:\, l\in r} \lambda_l < \alpha T_{p_r}$

בצורה דומה:

② אם $x_r \sum \lambda_l > \alpha T_{p_r} \Leftarrow$ נקטין את W ב-1

* בצורה קיבלנו שגילית התור מתעדכן של הקו כך:

השהיית התור מתעדכנת על הלינק כך:

$$\dot{\lambda}_l = \frac{1}{c_l}\left(y_l - c_l\right)^+_{\lambda_l}$$

source r update:

$$\dot{w}_r = \left[\frac{1}{T_{p_r}+T_{q_r}} \cdot \mathrm{sign}\left(\alpha T_{p_r} - x_r T_{q_r}\right)\right]^+_{w_r}$$

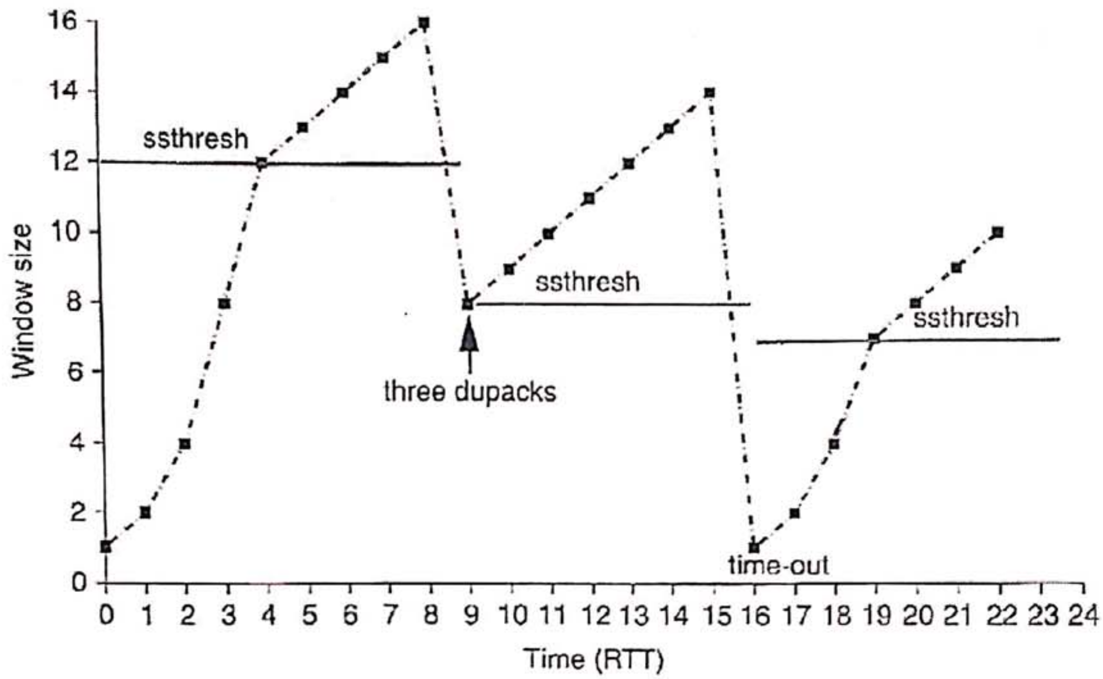
so: x_r=w_r/(T_p_r+T_q_r)

↓

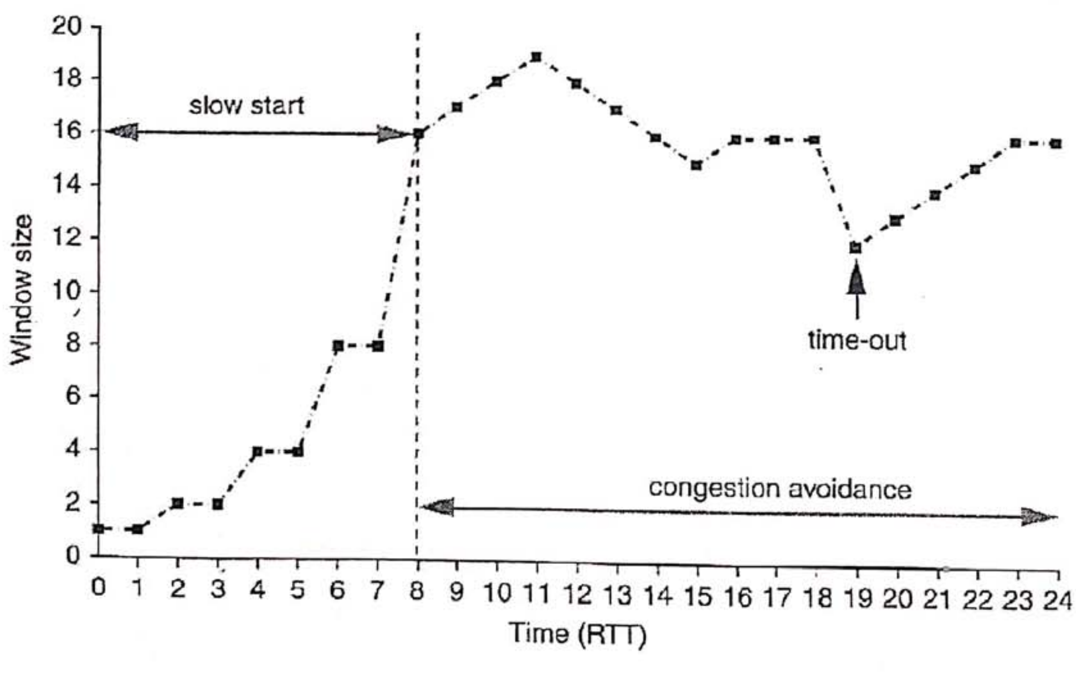
תלוי בתורים שמתעדנים בלינקים

* דומה לאלגוריתם הקודם אך כאן ... מעדכן וקצב
נעשה בצורה עדינה יותר.

TCP Reno



TCP Vegas:



26/11/19

ביצועי רשתות - הרצאה 4

סיכום ביניים

Transport	בקרת עומס
Net	ניתוב
Link (MAC)	
PHY	תקשורת ספרתית

בקרת עומס בשכבת התעבורה

↓

קריטריונים שונים של הוגנות

↓

בעיית NUM (מרוכזת)

↓

אלגוריתמים מבוזרים ל-NUM

↓

מימוש של TCP

↓

שכבת הרשת Net layer

אלגוריתמים לניתוב בשכבת הרשת

* עד כה התייחסנו למסלולים כנתונים. עכשיו נלמד כיצד המסלולים נקבעים בשכבת הרשת.

שקול למינוח בתורת הגרפים:

$G=(V,E)$ — V: vertices, E: edges

* נגדיר "גרף מכוון" $G=(N,L)$ — N: nodes, L: links

* (i,j) לינק מצומת i לצומת j
* $c_{ij}>0$ מחיר ללינק (i,j) (ייתכן $c_{ji} \neq c_{ij}$)
* אלגוריתם לניתוב רוצה למזער מחיר לאורך מסלול

מחיר יכול להיות תלוי בקיבול, השהייה וכו'.

אלגוריתם דייקסטרה (link state routing)

* משמש את פרוטוקול OSPF (open shortest path first)
* ברמת ה-Net יכול להיות אלגוריתם ניהולי מעל OSPF
שנקרא BGP

* כל צומת יודע טופולוגיה והמחירים של כל הלינקים
כל צומת מחשב איטרטיבית מחיר מינימלי ממנו לשאר הצמתים
נראה באמצעות דוגמה בעמוד הבא
^
K **שלא בקבוצה**
* סיבוכיות: בכל איטרציה עוברים על כל הצמתים $O(n^2)$
ניתן להוריד ל- $O(n\log n)$
* זיכרון: כל צומת מחזיק טבלה $O(n)$
ובנוסף ייצוג מלא של טופולוגיית הרשת
* רוחב פס להתחלה: כל צומת צריך מידע על L לינקים
ולכן צריך $O(n\cdot L)$ הודעות **על פני כל הרשת.**

אלגוריתם בלמן-פורד (distance vector routing)

* משמש את פרוטוקול RIP (routing information protocol)
פתרון נפוץ. **זמן התכנסות ארוך יותר אך פשוט יותר למימוש.**
מבוזר, אסינכרוני , תקשורת בין שכנים בלבד
* מחשב איטרטיבית מינימום מחיר ליעד ואת ה-hop הבא.
* משתמש בתכנות דינאמי. ומתבסס על משוואת בלמן-פורד.

$$W^*_{ui} = \min_{j:(u,j)\in \hat{L}} \left(C_{uj} + W^*_{ji} \right)$$

↓
cost between
node u
and its neighbor j

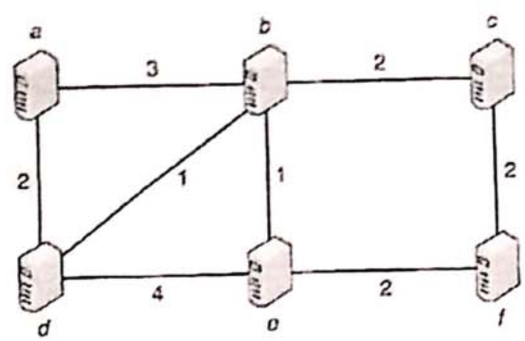
מחיר מינימלי בין
j ל- i

* אם אנחנו יודעים מחירים מינימליים מכל הצמתים (חוץ מ- u)
לצומת i , אז נבחר בצומת j שיתן מחיר מינימלי
$C_{uj} + W^*_{ji}$.

Dijkstra Algorithm at node u:

Input: c_{ij} for all $(i, j) \in \mathcal{L}$.
Set $\mathcal{K} = \{u\}$,
for $i \in \mathcal{N}$ **do**
  Set $\omega_{ui} = c_{ui}$ if $(u, i) \in \mathcal{L}$, i.e., node i is a neighbor of node u, and $\omega_{ui} = \infty$ otherwise.
  Set $p_{ui} = u$ if $(u, i) \in \mathcal{L}$, i.e., node i is a neighbor of node u, and $p_{ui} = -1$ otherwise, where -1 indicates that the previous hop is unknown.
end for
while $\mathcal{K} \neq \mathcal{N}$ **do**
  Find node i^* such that $i^* \in \arg\min_{i \notin \mathcal{K}} \omega_{ui}$. Ties are broken arbitrarily.
  Set $\mathcal{K} = \mathcal{K} \cup \{i^*\}$.
  for $i \notin \mathcal{K}$ **do**
    if $\omega_{ui} > \omega_{ui^*} + c_{i^*i}$ **then**
      Set $\omega_{ui} = \omega_{ui^*} + c_{i^*i}$ and $p_{ui} = i^*$.
    end if
  end for
end while

Example (of a bidirectional graph):



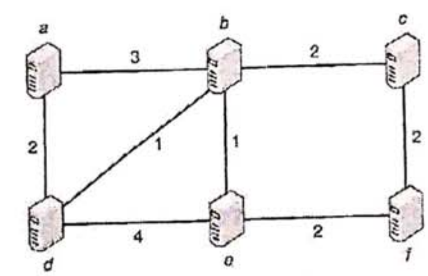
Example of table at node a:

Iteration	$\mathcal{K}$	(ω_{ab}, p_{ab})	(ω_{ac}, p_{ac})	(ω_{ad}, p_{ad})	(ω_{ae}, p_{ae})	(ω_{af}, p_{af})
0	$\{a\}$	$(3, a)$	$(\infty, -1)$	$(2, a)$	$(\infty, -1)$	$(\infty, -1)$
1	$\{a, d\}$	$(3, a)$	$(\infty, -1)$	·	$(6, d)$	$(\infty, -1)$
2	$\{a, d, b\}$	·	$(5, b)$	·	$(4, b)$	$(\infty, -1)$
3	$\{a, d, b, e\}$	·	$(5, b)$	·	·	$(6, e)$
4	$\{a, d, b, e, c\}$	·	·	·	·	$(6, e)$
5	$\{a, d, b, e, c, f\}$	·	·	·	·	·

Bellman-Ford Algorithm at node u:

```
for u ∈ N do
  for i ∈ N \ {u} do
    Set ω_ui = c_ui if (u, i) ∈ L, i.e., node i is a neighbor of node u, and
    ω_ui = ∞ otherwise.
    Set n_ui = i if (u, i) ∈ L, i.e., node i is a neighbor of node u, and
    n_ui = −1 otherwise, where −1 indicates the next hop is unknown.
  end for
end for
while t ≥ 0 do
  for u ∈ N do
    Node u sends out its ω_u = [ω_uj]_{j∈N} to all its neighbors if ω_u was
    updated during iteration t−1.
  end for
  for u ∈ N do
    for i ∈ N \ {u} do
      if ω_ui ≠ min_{j:(u,j)∈L} (c_uj + ω_ji) then
        ω_ui = min_{j:(u,j)∈L} (c_uj + ω_ji)
        n_ui ∈ argmin_{j:(u,j)∈L} (c_uj + ω_ji)
      end if
    end for
  end for
end while
```

Example (of a bidirectional graph):



Example of seaching for minimal path to node c:

	Iteration	(ω_{ac}, n_{ac})	(ω_{bc}, n_{bc})	(ω_{cc}, n_{cc})	(ω_{dc}, n_{dc})	(ω_{ec}, n_{ec})	(ω_{fc}, n_{fc})
Initializing:	0	$(\infty, -1)$	$(\infty, -1)$	$(0, c)$	$(\infty, -1)$	$(\infty, -1)$	$(\infty, -1)$
message passing between neighbors:	1	$(\infty, -1)$	$(2, c)$	$(0, c)$	$(\infty, -1)$	$(\infty, -1)$	$(2, c)$
message passing between neighbors:	2	$(5, b)$	$(2, c)$	$(0, c)$	$(3, b)$	$(3, b)$	$(2, c)$

26/11/19

המשך בלמן-פורד

* האינפורמציה הדרושה: כל צומת u מחזיק שני פרמטרים לגבי כל צומת ברשת (נניח i):

W_{ui} = שערוך עדכני למחיר מינימלי מ-u ל-i

n_{ui} = ה- hop הבא במסלול שמשיג את W_{ui}

distance vector

מידי פעם שכנים מחליפים אינפורמציה ומעדכנים את וקטור המרחק לפי משוואת בלמן.
יתרונות: פשוט למימוש, איטרטיבי, מבוזר, תקשורת בין שכנים בלבד
חסרונות: עדכון צריך לחלחל ברשת להתכנסות, רגישה לשינויים, ניתן להיכנס למעגלים

סיכום ביניים

בקרת זרימה בשכבת התעבורה (TCP)

↓

אלגוריתמים לניתוב בשכבת הרשת (Net)

↓

בקרת גישה בשכבת הלינק (MAC) = Medium Access Control

* גישות קלאסיות **בקצרה**
* גישות מודרניות MAC/PHY **בהרחבה**

בקרת גישה לערוץ MAC

פרוטוקול ALOHA (slotted Aloha)

* N משתמשים, כל משתמש משדר בהסתברות p.

אם 2 **או יותר** משתמשים משדרים ביחד ⇐ התנגשות ושניהם נפסלים

* "ניצולת" - הזמן היחסי בו חבילות מתקבלות בהצלחה (throughput)

נחשב ניצולת:

$$\eta = P_r\left(\text{הסתברות שחבילה מתקבלת בהצלחה}\right)$$

$$= P_r\left(\text{משתמש 1 קיבל בהצלחה} \cup \text{משתמש 2 קיבל בהצלחה} \cup \text{משתמש N קיבל בהצלחה} \cdots\right)$$

מאורעות זרים

$$= \sum_{i=1}^{N} P_r\left(\text{משתמש קיבל בהצלחה}\right) = \sum_{i=1}^{N} p\,(1-p)^{N-1}$$

p – אחד משדר; $(1-p)^{N-1}$ – כל השאר לא משדרים

$$\eta = Np(1-p)^{N-1}$$

נחפש הסתברות שידור שתמקסם ניצולת:

כדי למצוא ניצולת מקסימלית נגזור $\frac{\partial \eta}{\partial p}$ ונמצא max.

$$\frac{\partial \eta}{\partial p} = N(1-p)^{N-1} - N(N-1)p(1-p)^{N-2} \underset{\text{נדרש}}{=} 0 \quad /:(1-p)^{N-2}$$

$$(1-p) - (N-1)p = 0$$

$$\boxed{p^* = \frac{1}{N}}$$

נציב לקבל ניצולת מקסימלית:

$$\eta = \left(1 - \frac{1}{N}\right)^{N-1} \xrightarrow[N \to \infty]{} e^{-1}$$

* משיגים 37% ניצולת מקסימלית.

* היום לא משתמשים ב-Aloha המקורי.

משתמשים ב- Slotted ALOHA כדי ~~למנוע~~ להפחית התנגשויות

<u>יתרונות לשימוש Aloha</u>

* פשוט לשימוש
* טוב בעומסים נמוכים
* השהייה נמוכה

איך נעלה ניצולת? נשתמש בפרוטוקול אחר!

CSMA - Carrier Sensing Multiple Access

CSMA - ומשתמשים ברשת מאזינים, אם משתמש אחד משדר אז השאר יודעים לא לשדר ברגע הזה.

טוב לרשתות עם עומסים גבוהים, אפשר להגיע ל 60% ניצולת ולשלם בהשהייה גבוהה יותר.

26/11/19

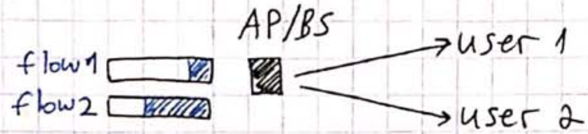
פרוטוקולים מבוססי MAC/PHY

נסתכל על המודל הבא:

משדר יחיד - הרבה משתמשים = מודל Downlink

AP = Access Point

BS = Base Station



רכיב AP שומר באפרים את המידע שצריך להגיע למשתמשים כדי לנהל את קצב שליחת המידע אליהם.

* נמחיש חשיבות של CSI בדוגמה הבאה:

(CSI = Channel state Information)

- נניח ערוצי התנגשות, כלומר בכל יחידת זמן משדרים למשתמש אחד בלבד.
- נניח ערוצים בינאריים ON/OFF עם הסתברות $\frac{1}{2}$ לכל מצב.

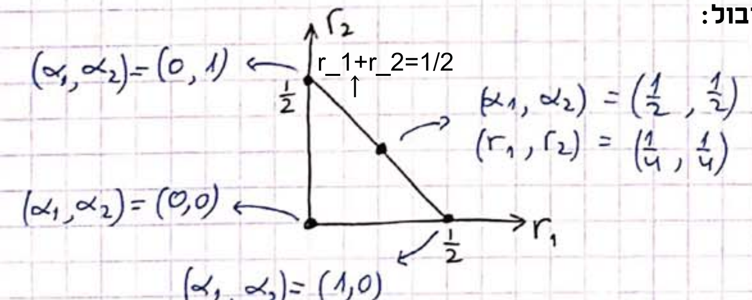
* גישה 1: תזמון שידור לא מבוסס CSI

BS בוחר משתמש i בהסתברות α_i ומשדר לו חבילה $(\alpha_1+\alpha_2 \leq 1)$ נחשב קצב של משתמש i:

$$\Pr\begin{pmatrix}\text{חבילה למשתמש}\\ i \text{ מתקבלת}\\ \text{בהצלחה}\end{pmatrix} = \alpha_i \cdot \Pr\begin{pmatrix}\text{Channel } i\\ \text{is ON}\end{pmatrix} = \frac{\alpha_i}{2}$$

כלומר וקטור הקצבים הוא $(r_1, r_2) = \left(\frac{\alpha_1}{2}, \frac{\alpha_2}{2}\right)$

נשרטט את אזור הקיבול:



* גישה 2: תזמון שידור מבוסס CSI

בפועל ניתן לשדר סיגנל בקרה לשערוך ערוץ

1. נניח שיודעים את מצב הערוץ לפני השידור (ON/OFF)
2. אם רק אחד מהערוצים במצב ON אז ה-BS משדר לו בהסתברות β_i
3. אם שני הערוצים במצב ON BS יבחר אחד מהם בהסתברות α_i

(המשך) תזמון מבוסס CSI

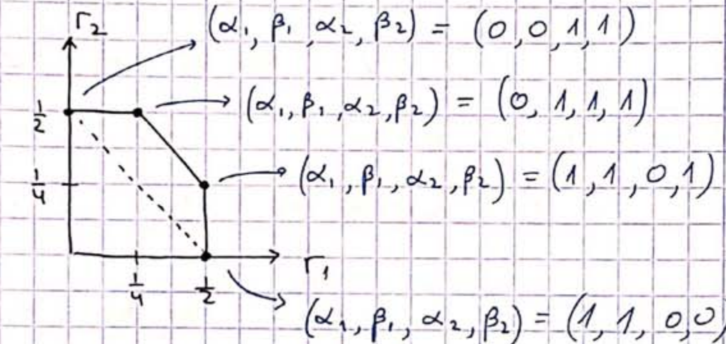
נחשב קצב של משתמש i

$$P_r\begin{pmatrix}\text{פונה למשתמש}\\ i\ \text{מתקבל}\\ \text{בהצלחה}\end{pmatrix} = \beta_i \cdot P_r\begin{pmatrix} i=ON \\ \text{אחרים} \\ \text{ב-}OFF\end{pmatrix} + \alpha_i \cdot P_r\begin{pmatrix}\text{שני}\\ \text{המשתמשים}\\ \text{ב-}ON\end{pmatrix}$$

$$= \beta_i \cdot \frac{1}{4} + \alpha_i \cdot \frac{1}{4} = \frac{\beta_i + \alpha_i}{4}$$

$$(r_1, r_2) = \left(\frac{\beta_1+\alpha_1}{4}, \frac{\beta_2+\alpha_2}{4}\right) \qquad \alpha_1+\alpha_2 \le 1$$

$$\beta_1, \beta_2 \le 1$$



מסקנות

1. ע"י קביעת $\beta_1=\beta_2=1$ ניתן להגדיל קצב של משתמש
אחד ללא פגיעה במשתמש אחר.

2. השיפור: $r_1+r_2=\frac{1}{2}$ ללא CSI
השתפר ל: $r_1+r_2=\frac{3}{4}$ עם CSI
כלומר הגדלנו ניצולת ב- 50%

3. מבחינת קריטריון הגינות השיפור הוא הרבה יותר משמעותי