

Proses Video Streaming Dengan Protokol SRT: Analisis Parameter yang Dapat Mempengaruhi Kualitas Pengiriman Data

Dimas Tri Handika, Andy Haryoko

dimastrihandika@gmail.com, andyharyoko@gmail.com

Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Terbuka

ABSTRAK

Proses video *streaming* merupakan aspek krusial dalam distribusi konten multimedia secara *real-time*. Dalam kaitannya, protokol *Source Reliable Transport* (SRT) telah muncul sebagai solusi yang menjanjikan untuk meningkatkan keandalan dan efisiensi pengiriman data. Penelitian ini bertujuan untuk mengeksplorasi dan menganalisis proses video *streaming* menggunakan protokol SRT, dengan fokus pada aspek-aspek kritis seperti latensi, stabilitas dan throughput. Penelitian ini membahas implementasi SRT pada proses video *streaming* dengan fokus pada evaluasi kinerja protokol tersebut. Implementasi dilakukan menggunakan OBS Studio sebagai Encoder dan Titan Edge sebagai Decoder SRT serta pengujian praktis terkait parameter *latency* dan *maxbw* pada SRT untuk pengiriman data yang optimal. Hasil penelitian diharapkan dapat memberikan pandangan yang mendalam tentang potensi SRT dalam mendukung pengalaman pengguna yang lebih baik dalam video *streaming*.

Kata kunci: *Source Reliable Transport, SRT, Live Streaming*

PENDAHULUAN

Kemajuan teknologi informasi di era modern saat ini semakin maju. Perkembangan teknologi kian memudahkan manusia dalam menjalani aktivitas dan rutinitas sehari-hari, tidak terkecuali dalam hal hiburan. Saat ini, berbagai macam hiburan berkualitas sudah hadir di tengah-tengah masyarakat baik melalui perangkat komputer atau hanya melalui gawai pintar. Salah satu layanan hiburan berbasis teknologi yang kian populer saat ini adalah *streaming* video online, seperti Twitch, YouTube, Vidio dan platform *streaming* lainnya. Dengan *streaming*, masyarakat luas dapat mengakses ribuan jam tayangan film *box office*, serial TV populer, video blog, podcast, gaming, maupun konten kreatif lainnya dari seluruh penjuru dunia. Kemudahan akses konten digital inilah yang membuat layanan *streaming* kian mendominasi pasar industri hiburan saat ini.

Live streaming yaitu sebuah teknologi pengiriman data video atau audio yang dikompresi secara online dan ditampilkan secara *real-time* atau pantas permintaan (Faradiba & Syarifudin, 2021). *Live streaming* telah menjadi fenomena yang mendominasi dunia digital, mengubah cara masyarakat berinteraksi dan mengonsumsi konten secara drastis. Kemajuan teknologi dan konektivitas internet yang semakin baik telah memberikan peluang besar untuk menyampaikan konten secara langsung kepada masyarakat di seluruh dunia. Pentingnya protokol *live streaming* juga tercermin dalam peranannya dalam mendukung industri kreatif. Pembuat

konten, baik itu individu maupun perusahaan, dapat dengan mudah menyampaikan karya-karya mereka kepada *masyarakat* luas, memotong batasan geografis dan menghadirkan kolaborasi yang lebih erat antara produsen dan konsumen.

Perkembangan teknologi, peningkatan kecepatan internet dan perubahan perilaku konsumen telah menjadi pendorong utama di balik lonjakan permintaan untuk layanan *streaming*. Keterjangkauan perangkat pintar dan konektivitas yang semakin mudah membuat masyarakat global semakin terhubung dengan dunia digital. Fenomena ini memunculkan tantangan teknis yang memerlukan solusi efisien dan andal, salah satunya melalui pengembangan protokol *Source Reliable Transport* (SRT).

Peningkatan popularitas dan permintaan layanan *streaming* ini berbanding lurus dengan permintaan akan konten video digital berkualitas tinggi, terutama untuk keperluan *streaming* langsung atau *live streaming*. Hal ini menuntut solusi *transportasi* data yang dapat mengatasi tantangan seperti kehilangan paket dan variasi jaringan (Bienik, et al., 2023).

Kehilangan paket dalam pengiriman data dapat mengakibatkan hilangnya informasi yang penting dan merugikan efisiensi komunikasi. Fenomena ini dapat terjadi karena berbagai faktor, mulai dari kegagalan perangkat keras hingga kondisi jaringan yang tidak stabil. Di sisi lain, variasi jaringan menciptakan tantangan lebih lanjut, di mana kecepatan dan kualitas pengiriman data dapat bervariasi secara signifikan tergantung pada kondisi jaringan yang berubah-ubah. Dalam upaya untuk mengatasi tantangan-tantangan tersebut, penelitian dan pengembangan solusi *transportasi* data yang inovatif menjadi semakin mendesak. Solusi yang efektif harus mampu menangani kehilangan paket dengan mengimplementasikan mekanisme pemulihan yang cepat dan dapat diandalkan.

Meskipun protokol *streaming* melalui TCP seperti RTMP mampu pada skala kecil, pertumbuhan eksponensial pasar *streaming* menimbulkan kebutuhan akan solusi yang lebih andal. Dalam konteks ini, Protokol SRT menyajikan inovasi dengan menggabungkan keandalan TCP dan efisiensi UDP. Meskipun SRT telah diterapkan dalam berbagai konteks, penelitian yang merinci proses implementasi dan dampak penerapannya secara khusus dalam proses *streaming* video masih terbatas.

Oleh karena itu, penelitian ini bertujuan untuk menginvestigasi dan menganalisis kinerja protokol SRT dalam konteks proses video *streaming*. Secara khusus, penelitian ini bertujuan untuk menganalisa bagaimana kinerja protokol SRT dalam hal keandalan, latensi dan efisiensi pengiriman data, dan untuk mengetahui parameter dalam implementasi protokol SRT apa yang paling memberikan pengaruh signifikan terhadap stabilitas dan latensi dalam kualitas pengiriman data. Peneliti berharap agar penelitian ini dapat bermanfaat untuk memberikan

wawasan mendalam tentang potensi SRT sebagai solusi yang andal dalam memenuhi kebutuhan video *streaming* modern serta dapat memberikan kontribusi dalam pemahaman lebih lanjut tentang potensi Protokol SRT bagi pengembang aplikasi, penyedia layanan *streaming*, dan peneliti dalam mengoptimalkan protokol *transport* untuk pengiriman data video *real-time*.

KERANGKA PIKIR

UDT (UDP Based Data Transfer Protokol)

Protokol UDT atau UDP-based Data Transfer Protocol, merupakan suatu solusi inovatif untuk pengiriman data *real-time*, khususnya dalam konteks seperti video *streaming*. Didesain untuk mengatasi keterbatasan yang dimiliki oleh protokol TCP, UDT menawarkan karakteristik utama yang membuatnya sesuai untuk aplikasi yang memerlukan keseimbangan antara kecepatan dan keandalan. Salah satu fitur utamanya adalah mekanisme pengendalian aliran sendiri, yang memungkinkan penanganan masalah kehilangan paket dan *Delay* tanpa bergantung pada pengendalian aliran TCP. Selain itu, UDT mendukung pengiriman data *real-time* dan *near real-time*, memenuhi kebutuhan aplikasi seperti video *streaming*, video *conference*, dan multimedia interaktif.

Protokol ini juga mampu menangani pengiriman data besar dengan efisien melalui fitur pengaturan ukuran paket yang dapat disesuaikan. Dengan adanya mekanisme pengendalian aliran berbasis jendela dan *adaptive timeout*, UDT dapat menyesuaikan kecepatan pengiriman data berdasarkan kondisi jaringan yang berubah. Pengguna juga memiliki kendali penuh terhadap parameter kualitas layanan (QoS), seperti *latency*, *throughput*, dan keandalan, sesuai dengan kebutuhan aplikasi.

Meskipun UDT tidak menyediakan enkripsi data secara langsung, keamanan dapat ditingkatkan dengan menggunakan lapisan keamanan tambahan seperti TLS atau SRTP. Terdapat berbagai implementasi pustaka UDT yang tersedia untuk berbagai platform, memudahkan pengembang untuk mengintegrasikan protokol ini ke dalam aplikasi mereka. Namun, penting untuk diingat bahwa kinerja UDT sangat tergantung pada kondisi jaringan, sehingga pemahaman mendalam tentang karakteristik jaringan tempat aplikasi dijalankan menjadi kunci dalam mencapai kinerja maksimal. Dengan demikian, UDT memberikan solusi yang efisien dan ketahanan terhadap *Delay* untuk pengiriman data video *real-time*, dengan keberhasilan implementasi tergantung pada pengaturan parameter yang cermat sesuai dengan kebutuhan spesifik aplikasi.

SRT (*Source Reliable Transport*)

Transmisi video *streaming* dengan latensi rendah melalui jaringan IP yang dapat diandalkan, biasanya di lingkungan lokal, umumnya menggunakan format MPEG-TS. Aliran dapat berupa *unicast* atau *multicast* dengan menggunakan protokol UDP/RTP. Menyamakan tingkat latensi di antara lokasi yang berbeda, seperti kota, negara, atau bahkan benua, merupakan suatu tantangan. Penggunaan satelit atau jaringan MPLS khusus dapat memungkinkan hal ini, tetapi solusi ini tergolong mahal. Meskipun konektivitas Internet publik lebih terjangkau, diperlukan *overhead bandwidth* yang signifikan untuk mencapai tingkat pemulihan kehilangan paket yang dibutuhkan. Pengenalan transmisi ulang paket selektif (UDP yang dapat diandalkan) membantu mengatasi batasan ini.

SRT, yang berasal dari protokol Transfer Data (UDT) berbasis UDP, merupakan protokol tingkat pengguna yang tetap mempertahankan konsep dan mekanisme dasar sambil mengenalkan sejumlah perbaikan dan peningkatan. Perubahan tersebut mencakup modifikasi pada paket kontrol, peningkatan kontrol aliran untuk mengelola *streaming* langsung, perbaikan kontrol kemacetan, dan mekanisme untuk mengenkripsi paket. SRT dirancang sebagai protokol *transport* untuk menjamin keamanan dan keandalan pengiriman data melalui jaringan yang tidak dapat diprediksi, seperti Internet. SRT sangat ideal untuk *streaming* video dengan latensi rendah. Protokol ini meningkatkan efisiensi *bandwidth* jika dibandingkan dengan RTMP, memungkinkan kontribusi *bitrate* yang lebih tinggi melalui koneksi jarak jauh.

Saat mengirimkan paket dari sumber ke tujuan, SRT secara *real-time* mendeteksi dan menyesuaikan diri dengan kondisi jaringan antara kedua titik akhir. Hal ini membantu mengatasi fluktuasi *jitter* dan *bandwidth* yang mungkin terjadi pada jaringan. Mekanisme pemulihan kesalahan SRT berperan dalam mengurangi kehilangan paket yang umumnya terjadi dalam koneksi Internet. Untuk mencapai *streaming* dengan latensi rendah, SRT perlu mengatasi tantangan waktu yang timbul saat aliran data melintasi Internet publik. SRT dilengkapi dengan mekanisme yang memastikan agar latensi *end-to-end* tetap konsisten, mengoptimalkan sinyal di sisi penerima, dan mengurangi kebutuhan *buffering*.

Seperti TCP, SRT mengadopsi model *listener/caller*, memungkinkan arus data dua arah tanpa bergantung pada inisiasi koneksi. Protokol ini mendukung multiplexing internal, memungkinkan beberapa koneksi SRT untuk berbagi port UDP yang sama, dengan kontrol akses yang memudahkan identifikasi *caller* di sisi *listener*.

Menurut Arnao dan Buttmer (2022), Kecepatan transmisi video dapat bervariasi berdasarkan pada berbagai faktor, misalnya, kemacetan jaringan dan/atau kualitas komunikasi.

Variasi ini bisa menyebabkan paket video dalam jaringan protokol internet (IP) diterima dengan penundaan yang bervariasi antara satu atau lebih paket video.

Dilansir dari halaman haivision.com yang merupakan pengembang protokol SRT, untuk menentukan *latency value* SRT dapat menggunakan formula $SRT\ latency = RTT\ multiplier * RTT$. RTT dapat dilihat dengan cara melakukan ping ke server tujuan, namun cara ini kurang efektif karena beban ping sangat kecil, akan lebih relevan jika melakukan simulasi pengiriman data terlebih dahulu setidaknya selama satu menit kemudian lakukan pengecekan detail statistiknya pada server tujuan. Jika $RTT < 20\ ms$ dan *packet loss* berada pada kisaran 1 sampai 10 persen, peneliti menganjurkan untuk menggunakan *latency* setidaknya 120 ms yang merupakan *default latency* pada SRT atau masyarakat juga dapat menentukan *latency value* berdasarkan referensi dari halaman haivision.com berikut:

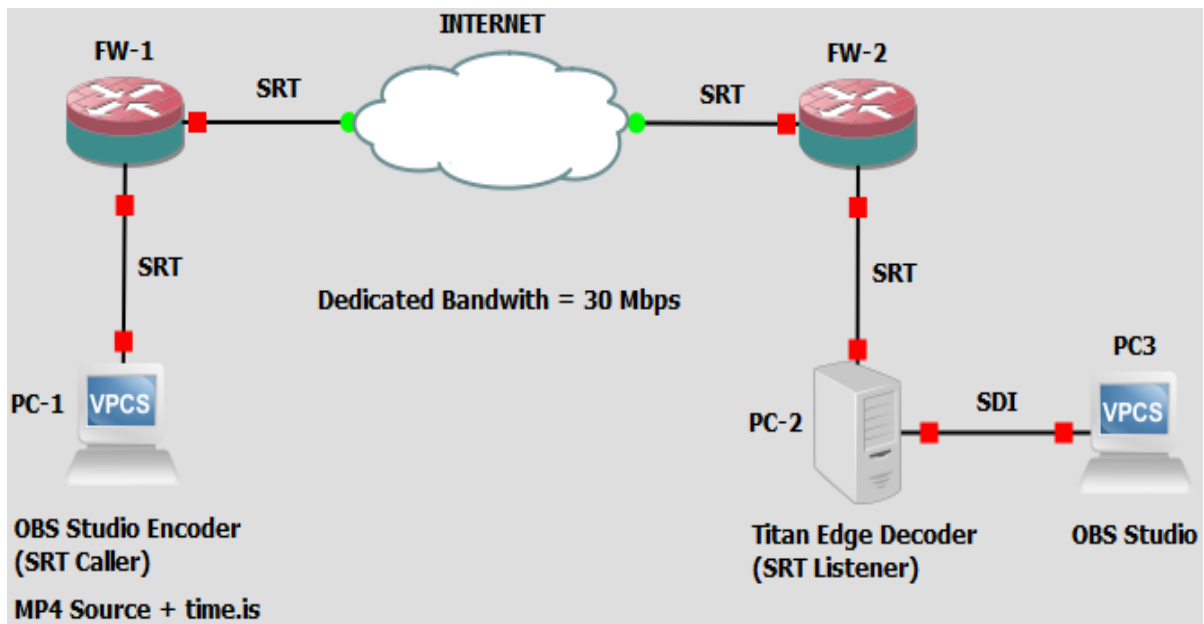
| <i>Worst Case Loss Rate (%)</i> | RTT Multiplier | Bandwidth Overhead (%) | Minimum SRT <i>Latency</i> (for $RTT \leq 20ms$) |
|---------------------------------|----------------|------------------------|--|
| ≤ 1 | 3 | 33 | 60 |
| ≤ 3 | 4 | 25 | 80 |
| ≤ 7 | 5 | 20 | 100 |
| ≤ 10 | 6 | 17 | 120 |

Tabel 1. Latency Value
Sumber: haivision.com, 2023

Ketika jaringan mengalami fluktuasi atau memiliki keterbatasan *bandwidth*, dapat terjadi situasi di mana permintaan SRT untuk penggunaan *bandwidth* melebihi kapasitas yang sebenarnya tersedia. Hal ini dapat menyebabkan penurunan kinerja, peningkatan latensi, atau bahkan *packet loss*. Dengan mengatur *maxbw*, pengguna dapat memastikan bahwa protokol SRT tidak menggunakan lebih banyak *bandwidth* daripada yang tersedia, sehingga dapat mengoptimalkan penggunaan sumber daya jaringan dan mencegah terjadinya *packet loss*. Pada dasarnya, *maxbw* berfungsi sebagai kendali yang membantu SRT beradaptasi dengan kondisi jaringan yang dinamis dan berubah-ubah. Dengan melakukan penyesuaian terhadap parameter ini sesuai dengan karakteristik jaringan, pengguna dapat meningkatkan stabilitas transmisi video dan audio serta mengurangi risiko kehilangan paket data. Adapun menentukan *value maxbw* pada SRT dapat menggunakan formula $maxbw = 2 * input\ bitrate\ (in\ bytes)$

Pada penelitian ini, peneliti akan membahas mengenai implementasi parameter *latency* dan *maxbw* dalam SRT untuk memberikan gambaran komprehensif terhadap pengaruh kritis kedua parameter tersebut terhadap kinerja sistem. Integrasi bijak antara *latency* dan *maxbw*

menjadi kunci utama dalam menciptakan keseimbangan optimal antara keandalan dan efisiensi transfer data. Hasil eksperimen ini memberikan wawasan yang mendalam tentang cara memilih dan mengatur parameter-parameter ini untuk mencapai implementasi SRT yang responsif, efisien, dan dapat diandalkan dalam berbagai skenario penggunaan. Rekomendasi disampaikan untuk mempertimbangkan karakteristik jaringan dan kebutuhan aplikasi spesifik guna mencapai hasil yang optimal. Berikut topologi yang digunakan:

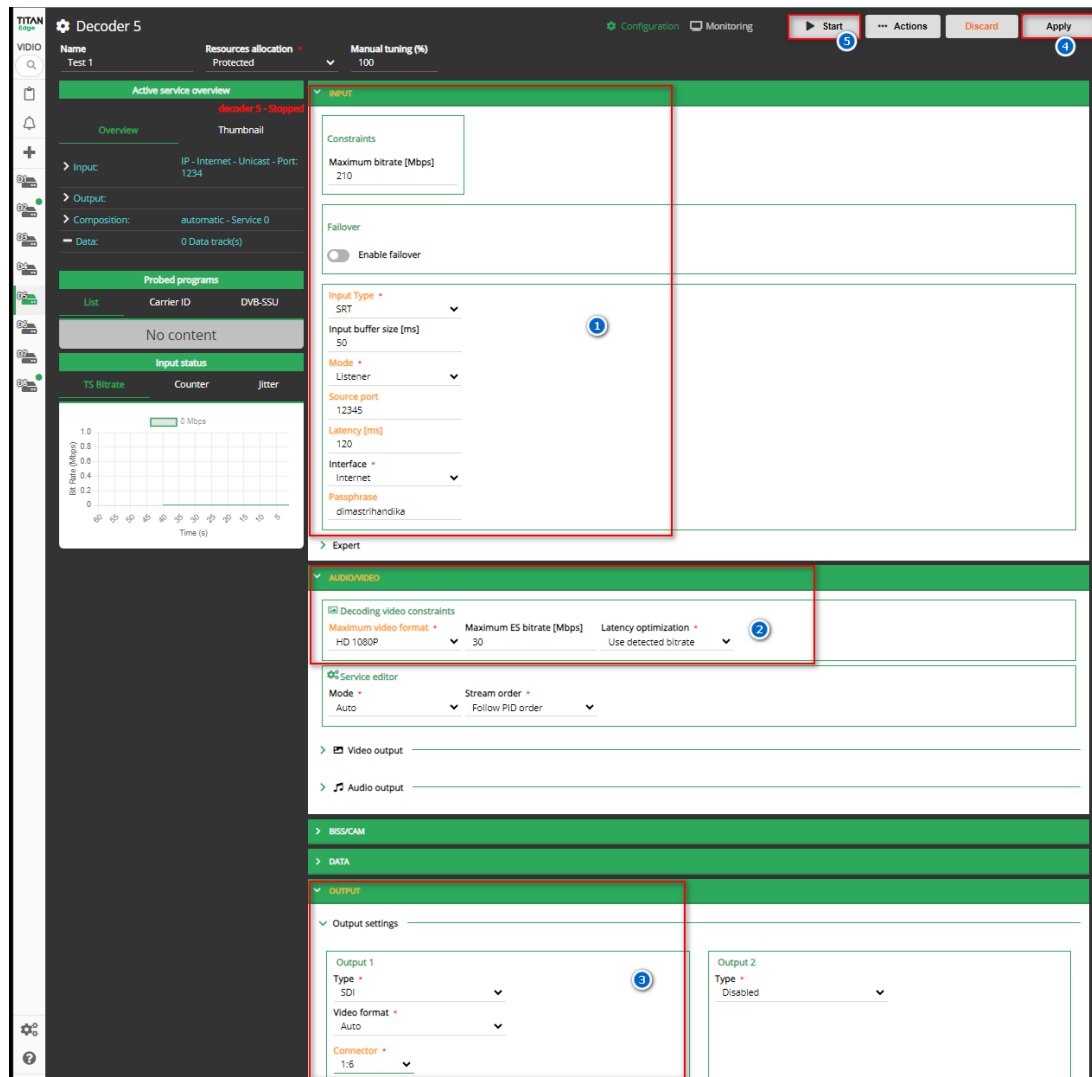


Gambar 1. Topologi Jaringan
Sumber: Dokumen Pribadi, 2023

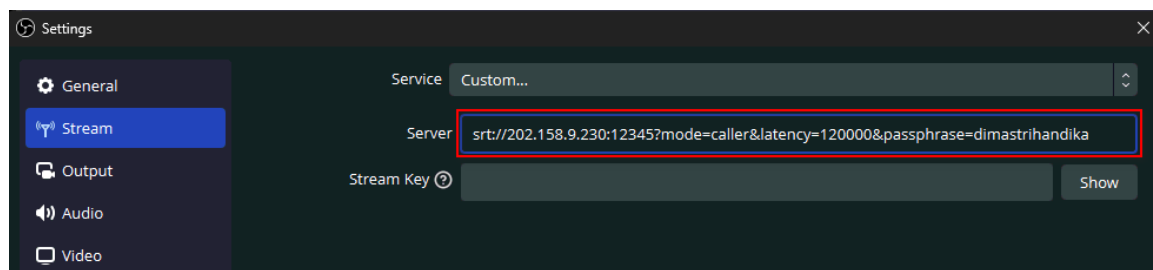
Metode Pengujian akan dilakukan masing-masing selama 10 menit dengan implementasi konfigurasi awal yang ditemukan *dropped packets*, setelah itu akan dilakukan beberapa skenario untuk menangani kendala tersebut dengan pemanfaatan parameter *latency* dan *maxbw*.

HASIL DAN PEMBAHASAN

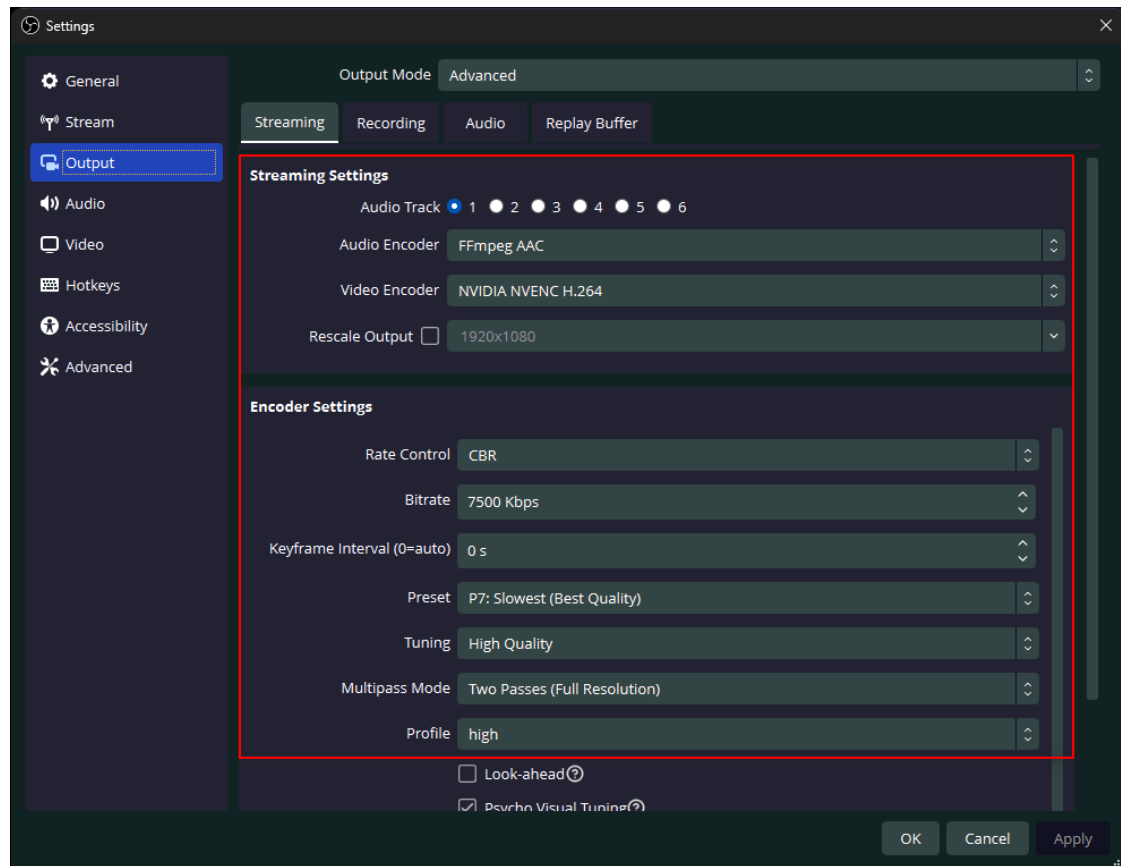
Konfigurasi Awal



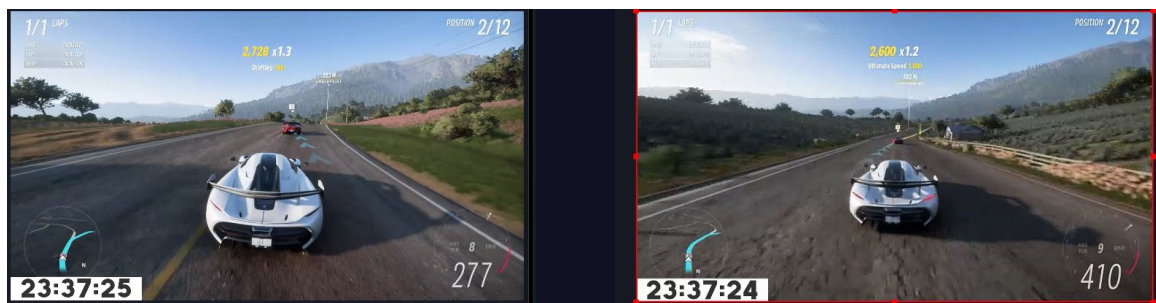
Gambar 2. Konfigurasi *Input* dan *Output Channel* pada Titan Edge
Sumber: Dokumen Pribadi, 2023



Gambar 3. Konfigurasi Parameter SRT pada Aplikasi OBS Studio
Sumber: Dokumen Pribadi, 2023



Gambar 4. Konfigurasi *Preset Encoding* pada OBS Studio
 Sumber: Dokumen Pribadi, 2023



Gambar 5. Perbandingan *Delay* antara Pengirim dan Penerima
 Sumber: Dokumen Pribadi, 2023

| Input statistics | |
|----------------------|-----------|
| Primary | |
| Source: | Primary |
| Bitrate: | 8.73 Mbps |
| Peer version: | 1.5.2 |
| Payload: | TS |
| Reconnections: | 0 |
| Buffer duration [s]: | 0.118 |
| Uptime: | 10m 7s |
| RTT: | 5 ms |
| Available buffer: | 36.48 MB |
| Reorder distance: | 72 |
| Reorder tolerance: | 0 |
| Received packets: | 488630 |
| Buffer packets: | 94 |
| Lost packets: | 4406 |
| Dropped packets: | 12 |
| Belated packets: | 0 |
| Undecrypted packets: | 0 |
| Sent NAK: | 1405 |
| Received bytes: | 602.95 MB |
| Buffer bytes: | 114.94 KB |
| Lost bytes: | 5.29 MB |
| Dropped bytes: | 14.72 KB |
| Undecrypted bytes: | 0 B |
| Reset | |

Gambar 6. Statistik Penerimaan pada Titan Edge
Sumber: Dokumen Pribadi, 2023

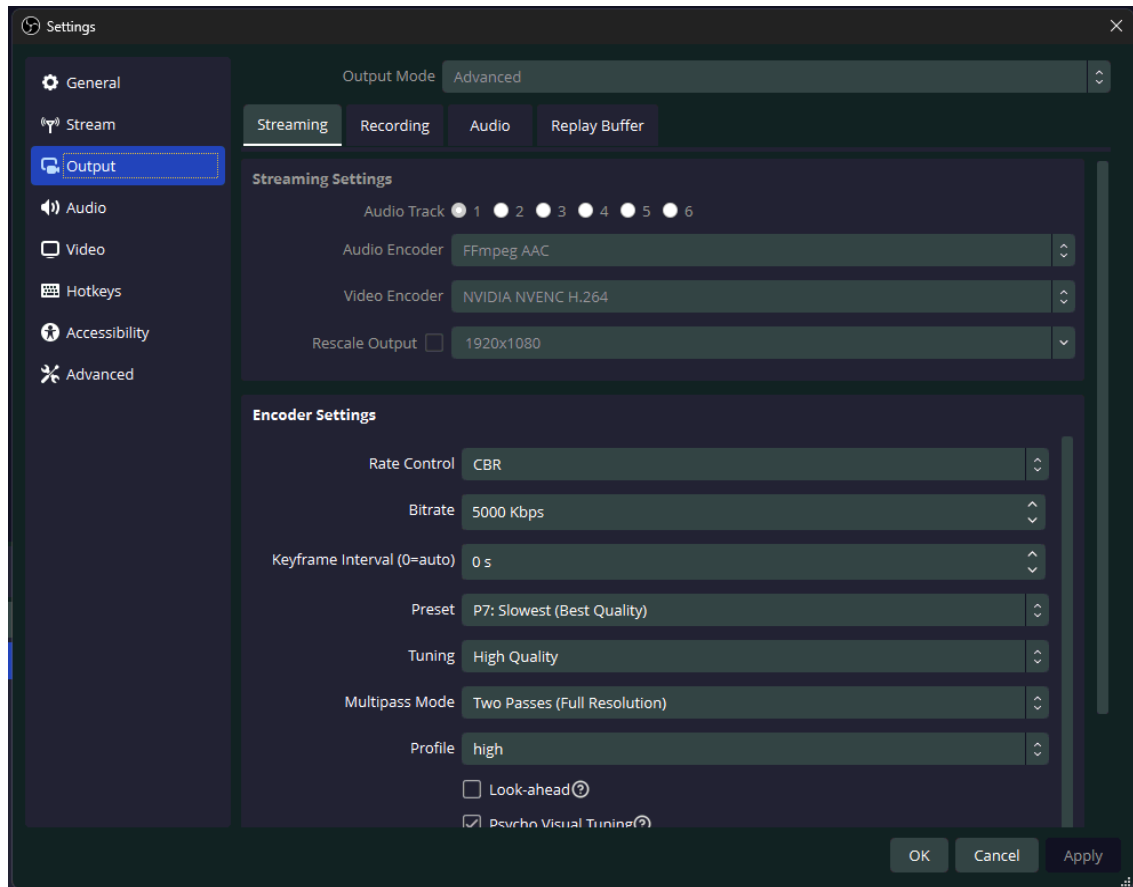


Gambar 7. Implikasi dari *Dropped Packets*
Sumber: Dokumen Pribadi, 2023

Berdasarkan hasil pengujian, terlihat pada gambar 5, *Delay* yang di dapat hanya 1 detik antara pengirim dan penerima, namun pada gambar 6 terlihat adanya *dropped packets* yang dapat disebabkan oleh beberapa faktor, salah satunya kemacetan dalam jaringan sehingga paket gagal di kirim ulang. Dampak dari kasus ini bisa di lihat pada Gambar 7 yaitu adanya artefak yang artinya terdapat *frame* yang hilang akibat *dropped packets* tersebut.

Hasil Skenario 1

Mempertahankan *Delay* namun Menurunkan Kualitas Gambar:



Gambar 8. Menurunkan *Bitrate* dari 7500 Kbps menjadi 5000 Kbps

Sumber: Dokumen Pribadi, 2023



Gambar 9. Perbedaan *Delay* antara Pengirim dan Penerima

Sumber: Dokumen Pribadi, 2023

| Input statistics | |
|----------------------|-----------|
| Primary | |
| Source: | Primary |
| Bitrate: | 6.11 Mbps |
| Peer version: | 1.5.2 |
| Payload: | TS |
| Reconnections: | 0 |
| Buffer duration [s]: | 0.116 |
| Uptime: | 10m 3s |
| RTT: | 5 ms |
| Available buffer: | 36.55 MB |
| Reorder distance: | 30 |
| Reorder tolerance: | 0 |
| Received packets: | 336583 |
| Buffer packets: | 66 |
| Lost packets: | 1676 |
| Dropped packets: | 0 |
| Belated packets: | 0 |
| Undecrypted packets: | 0 |
| Sent NAK: | 456 |
| Received bytes: | 405.77 MB |
| Buffer bytes: | 78.08 KB |
| Lost bytes: | 1.99 MB |
| Dropped bytes: | 0 B |
| Undecrypted bytes: | 0 B |
| Reset | |

Gambar 10. Statistik Penerimaan pada Titan Edge
Sumber: Dokumen Pribadi, 2023

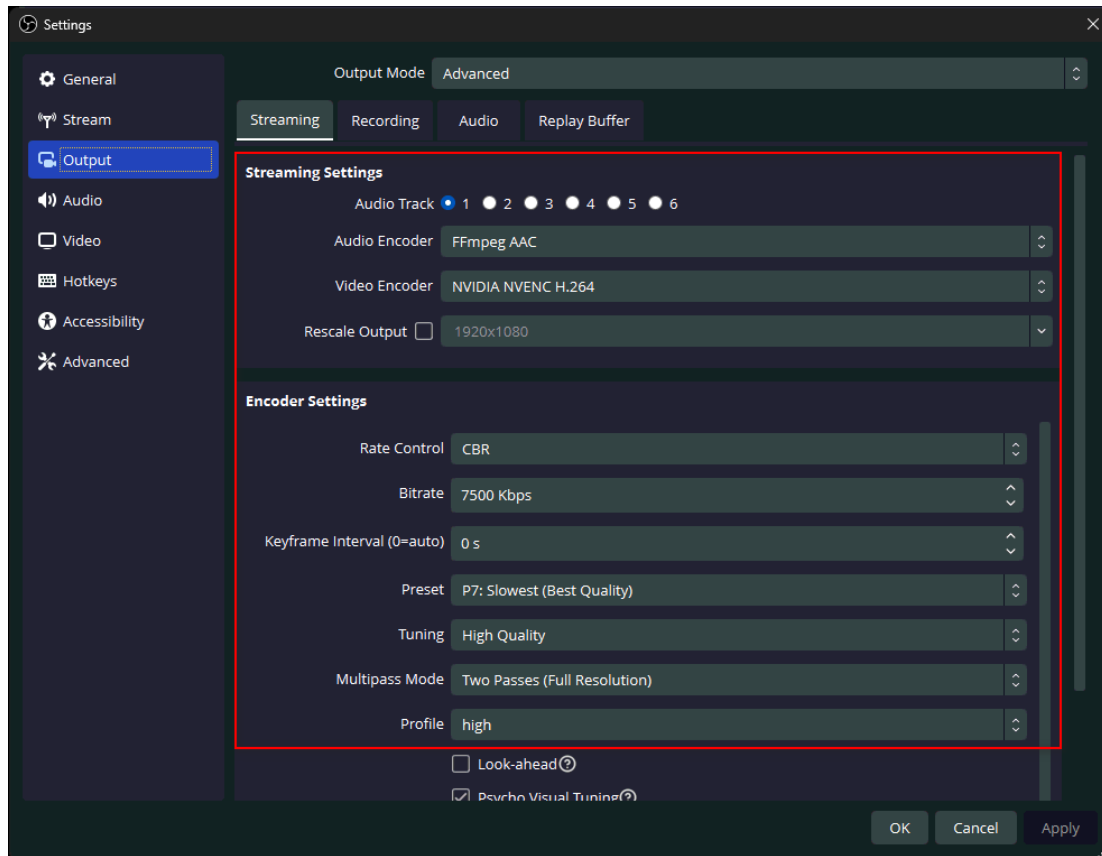
Setelah dilakukan penurunan *Bitrate* dari 7500 Kbps menjadi 5000 Kbps, terlihat tidak ada lagi *dropped packet*, *packet loss* lebih sedikit, dan *Delay* antara pengirim dan penerima tetap terjaga di 1 detik.

Hasil Skenario 2

Mempertahankan Kualitas Gambar, namun *Delay* Lebih Besar:

| | | |
|------------------------|-----------------|---|
| Input Type * | SRT | ▼ |
| Input buffer size [ms] | 50 | |
| Mode * | Listener | ▼ |
| Source port | 12345 | |
| Latency [ms] | 500 | |
| Interface * | Internet | ▼ |
| Passphrase | dimastrihandika | |

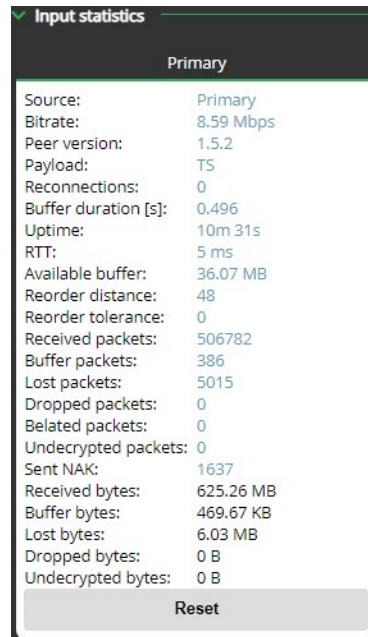
Gambar 11. Mengubah *Latency* dari 120 menjadi 500 ms pada Titan Edge
Sumber: Dokumen Pribadi, 2023



Gambar 12. Mengembalikan *Bitrate* ke 7500 Kbps
 Sumber: Dokumen Pribadi, 2023



Gambar 13. Perbedaan *Delay* antara Pengirim dan Penerima
 Sumber: Dokumen Pribadi, 2023



| Input statistics | |
|----------------------|-----------|
| Primary | |
| Source: | Primary |
| Bitrate: | 8.59 Mbps |
| Peer version: | 1.5.2 |
| Payload: | TS |
| Reconnections: | 0 |
| Buffer duration [s]: | 0.496 |
| Uptime: | 10m 31s |
| RTT: | 5 ms |
| Available buffer: | 36.07 MB |
| Reorder distance: | 48 |
| Reorder tolerance: | 0 |
| Received packets: | 506782 |
| Buffer packets: | 386 |
| Lost packets: | 5015 |
| Dropped packets: | 0 |
| Belated packets: | 0 |
| Undecrypted packets: | 0 |
| Sent NAK: | 1637 |
| Received bytes: | 625.26 MB |
| Buffer bytes: | 469.67 KB |
| Lost bytes: | 6.03 MB |
| Dropped bytes: | 0 B |
| Undecrypted bytes: | 0 B |

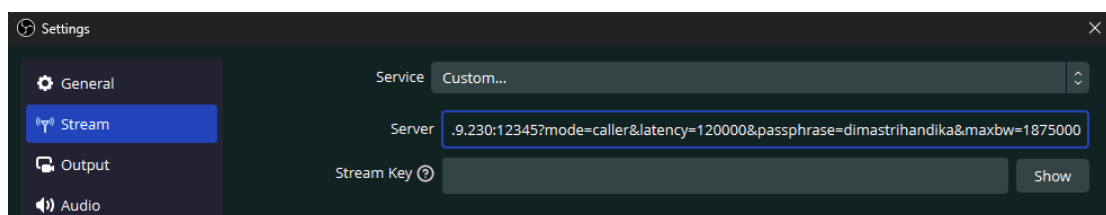
Reset

Gambar 14. Statistik Penerimaan pada Titan Edge
 Sumber: Dokumen Pribadi, 2023

Setelah dilakukan perubahan *latency* dari 120 menjadi 500 ms, terlihat sudah tidak ada lagi *dropped packet* karena semua kehilangan paket berhasil dikirim ulang akibat ruang latensi yang cukup, namun *Delay* antara pengirim dengan penerima kini menjadi 2 detik.

Hasil Skenario 3

Mempertahankan *Delay* dan Mempertahankan Kualitas Gambar:



Gambar 15. Menambahkan parameter *maxbw* pada OBS Studio Pengirim
 Sumber: Dokumen Pribadi, 2023



Gambar 16. Perbedaan *Delay* antara Pengirim dan Penerima
Sumber: Dokumen Pribadi, 2023

| Input statistics | |
|----------------------|-----------|
| Primary | |
| Source: | Primary |
| Bitrate: | 9.04 Mbps |
| Peer version: | 1.5.2 |
| Payload: | TS |
| Reconnections: | 0 |
| Buffer duration [s]: | 0.116 |
| Uptime: | 10m 58s |
| RTT: | 5 ms |
| Available buffer: | 36.53 MB |
| Reorder distance: | 31 |
| Reorder tolerance: | 0 |
| Received packets: | 526492 |
| Buffer packets: | 90 |
| Lost packets: | 668 |
| Dropped packets: | 0 |
| Belated packets: | 0 |
| Undecrypted packets: | 0 |
| Sent NAK: | 147 |
| Received bytes: | 649.28 MB |
| Buffer bytes: | 109.97 KB |
| Lost bytes: | 816.92 KB |
| Dropped bytes: | 0 B |
| Undecrypted bytes: | 0 B |
| Reset | |

Gambar 17. Statistik Penerimaan pada Titan Edge
Sumber: Dokumen Pribadi, 2023

Setelah dilakukan penambahan parameter *maxbw* pada OBS Studio Pengirim, terlihat dengan *bitrate* 7500 Kbps sudah tidak ada lagi *dropped packet*, *loss packet* mengalami penurunan yang signifikan, *Delay* antara pengirim dan penerima terjaga di 1 detik

PEMBAHASAN

Hasil penelitian pada scenario 1 sampai dengan 3 menunjukkan implementasi protokol SRT dapat menciptakan keandalan dan efisiensi pengiriman data. Hal ini dapat dilihat pada gambar 16 & 17 yang menunjukkan *Delay* yang sangat minim untuk ukuran paket yang melewati internet, *loss packets* yang hanya 0.1%, dan tidak adanya *dropped packets* selama 10 menit pengiriman data dengan *bitrate* 7500 Kbps. Hal di dukung dengan penerapan parameter

latency dan *maxbw* pada SRT yang menghasilkan temuan yang cukup signifikan dalam optimalisasi kinerja transmisi data. Penelitian ini menunjukkan bahwa penyesuaian parameter *latency* secara cermat mampu mengurangi jeda waktu transmisi, meminimalkan keterlambatan, dan meningkatkan keandalan pengiriman data. Sementara itu, penerapan parameter *maxbw* memberikan kontribusi positif dalam memastikan efisiensi penggunaan *bandwidth*.

Dari hasil penelitian menunjukkan bahwa penyesuaian parameter ini dapat meningkatkan *throughput* secara signifikan, memastikan pemanfaatan maksimal kapasitas jaringan yang tersedia. Kombinasi optimal antara *latency* dan *maxbw* membuktikan keefektifan SRT dalam mendukung transmisi data yang handal dan efisien, serta memberikan dasar yang kokoh untuk implementasi teknologi ini dalam skenario kebutuhan jaringan yang beragam.

SIMPULAN DAN SARAN

Simpulan

Menurut hasil dari penelitian ini menunjukkan bahwa protokol *Source Reliable Transport* (SRT) memiliki potensi yang signifikan dalam meningkatkan keandalan dan efisiensi pengiriman data dalam proses video *streaming*. Hasil analisis menunjukkan bahwa parameter tertentu dalam implementasi protokol SRT memiliki pengaruh yang cukup besar terhadap stabilitas dan latensi dalam kualitas pengiriman data. Hasil pengujian parameter *latency* dan *maxbw* pada SRT memberikan wawasan yang mendalam terkait optimalisasi pengiriman data, dengan menemukan titik keseimbangan yang dapat mendukung pengalaman pengguna yang lebih baik.

Oleh karena itu, implementasi SRT dalam proses video *streaming* dapat dianggap sebagai solusi yang efektif dan dapat diandalkan. Dengan demikian, penelitian ini memberikan wawasan mendalam tentang potensi SRT sebagai solusi andal untuk memenuhi kebutuhan video *streaming* modern. Hasilnya diharapkan dapat memberikan kontribusi berharga bagi pengembang aplikasi, penyedia layanan *streaming*, dan peneliti dalam mengoptimalkan protokol *transport* untuk pengiriman data video *real-time*.

Saran

Meskipun penelitian ini memberikan wawasan yang cukup mendalam tentang implementasi protokol *Source Reliable Transport* (SRT) dalam proses video *streaming*, terdapat beberapa aspek yang dapat diperhatikan untuk peningkatan lebih lanjut. Pertama, penelitian dapat diperkaya dengan menyertakan perbandingan langsung antara protokol SRT dengan protokol *streaming* lainnya untuk memberikan konteks lebih mendalam terkait

keunggulan yang dihasilkan. Selain itu, untuk memperluas generalitas hasil, peneliti dapat mempertimbangkan variasi konfigurasi perangkat keras dan jaringan dalam pengujian, untuk menggambarkan sejauh mana implementasi SRT dapat bersifat fleksibel dan dapat diandalkan dalam berbagai konteks. Selanjutnya, perlu dipertimbangkan juga untuk mengevaluasi efek penggunaan SRT dalam skenario jaringan yang lebih kompleks, seperti kondisi jaringan yang tidak stabil atau berkapasitas rendah. Keseluruhan, penelitian ini telah berhasil memberikan wawasan yang berharga terkait potensi SRT, namun demikian, untuk memastikan keberlanjutan dan relevansi, disarankan untuk melakukan eksperimen lebih lanjut dan mempertimbangkan faktor-faktor tambahan yang dapat memengaruhi kinerja protokol SRT secara menyeluruh. Sebagai saran, pengembang dan penyedia layanan *streaming* disarankan untuk lebih menggali potensi SRT dalam implementasi praktis dan mempertimbangkan penyesuaian parameter yang tepat untuk situasi penggunaan spesifik guna memaksimalkan hasil yang diperoleh dalam konteks *live streaming*.

DAFTAR PUSTAKA

- Bienik, J, Uhrina, M, Sevcik, L, & Holesova, A (2023). Impact of Packet Loss Rate on Quality of Compressed High Resolution Videos. *Sensors*, mdpi.com, <https://www.mdpi.com/1424-8220/23/5/2744>
- Chang, W, & Sonwalkar, S (2020). *Live video streaming services*. *US Patent 10,721,499*, Google Patents, <https://patents.google.com/patent/US10721499B2/en>
- Haivision. (2023). How to configure SRT settings on a video encoder for optimal performance. Diakses pada 2 Desember 2023, dari <https://www.haivision.com/blog/all/how-to-configure-srt-settings-video-encoder-optimal-performance/>
- Herr, DA, Kassimis, C, & Stevens, JW (2018). Protocol selection for transmission control protocol/internet protocol (TCP/IP). *US Patent 9,954,979*, Google Patents, <https://patents.google.com/patent/US9954979B2/en>
- ISO, "Information technology - Generic coding of moving pictures and associated audio information: Systems", ISO/ IEC 13818-1, September 2021.
- Maolin, Z (2022). Method for retransmitting lost network packet based on *transport* stream format and user datagram protocol. *US Patent 11,489,902*, Google Patents, <https://patents.google.com/patent/US11489902B2/en>
- Purbo, OW (2018). Internet-TCP/IP: Konsep & Implementasi. *Yogyakarta: Andi*, lms.onnocomer.or.id, <https://lms.onnocomer.or.id/pustaka/REVIEW-BUKU/2018-Internet-TCPIP%20Konsep%20Dan%20Implementasi.pdf>
- Prasetyo, B, Fadilah, RM, Rahma, SA, & Kale, SF (2023). Pengaruh *Live Streaming* Video Promotion, Product Price, dan Kualitas Produk terhadap Buying Decision Produk Pengguna Tiktok Shop di Kelurahan Karang Pilang *SNHRP*, snhrp.unipasby.ac.id, <https://snhrp.unipasby.ac.id/prosiding/index.php/snhrp/article/view/643>
- Ranganathan, G, & Periyaeluvan, RE (2023). Secure media *streaming* communication via user datagram protocol. *US Patent 11,611,542*, Google Patents, <https://patents.google.com/patent/US11611542B2/en>
- Roimela, K, & You, Y (2020). Video *streaming* method. *US Patent 10,600,153*, Google Patents, <https://patents.google.com/patent/US10600153B2/en>
- Smirnov, V (2018). Maximizing bandwidth utilization in networks with high latencies and packet drops using transmission control protocol. *US Patent 9,882,831*, Google Patents, <https://patents.google.com/patent/US9882831B2/en>
- Steffen, J, Salem, AS, Leicht, H, Mertens, M, & Khalil, M (2021). Technologies for managing TCP/IP packet delivery. *US Patent 11,012,367*, Google Patents, <https://patents.google.com/patent/US11012367B2/en>