شبکه‌های کامپیوتری

کسری کاشانی، مرضیه موسوی

پروژه ۲ فاز دوم

۸۱۰۱۰۱۴۹۰

۸۱۰۱۰۱۵۲۶

۱ مقدمه

در شبکه‌های کامپیوتری، پروتکل TCP به‌صورت ذاتی دارای مکانیزم‌های کنترل ازدحام است که باعث می‌شود منابع شبکه به‌صورت نسبتاً منصفانه بین جریان‌ها تقسیم شود. در مقابل، UDP به‌طور پیش‌فرض فاقد هرگونه کنترل نرخ یا واکنش به ازدحام است و می‌تواند باعث اشغال بخش عمده‌ای از پهنای باند و تضعیف شدید عملکرد TCP شود.
هدف این پروژه، بررسی این مسئله و همچنین پیاده‌سازی یک مکانیزم ساده‌ی TCP-Friendly برای جریان UDP و مقایسه‌ی رفتار آن با TCP است.
در این پروژه، سه سناریوی مختلف مورد بررسی قرار گرفته است:

- فقط TCP (Baseline)
- TCP به‌همراه UDP بدون کنترل نرخ
- TCP به‌همراه UDP با کنترل نرخ TFRC-like

در هر سه حالت معیارهای گذردهی (Throughput)، نرخ از دست رفتن بسته (Packet Loss)، تأخیر متوسط (Average Delay) و شاخص انصاف Jain محاسبه و مقایسه شده‌اند.

۲ توپولوژی شبکه و تنظیمات شبیه‌سازی

توپولوژی مورد استفاده شامل دو فرستنده (یکی TCP و یکی UDP)، دو روتر میانی و یک گیرنده است. هر دو جریان TCP و UDP از یک لینک گلوگاه مشترک عبور می‌کنند که بین دو روتر R1 و R2 قرار دارد.
ویژگی‌های لینک‌ها به صورت زیر تنظیم شده است:

- لینک‌های دسترسی: ظرفیت 10Mb و تأخیر 10ms
- لینک گلوگاه (R1–R2): ظرفیت 1.5Mb و تأخیر 20ms

- نوع صف: DropTail با محدودیت صف 50 بسته

هدف از این توپولوژی ایجاد شرایط رقابتی واقعی بین TCP و UDP بر روی یک لینک مشترک است.

۳ سناریوی اول: فقط TCP (Baseline)

در این سناریو تنها یک جریان TCP از فرستنده به گیرنده فعال است و هدف اندازه‌گیری حداکثر گذردهی قابل دستیابی در لینک گلوگاه و بررسی رفتار پایه‌ی TCP است.

۱.۳ نتایج شبیه‌سازی

خروجی اجرای اسکریپت ns-2 برای این سناریو به‌صورت زیر ثبت شده است:

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ ns tcp.tcl
     MODE TCP
TCP throughput: 1.4818297435897434 Mbps
```

شکل ۱: اجرای سناریوی اول با دستور ns

تحلیل فایل Trace نیز نتایج زیر را نشان می‌دهد:

- Throughput: حدود 1.48 Mbps
- Packet Loss: حدود 0.0037%
- Average Delay: حدود 70.6 ms

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ python3 trace_analyze.py traces/tcp.tr --start 0.5 --stop 20

Results of traces/tcp.tr:
   Window: 0.5 .. 20.0 (duration=19.50 sec)

   Flow TCP:
      - Throughput = 1.4763 Mbps
      - Packet Loss = 0.0037% (sent=3473, recv=3460)
      - Average Delay = 70.5588 ms
```

شکل ۲: نتایج سناریوی اول

۲.۳ تحلیل

از آنجا که تنها یک جریان TCP وجود دارد، تقریباً کل ظرفیت لینک گلوگاه در اختیار این جریان قرار می‌گیرد. مقدار گذردهی به ظرفیت لینک (1.5 Mbps) بسیار نزدیک است که نشان می‌دهد TCP به‌درستی از پهنای باند استفاده

می‌کند. نرخ از دست رفتن بسته بسیار ناچیز است که نشان‌دهنده‌ی پایداری صف و نبود ازدحام شدید در شبکه است. همچنین تأخیر متوسط نسبتاً کم است که با توجه به نبود رقابت روی لینک منطقی به نظر می‌رسد.
این سناریو به‌عنوان مبنای مقایسه برای دو حالت بعدی استفاده می‌شود.

۴ سناریوی دوم: TCP به‌همراه UDP بدون کنترل نرخ

در این سناریو یک جریان UDP با نرخ ثابت (CBR) به شبکه اضافه شده است بدون اینکه هیچ مکانیزم کنترلی نسبت به ازدحام داشته باشد. هدف بررسی اثر مخرب UDP بر عملکرد TCP است.

۱.۴ نتایج شبیه‌سازی

خروجی اجرای اسکریپت ns-2 برای این سناریو به‌صورت زیر ثبت شده است:

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ ns udp_plain.tcl
     MODE UDP Plain
TCP throughput: 0.14636307692307693 Mbps
UDP throughput: 1.3567179487179488 Mbps
Jain fairness index: 0.60663917141863388
```

شکل ۳: اجرای سناریوی دوم با دستور ns

نتایج تحلیل فایل Trace:

- Throughput UDP: حدود 1.35 Mbps
- Throughput TCP: حدود 0.15 Mbps
- Packet Loss UDP: حدود 0.036%
- Packet Loss TCP: حدود 0.044%
- Average Delay UDP: حدود 284 ms
- Average Delay TCP: حدود 276 ms
- Jain Fairness Index: حدود 0.61

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ python3 trace_analyze.py traces/udp_plain.tr --start 0.5 --stop 20

Results of traces/udp_plain.tr:
   Window: 0.5 .. 20.0 (duration=19.50 sec)

   Flow UDP:
      - Throughput = 1.3489 Mbps
      - Packet Loss = 0.0366% (sent=3413, recv=3288)
      - Average Delay = 284.4393 ms

   Flow TCP:
      - Throughput = 0.1472 Mbps
      - Packet Loss = 0.0443% (sent=361, recv=345)
      - Average Delay = 276.0193 ms

   Jain Fairness Index = 0.6078
```

شکل ۴: نتایج سناریوی دوم

۲.۴ تحلیل

در این حالت مشاهده می‌شود که UDP تقریباً کل ظرفیت لینک گلوگاه را اشغال کرده است، در حالی که سهم TCP به‌شدت کاهش یافته است. این رفتار به‌خوبی نشان‌دهنده‌ی ماهیت غیرمنصفانه‌ی UDP بدون کنترل نرخ است. از آنجا که UDP به از دست رفتن بسته واکنش نشان نمی‌دهد، همچنان با نرخ بالا ارسال می‌کند و باعث می‌شود TCP که دارای مکانیزم کاهش نرخ است به‌طور مداوم نرخ خود را کاهش دهد.

افزایش شدید تأخیر برای هر دو جریان نیز نشان‌دهنده‌ی پر شدن صف در روتر گلوگاه و افزایش زمان انتظار بسته‌ها است. شاخص Jain برابر حدود 0.61 است که فاصله‌ی زیادی با مقدار ایده‌آل 1 دارد و نشان‌دهنده‌ی توزیع بسیار ناعادلانه‌ی منابع است.

این سناریو دقیقاً همان رفتاری را نشان می‌دهد که در صورت مسئله پروژه به‌عنوان حالت غیرمنصفانه‌ی UDP توصیف شده بود.

۵ سناریوی سوم: TCP به‌همراه UDP با کنترل نرخ TFRC-like

در این سناریو یک مکانیزم ساده‌ی کنترل نرخ مشابه TFRC برای UDP پیاده‌سازی شده است. الگوریتم مورد استفاده از نوع AIMD بوده و شامل دو بخش اصلی است:

- در صورت تشخیص Loss: کاهش نرخ به صورت ضربی $(\text{rate} = 0.85 \times \text{rate})$
- در صورت نبود Loss در یک بازه‌ی زمانی، افزایش آهسته نرخ به اندازه 1% نرخ فعلی

کنترل نرخ در سطح Application انجام شده و هر 0.2 ثانیه نرخ ارسال CBR به‌روزرسانی شده است.

۱.۵ نتایج شبیه‌سازی

خروجی اجرای اسکریپت ns-2 برای این سناریو به‌صورت زیر ثبت شده است:

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ ns udp_tfrc.tcl
     MODE UDP TFRC
TCP throughput: 0.98860307692307692 Mbps
UDP throughput: 0.50051282051282053 Mbps
Jain fairness index: 0.9029880409875769
```

شکل ۵: اجرای سناریوی سوم با دستور ns

نتایج تحلیل Trace:

- Throughput UDP: حدود 0.50 Mbps
- Throughput TCP: حدود 0.99 Mbps
- Packet Loss UDP: حدود 0.015%
- Packet Loss TCP: حدود 0.007%
- Average Delay UDP: حدود 134 ms
- Average Delay TCP: حدود 125 ms
- Jain Fairness Index: حدود 0.90

```
kasra@kasra-VirtualBox:~/Desktop/CA2_P2_NS2$ python3 trace_analyze.py traces/udp_tfrc.tr --start 0.5 --stop 20

Results of traces/udp_tfrc.tr:
   Window: 0.5 .. 20.0 (duration=19.50 sec)

   Flow UDP:
      - Throughput = 0.4964 Mbps
      - Packet Loss = 0.0155% (sent=1229, recv=1210)
      - Average Delay = 134.1708 ms

   Flow TCP:
      - Throughput = 0.9847 Mbps
      - Packet Loss = 0.0069% (sent=2324, recv=2308)
      - Average Delay = 125.3189 ms

   Jain Fairness Index = 0.9020
```

شکل ۶: نتایج سناریوی سوم

۲.۵ تحلیل

در این سناریو رفتار UDP به‌صورت قابل توجهی به TCP نزدیک شده است. به‌جای اشغال کامل لینک، UDP بخشی از پهنای باند را به TCP واگذار می‌کند و تقسیم منابع به‌مراتب منصفانه‌تر انجام می‌شود. شاخص Jain حدود 0.9 است که بسیار نزدیک به مقدار ایده‌آل 1 می‌باشد و نشان می‌دهد مکانیزم TFRC-like موفق شده است رفتار TCP-friendly ایجاد کند.

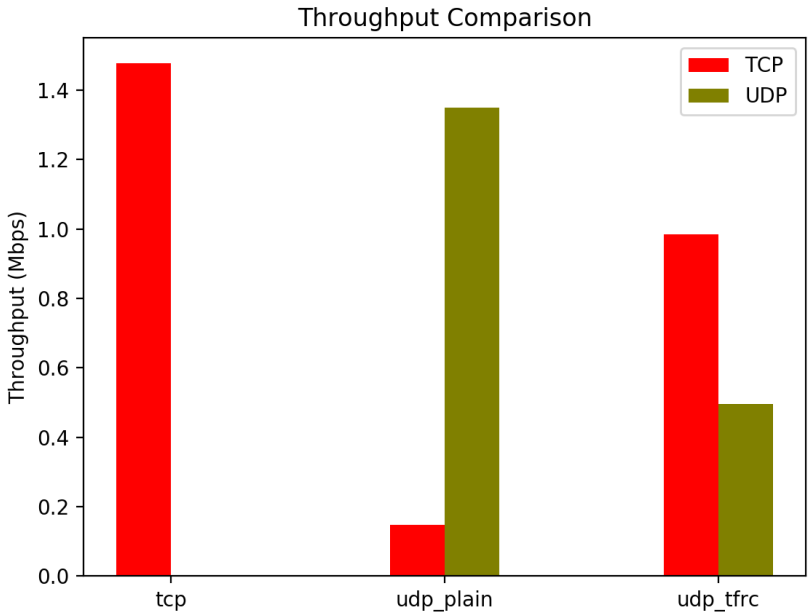
همچنین کاهش محسوس تأخیر و نرخ از دست رفتن بسته‌ها نسبت به سناریوی UDP بدون کنترل، نشان‌دهنده‌ی کاهش ازدحام در لینک گلوگاه است. این نتیجه کاملاً با منطق الگوریتم AIMD سازگار است، زیرا در زمان مشاهده‌ی Loss نرخ ارسال کاهش می‌یابد و اجازه داده می‌شود صف تخلیه شود.
این رفتار نشان می‌دهد که حتی یک الگوریتم ساده‌ی کنترل نرخ در سطح Application نیز می‌تواند به‌طور مؤثری عدالت در استفاده از منابع شبکه را بهبود دهد.

۶ مقایسه نهایی سناریوها

برای مقایسه‌ی نهایی، چهار نمودار شامل Throughput، Packet Loss، Average Delay و Jain Fairness رسم شده است.

۱.۶ تحلیل Throughput

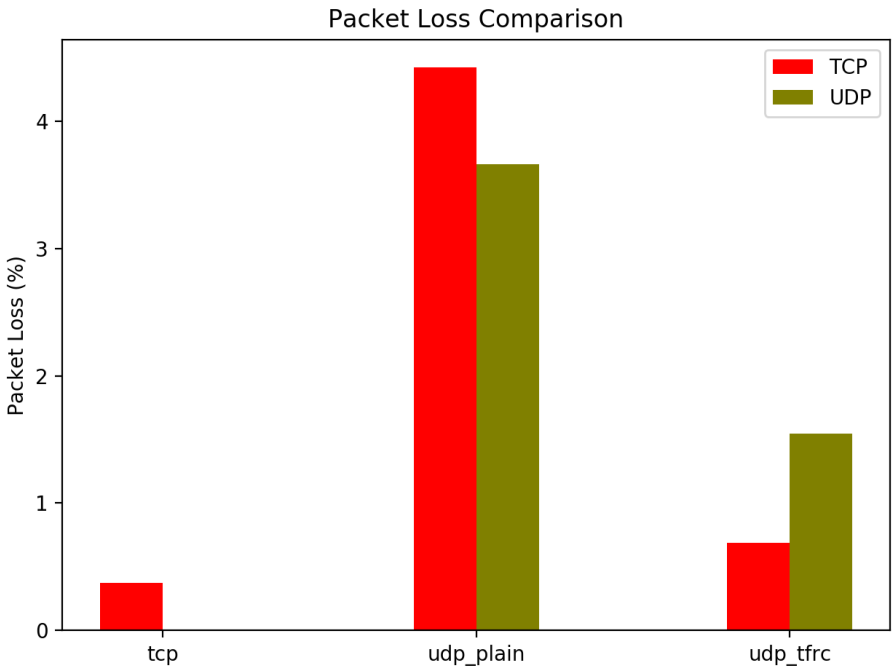
در حالت فقط TCP، کل ظرفیت لینک در اختیار TCP است. در حالت UDP بدون کنترل، UDP تقریباً تمام ظرفیت لینک را اشغال می‌کند و TCP دچار starvation می‌شود. در حالت TFRC-like، ظرفیت لینک بین دو جریان تقسیم شده و TCP همچنان سهم قابل توجهی دریافت می‌کند.



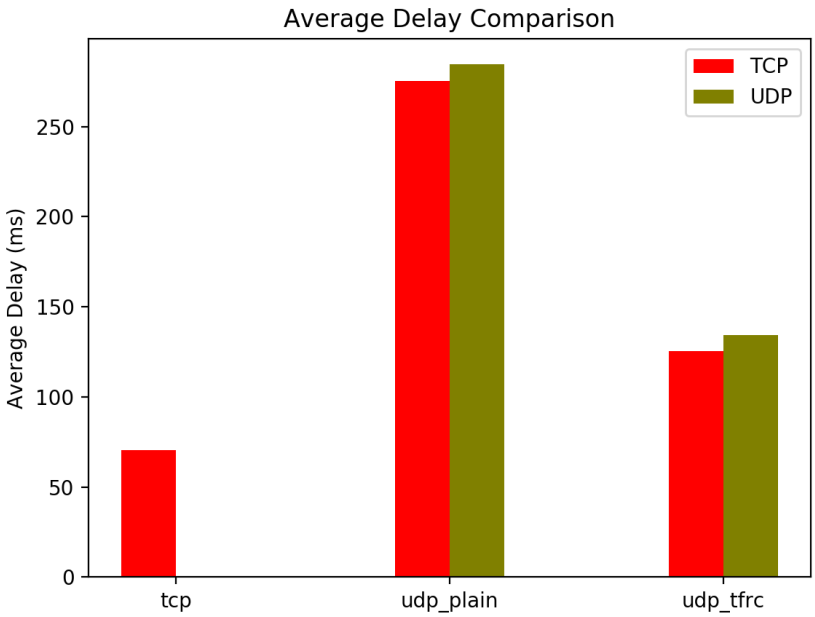
شکل ۷: مقایسه‌ی throughput هر سه سناریو

۲.۶ تحلیل Packet Loss و Delay

در سناریوی UDP بدون کنترل، صف لینک گلوگاه دائماً پر شده و باعث افزایش تأخیر و Loss می‌شود. در مقابل، در سناریوی TFRC-like، با کاهش نرخ هنگام ازدحام، فشار روی صف کمتر شده و هم Delay و هم Loss کاهش می‌یابد.



شکل ۸: مقایسه‌ی packet loss هر سه سناریو

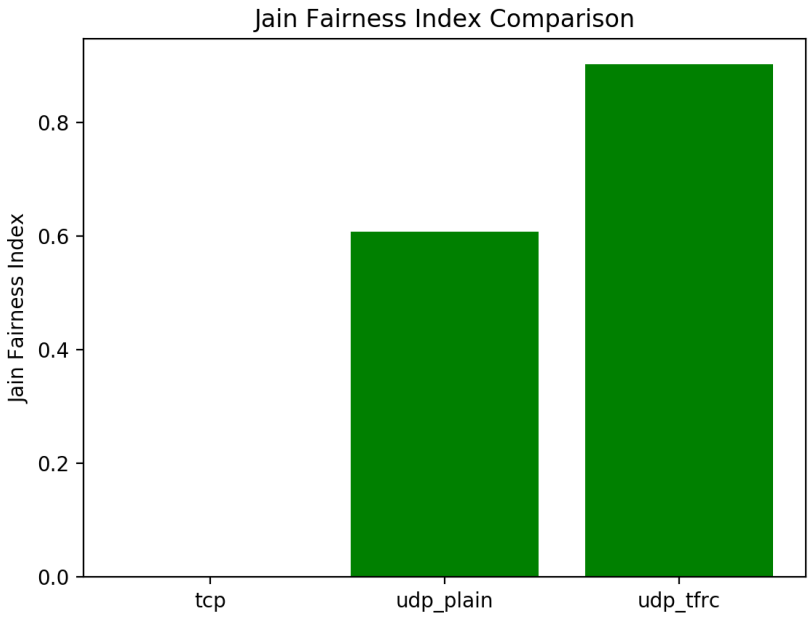


شکل ۹: مقایسه‌ی average delay هر سه سناریو

۳.۶ تحلیل Fairness

شاخص Jain در سه سناریو به ترتیب:

- فقط TCP: عملاً مفهوم fairness مطرح نیست چون تنها یک جریان وجود دارد.
- UDP بدون کنترل: حدود 0.6 (بسیار ناعادلانه)
- UDP با TFRC-like: حدود 0.9 (نزدیک به حالت منصفانه)



شکل ۱۰: مقایسه‌ی Jain fairness index هر سه سناریو

این نتایج به‌وضوح نشان می‌دهد که الگوریتم کاهش-افزایش به سبک TCP باعث می‌شود UDP رفتاری سازگار با TCP داشته باشد و منابع شبکه به‌صورت منصفانه‌تری تقسیم شوند.

۷ پاسخ به سؤال TCP-Friendliness

هدف این بخش نشان دادن این موضوع است که چرا و چگونه الگوریتم ساده‌ی کاهش-افزایش (AIMD) باعث می‌شود جریان UDP رفتاری مشابه TCP از نظر استفاده‌ی منصفانه از منابع شبکه داشته باشد.

۱.۷ رفتار UDP بدون کنترل نرخ

در حالت UDP بدون کنترل، نرخ ارسال مستقل از وضعیت شبکه باقی می‌ماند. حتی در صورت بروز ازدحام و افزایش Loss، UDP هیچ واکنشی نشان نمی‌دهد و همچنان با نرخ ثابت ارسال می‌کند. در نتیجه، صف لینک گلوگاه به‌سرعت پر می‌شود و TCP که دارای مکانیزم کاهش نرخ است، مجبور به کاهش شدید پنجره ازدحام خود می‌شود. این مسئله منجر به پدیده‌ی TCP starvation می‌شود که در آن TCP سهم بسیار کمی از پهنای باند دریافت می‌کند.
نتایج سناریوی UDP بدون کنترل این رفتار را تأیید می‌کند:

- سهم UDP از پهنای باند بسیار بیشتر از TCP است.

- شاخص Jain حدود 0.6 است که نشان‌دهنده‌ی توزیع ناعادلانه‌ی منابع است.
- تأخیر و Loss برای هر دو جریان بالا است که نشان‌دهنده‌ی ازدحام پایدار در صف است.

بنابراین در این حالت UDP به‌هیچ‌وجه TCP-friendly نیست.

۲.۷ منطق الگوریتم کاهش-افزایش (AIMD)

الگوریتم استفاده‌شده برای UDP شامل دو قاعده‌ی اصلی است:

- **Multiplicative Decrease**: در صورت مشاهده‌ی Loss، نرخ ارسال به‌صورت ضربی کاهش می‌یابد.
- **Additive Increase**: در صورت نبود Loss در یک بازه‌ی زمانی، نرخ ارسال به‌صورت آهسته افزایش می‌یابد.

این دقیقاً همان منطق کلی کنترل ازدحام TCP است، با این تفاوت که به‌جای پنجره ازدحام، مستقیماً نرخ ارسال تنظیم می‌شود. کاهش ضربی باعث می‌شود هنگام ازدحام فشار روی شبکه سریعاً کم شود، در حالی که افزایش تدریجی باعث می‌شود رقابت بین جریان‌ها به‌صورت پایدار و تدریجی شکل بگیرد.

۳.۷ همگرایی رفتار UDP و TCP روی لینک گلوگاه

در لینک گلوگاه، زمانی که مجموع نرخ ارسال از ظرفیت لینک بیشتر می‌شود، صف پر شده و Loss رخ می‌دهد. در این لحظه:

- TCP پنجره ازدحام خود را کاهش می‌دهد.
- UDP با الگوریتم TFRC-like نیز نرخ ارسال خود را کاهش می‌دهد.

در نتیجه هر دو جریان به ازدحام واکنش مشابهی نشان می‌دهند و هیچ‌کدام به‌صورت پایدار بر دیگری غلبه نمی‌کند. پس از کاهش نرخ، هر دو جریان دوباره به‌صورت آهسته نرخ خود را افزایش می‌دهند تا به نقطه‌ی تعادل جدید برسند. این چرخه‌ی تکرارشونده‌ی افزایش تدریجی و کاهش سریع باعث می‌شود هر دو جریان به‌طور طبیعی به سمت تقسیم منصفانه‌ی ظرفیت لینک همگرا شوند.

۴.۷ شواهد تجربی از نتایج شبیه‌سازی

در نتایج سناریوی UDP با کنترل TFRC-like مشاهده شد که:

- سهم TCP و UDP از پهنای باند به یکدیگر نزدیک شده است.
- شاخص Jain حدود 0.9 شده است که بسیار نزدیک به مقدار ایده‌آل 1 است.
- میزان Delay و Packet Loss نسبت به حالت UDP بدون کنترل به‌طور محسوسی کاهش یافته است.

این نتایج نشان می‌دهد که UDP دیگر رفتار تهاجمی ندارد و همانند TCP به شرایط ازدحام واکنش نشان می‌دهد. در نتیجه منابع شبکه به‌صورت پایدارتر و منصفانه‌تر استفاده می‌شوند.

۵.۷ نتیجه‌گیری: چرا این UDP، TCP-friendly است؟

یک جریان زمانی TCP-friendly محسوب می‌شود که:

- در زمان ازدحام، سهم خود از پهنای باند را کاهش دهد.
- اجازه دهد سایر جریان‌ها نیز از ظرفیت لینک استفاده کنند.
- در شرایط بدون ازدحام، به‌صورت تدریجی نرخ خود را افزایش دهد.

الگوریتم کاهش-افزایش پیاده‌سازی‌شده برای UDP دقیقاً این سه ویژگی را فراهم می‌کند. بنابراین اگرچه UDP در سطح پروتکل همچنان بدون کنترل ازدحام است، اما در سطح Application رفتاری کاملاً سازگار با TCP پیدا می‌کند و می‌توان آن را TCP-friendly در نظر گرفت.
این موضوع نشان می‌دهد که TCP-friendliness الزاماً نیازمند پیاده‌سازی در سطح Transport نیست و حتی در سطح Application نیز می‌توان رفتار سازگار با ازدحام ایجاد کرد، هرچند پیاده‌سازی‌های استاندارد در سطح پروتکل دقت و پایداری بیشتری دارند.

۸ فرضیات و تصمیمات پیاده‌سازی

در این پروژه چند فرض و تصمیم پیاده‌سازی اتخاذ شده است که مستقیماً در صورت مسئله مشخص نشده بودند:

- کنترل نرخ UDP در سطح Application و با تغییر interval ارسال CBR انجام شد.
- بازه‌ی کنترل نرخ برابر با 0.2 ثانیه در نظر گرفته شد.
- Loss با استفاده از Agent/LossMonitor و شمارش بسته‌های از دست‌رفته تشخیص داده شد.
- افزایش نرخ به صورت درصدی از نرخ فعلی (1%) انجام شد که منجر به افزایش نمایی آهسته می‌شود.

این انتخاب‌ها باعث سادگی پیاده‌سازی شده و در عین حال رفتار کیفی مشابه TCP را ایجاد کرده‌اند، هرچند که پیاده‌سازی کامل TFRC در سطح پروتکل پیچیده‌تر است.

۹ جمع‌بندی نهایی

نتایج این پروژه به‌روشنی نشان می‌دهد که UDP بدون کنترل نرخ می‌تواند باعث تخریب شدید عملکرد TCP در لینک‌های مشترک شود. با این حال، حتی یک مکانیزم ساده‌ی کنترل نرخ به سبک AIMD می‌تواند رفتار UDP را TCP-friendly کرده و منجر به افزایش fairness، کاهش delay و کاهش packet loss شود.
این نتایج اهمیت طراحی پروتکل‌های سازگار با ازدحام را در شبکه‌های اشتراکی نشان می‌دهد و تأکید می‌کند که کاربردهای real-time مبتنی بر UDP نیز باید مکانیزم‌هایی برای تطبیق با شرایط شبکه داشته باشند.