Sivas Cumhuriyet Üniversitesi Bilgisayar Ağları Dersi

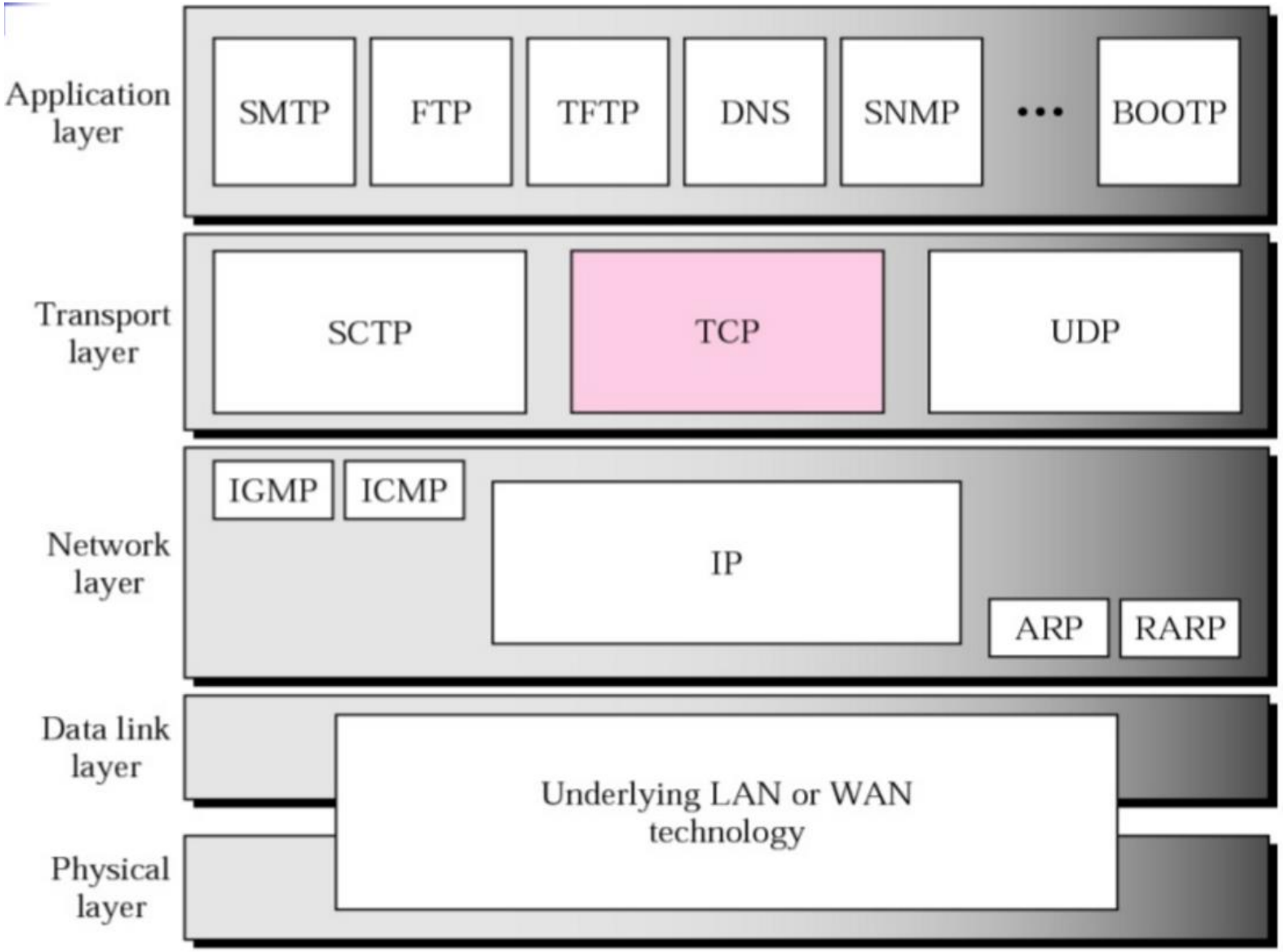
Bölüm 6
Aktarım Katmanı

Halil ARSLAN

Bölümün hedefleri

- Taşıma katmanı kavramını anlamak
- Taşıma katmanı protokolleri
  - User Datagram Protocol - UDP,
  - Transmission Control Protocol - TCP,
    - RTT
    - Akış denetimi
    - Tıkanıklık kontrolü
  - Real-time Transport Protocol – RTP
  - Real-time Transport Control Protocol - RTCP

Uçtan-uca Taşıma



Uçtan-uca Taşıma

Genel konsept olarak **taşıma**, uygulamalar arasında uçtan-uca, bağlantı sağlanması olarak tanımlanabilir. Bu katman bazen *end-to-end protocols* olarak da isimlendirilir.

Taşıma katmanı protokolleri, kaynak düğümden çıkan bir paketi, hedefe ulaşıncaya kadar güvenilir ve etkin olarak taşımalıdır.

Taşıma protokollerinin, uygulamalar açısından belirli gereksinimleri sağlamaları gerekir. Bu açıdan bu protokoller;

- Garantili mesaj dağıtımı (*mesajlar atılabilir*)
- Gönderilen sırayla mesajların dağıtımı (*mesaj sıraları bozulabilir*)
- Gönderilen her mesajın en fazla bir kopyasını dağıtma (*çoklanabilir*)
- İstenilen boyutlardaki mesajlara destek (*MTU sınırları*)
- Alıcı-gönderici arasındaki sankronizasyon (*uzun gecikmeler olabilir*)
- Alıcı-gönderici için akış kontrol desteği (*best-effort delivery*)
- Her düğümde, birden çok uygulama sürecine izin vermeli (*multitasking*)

User Datagram Protocol - UDP

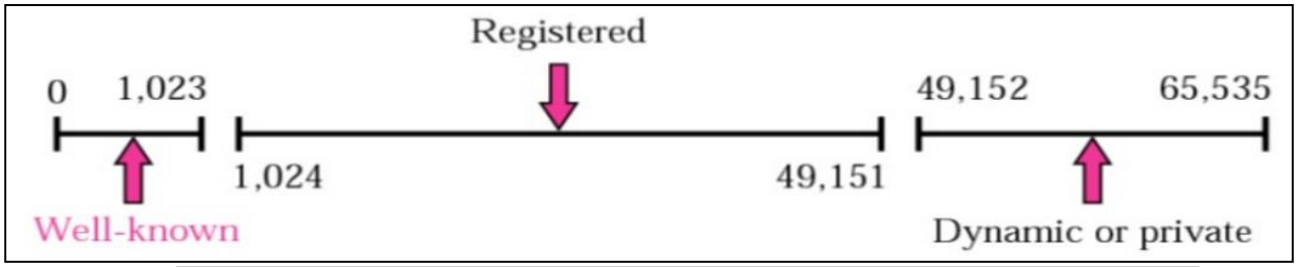
UDP, yapısı ve fonksiyonları oldukça basit bir protokoldür. UDP başlığı, kaynak port numarası, hedef port numarası, paket uzunluğu ve hata tespiti sağlamaktadır.

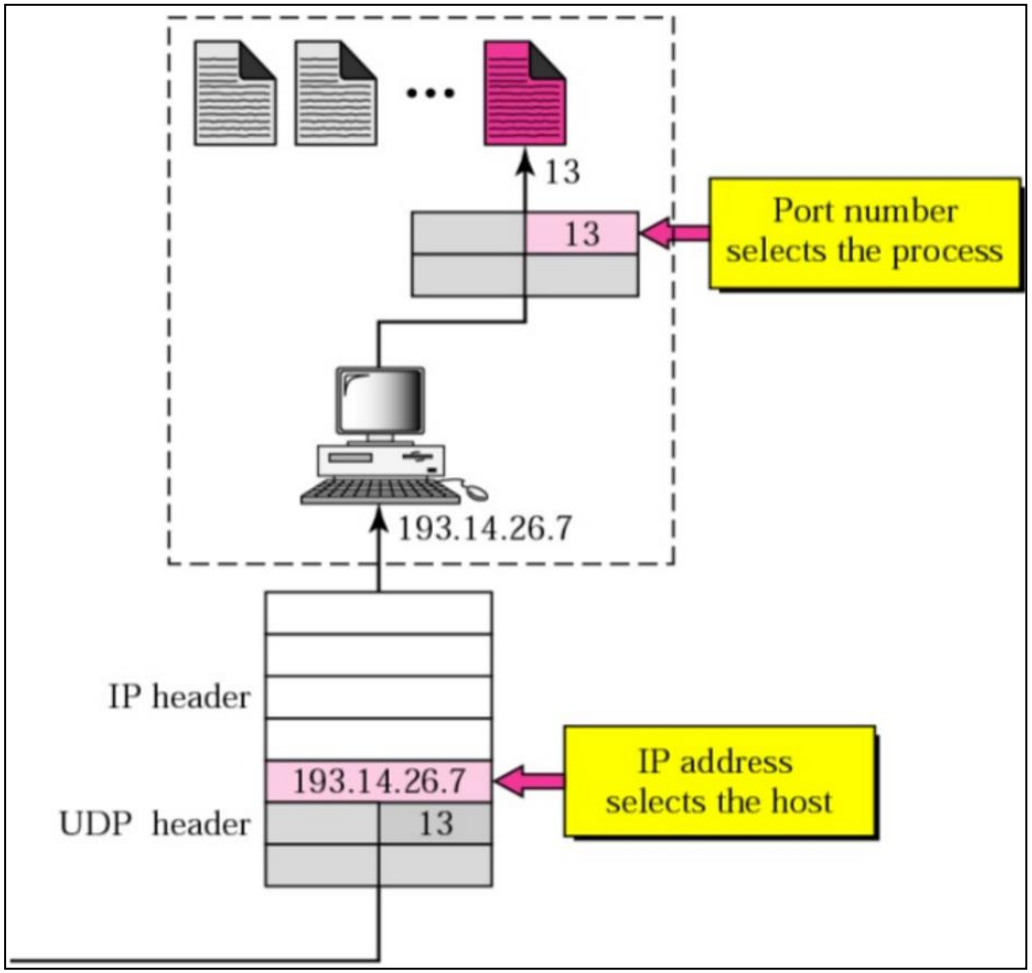
Uygulama noktasında UDP paketleri, kendi **kaynak port**u üzerinden **hedef port**a veri iletir. **Port**, uygulamalar arasında uçtan-uca taşıma işlevlerinin ayrıştırıldığı servis numaralarıdır. Örn: DNS 53, HTTP 80, SMTP 25 gibi

Geleneksel anlamda istemci-sunucu iletiminde; istemci, sunucunun bilinen portuna bağlanır. Bu bağlantı sonrasında sunucu, istemcinin portunu öğrenir ve karşılıklı iletişim sürdürülegelir.

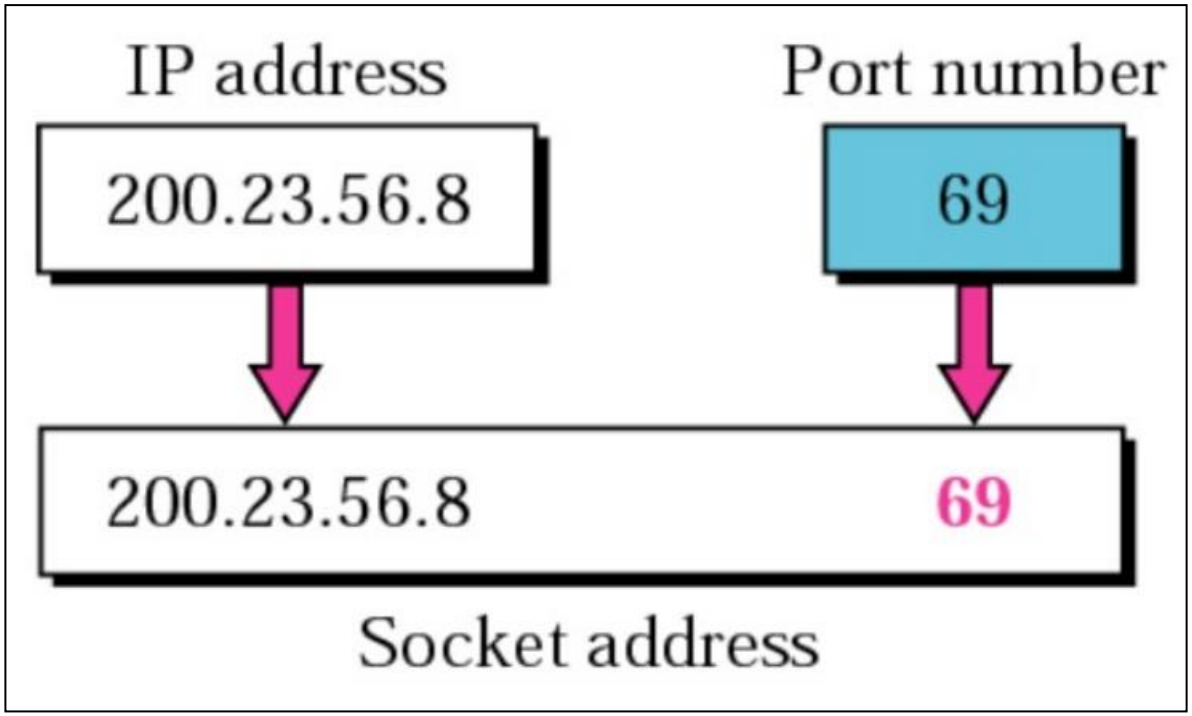
0-15	16-31
16 bit kaynak port	16 bit hedef port
16 bit paket uzunluğu	16 bit checksum

UDP – Port Numarası





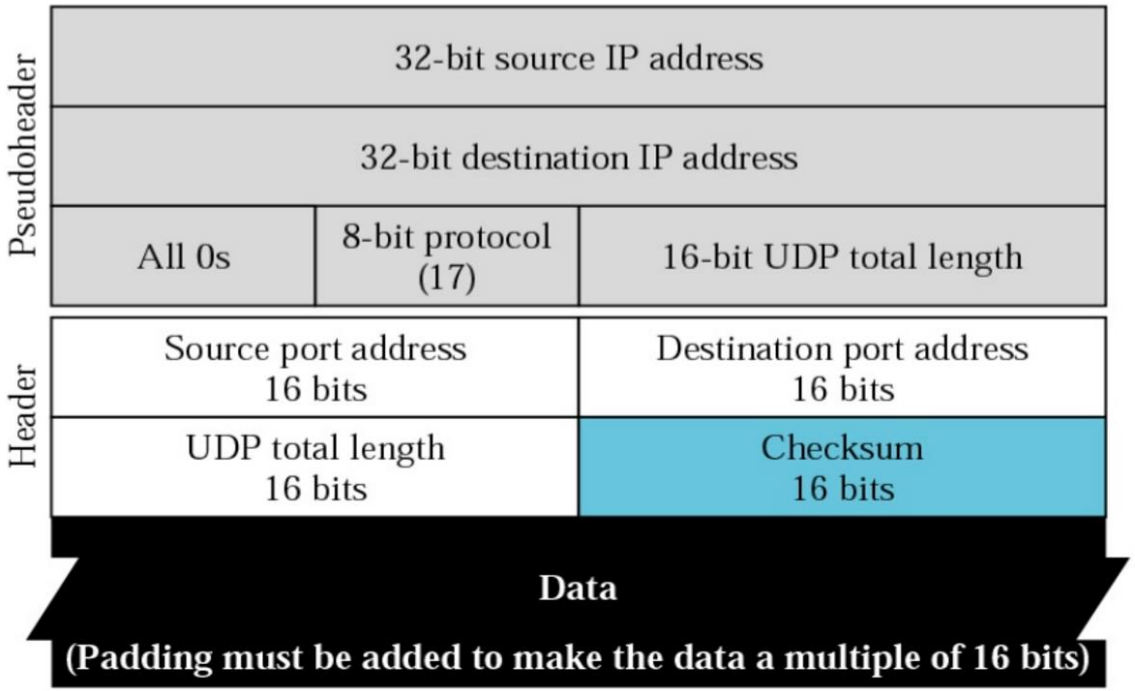
UDP – Soket adres



UDP – Pakat başlığı

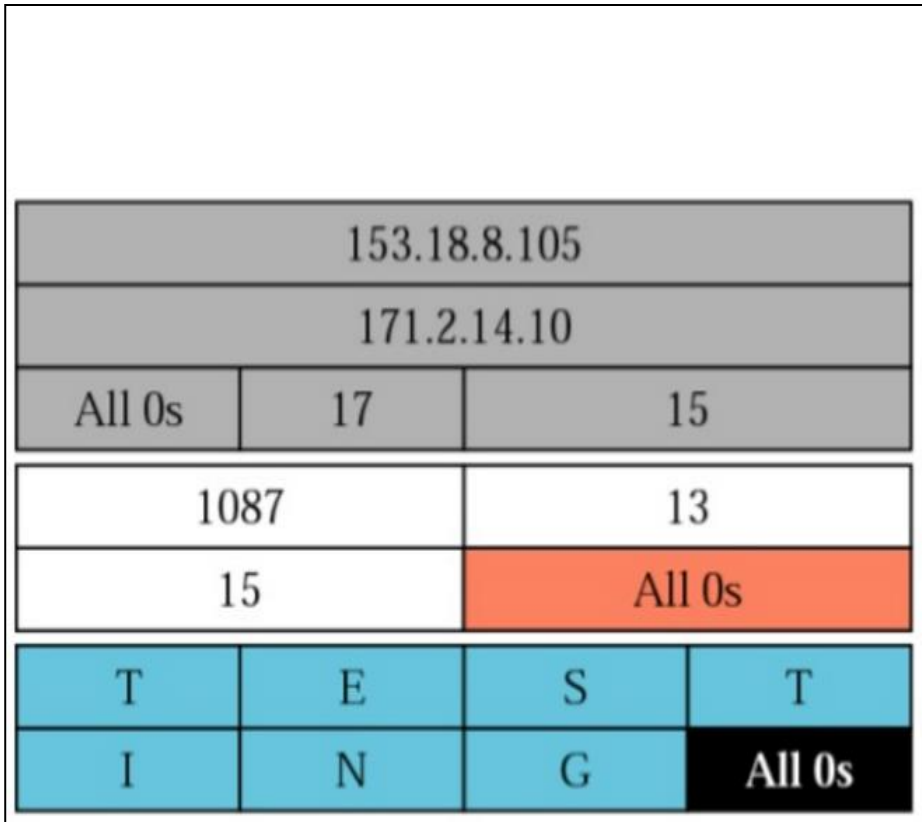
Paket Uzunluğu = IP datagram uzunluğu – IP Başlığı Length değeri
yada
Paket Uzunluğu = Data + UDP Header (8)

Checksum = PseudeHeader + UDP Header + UDP Data
PseudeHeader = Kaynak IP + Hedep IP + Protocol (IP'de) + UDP toplam uzunluk + 8 bit sıfır (12 byte tamamlamak için)



UDP – Checksum

UDP checksum hesaplaması, IPv4'de opsiyonel olmakla birlikte, bu hesaplama IP başlığından checksum'ın kaldırılması nedeniyle IPV6'da zorunludur.



10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

User Datagram Protocol - UDP

UDP, bağlantısız ve yeniden iletim mekanizması olmadığı için veri transferi daha hızlıdır.

Ayrıca sıra numarası ve onay mekanizması tanımlanmadığından, aktarılan paketlerin doğru sırası ve bir paketin alınıp alınmadığı bilinmemektedir.

İletilen verinin hedefe ulaşıp ulaşmadığı ağın o anki durumuna bağlıdır. Ayrıca IP ağlarının doğası gereği, bazı paketler çoğaltılmış olabilir ve bazı paketler de sırası bozulmuş şekilde gelebilir.

Ancak artan UDP temelli band genişliği kullanımı, ağ çöküşünü önlemek için kullanılan tıkanıklık kontrolü içermemesi nedeniyle ciddi endişeler ortaya çıkarmaktadır.

Herşeyden önce UDP bağlantı yükü azdır ancak UDP datagramlarının ağda kaybolması, iletilememesi, bozulması durumları ortaya çıkabilir.

Transmission Control Protocol - TCP

TCP protokolü, bağlantı yönelimli, noktadan noktaya güvenilir bir aktarım protokolüdür.

TCP, eksik segmentlerin ve sırası bozulmuş verilerin düzenlenmesi için mekanizmalar içermesi, internet uygulamalarının pek çoğunun TCP tabanlı olmasına yol açmıştır (şimdilik? - http, ftp, mail vb. trafikler).

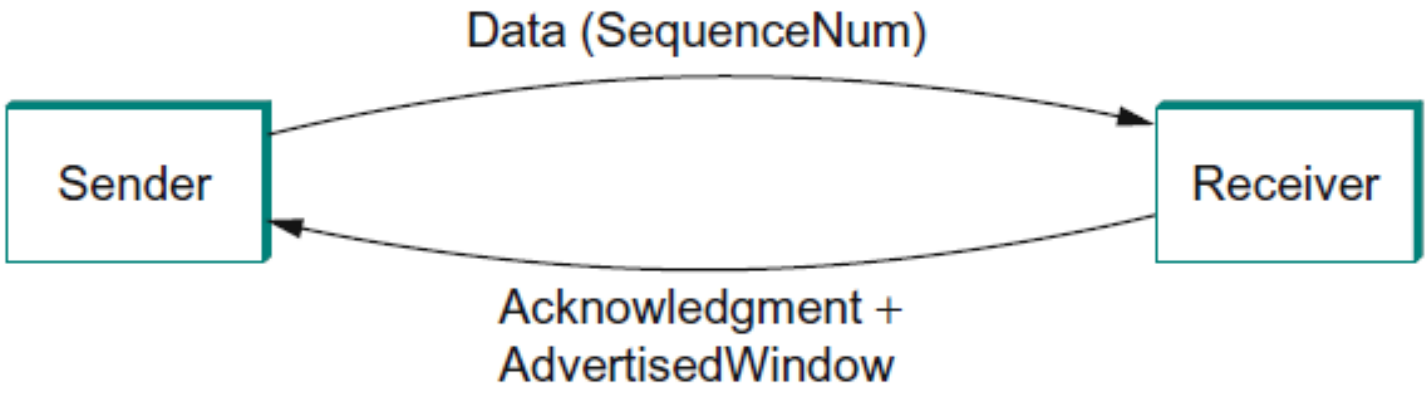
TCP, byte tabanlı bir protokol olmakla birlikte, veriler **segment** adı verilen paketler halinde taşınır.

<table>
<tr><th colspan="8">0-15</th><th>16-31</th></tr>
<tr><td colspan="8">16 bit kaynak port</td><td>16 bit hedef port</td></tr>
<tr><td colspan="9">32 bit sıra numarası</td></tr>
<tr><td colspan="9">32 bit onay numarası</td></tr>
<tr><td>Başlık Uzunluğu</td><td>Saklı</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16 bit pencere boyu</td></tr>
<tr><td colspan="8">16 bit checksum</td><td>16 bit acil gösterici</td></tr>
<tr><td colspan="9">Seçenekler (gerekiyorsa)</td></tr>
</table>

TCP – Segment yapısı

Sıra Numarası (*SequenceNum***)**, **Onay Numarası (***Acknowledgment***)** ve **Pencere Boyutu** (*AdvertisedWindow*) alanları, TCP'nin kayan pencere algoritmasında kullanılır.

Veri akışı gerçekte çift yönlü olsa da örneği basitleştirmek için tek yönlü bir akış üzerinden;
İletilen byte dizisini tanımlayan sıra numarası üzerinden alıcıya sıra numarasını onay numarası olarak kullanarak bu iletim hakkında cevap verecektir.



TCP – Sıra Numarası

Örnek: Bir TCP bağlantı üzerinden 5000 byte veri transfer edilmek isteniyor ve her segment'te 1000 byte veri gönderilecektir. İlk byte'ın sıra numarası 1 olarak tanımlandığına göre her segment'in sıra numarasını hesaplayın.

Cevap:

1. Segment → SN: 1 (veri: 1-1000)
2. Segment → SN: 1001 (veri: 1001-2000)
3. Segment → SN: 2001 (veri: 2001-3000)
4. Segment → SN: 3001 (veri: 3001-4000)
5. Segment → SN: 4001 (veri: 4001-5000)

şeklinde gönderilecektir.

TCP – Segment yapısı

6-bit bayrak alanında, TCP bağlantıları arasında kontrol bilgisinin taşınımı için kullanılan SYN, FIN, RESET, PUSH, URG ve ACK bulunur.

SYN ve **FIN**, bağlantının kurulumu ve sonlandırılmasında kullanılır.

ACK bayrağı, Acknowledgment alanının doldurulduğu tüm durumlarda 1'e kurulur.

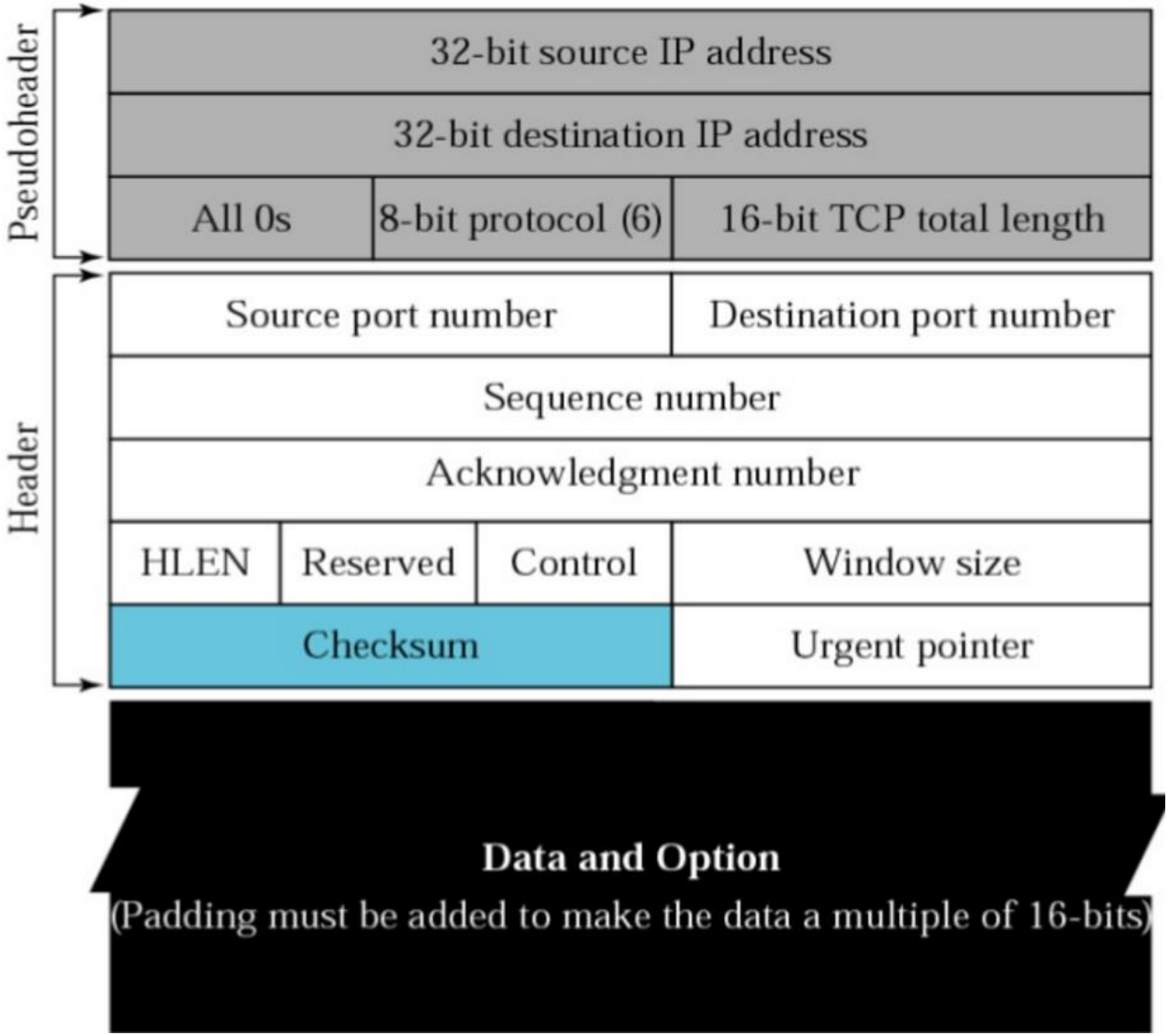
URG bayrağı, tamponda, ilgili segmentin öncelikli veri içerdiğini tanımlar. URG'li segment tampon dolmadan uygulamaya verilir. Çok az örneği var. Telnet'te özel bir komut hariç görmedim.

PUSH bayrağı, göndericinin alıcı tarafında tampon kilitlenmesi ile karşılaşmaması için, tamponun dolmasını beklemeksizin mevcut segmentleri uygulamaya aktarması için kurulur. Dosya bitti işle veya Real Time küçük paketlerde tamponun dolmasını bekleme işle. Kısacası şimdilik gönderecek başka veri yok işle der.

RESET bayrağı, alıcıda meydane gelen karışıklık sonucu bağlantının iptal edilmesini talep eder. (alınmaması gereken bir segment alındığında)

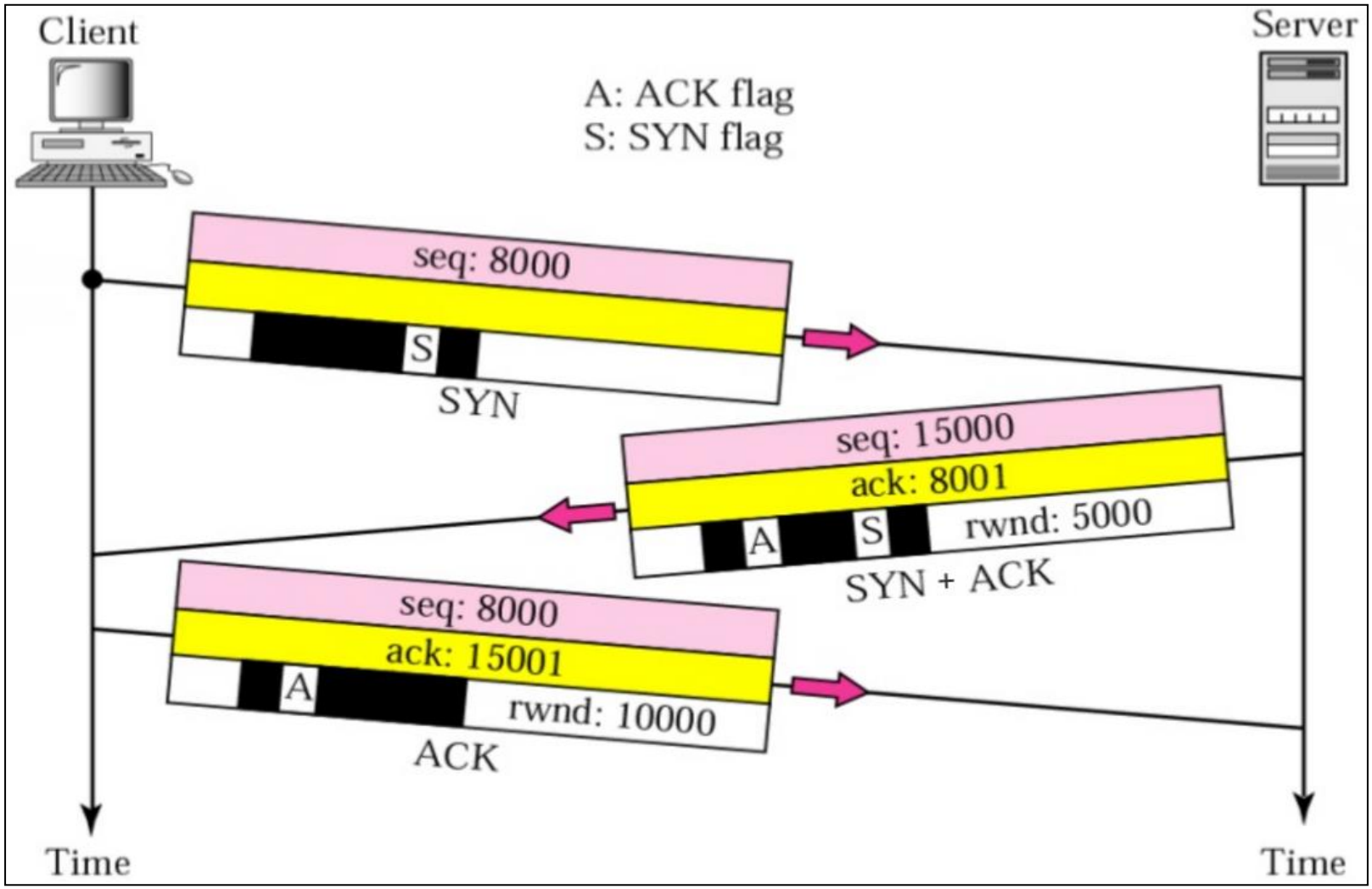
TCP – Hata denetimi (checksum)

TCP'de hata denetimi zorunludur ve UDP'ye çok benzer.

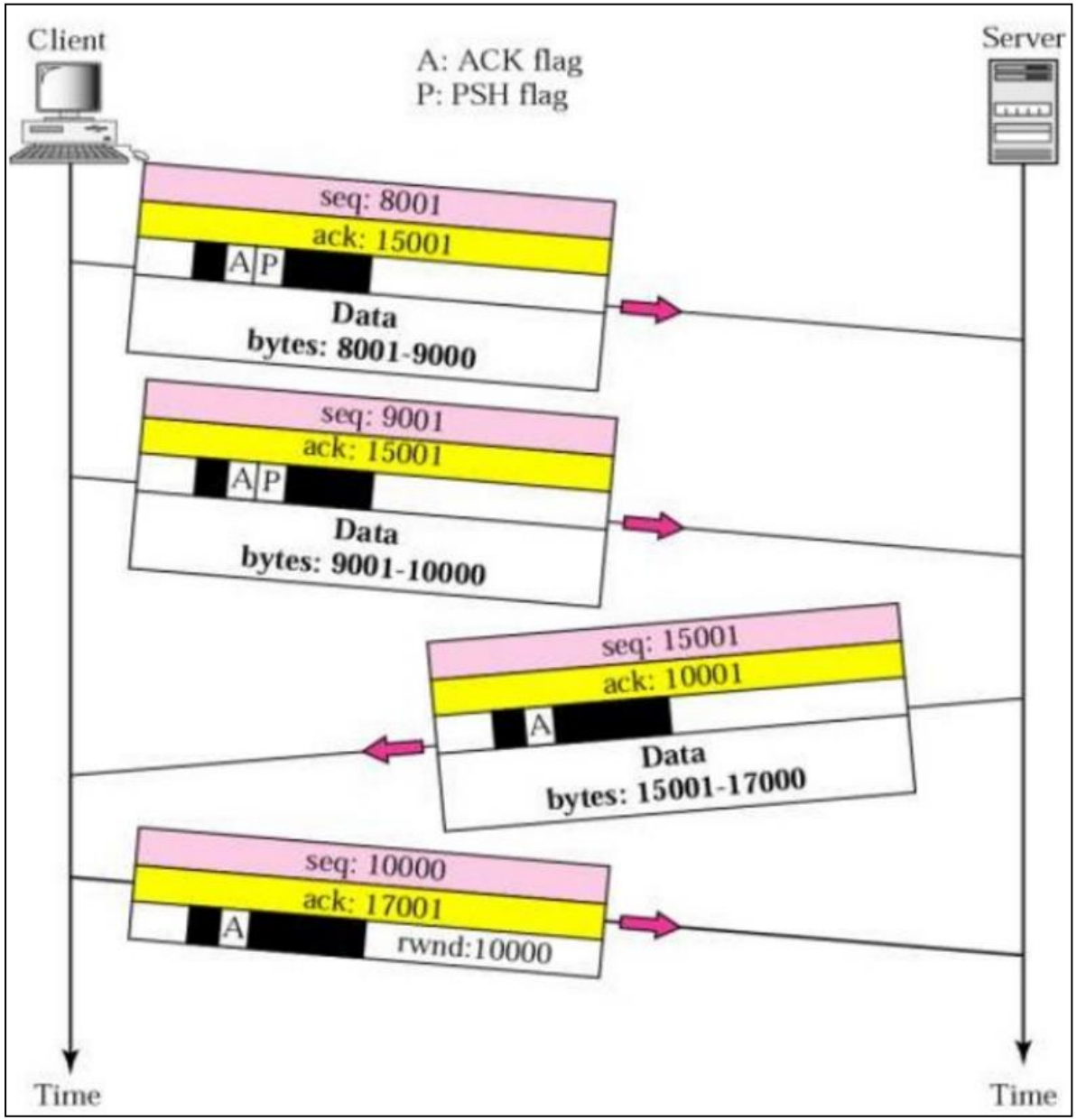


TCP – Bağlantı kurulumu

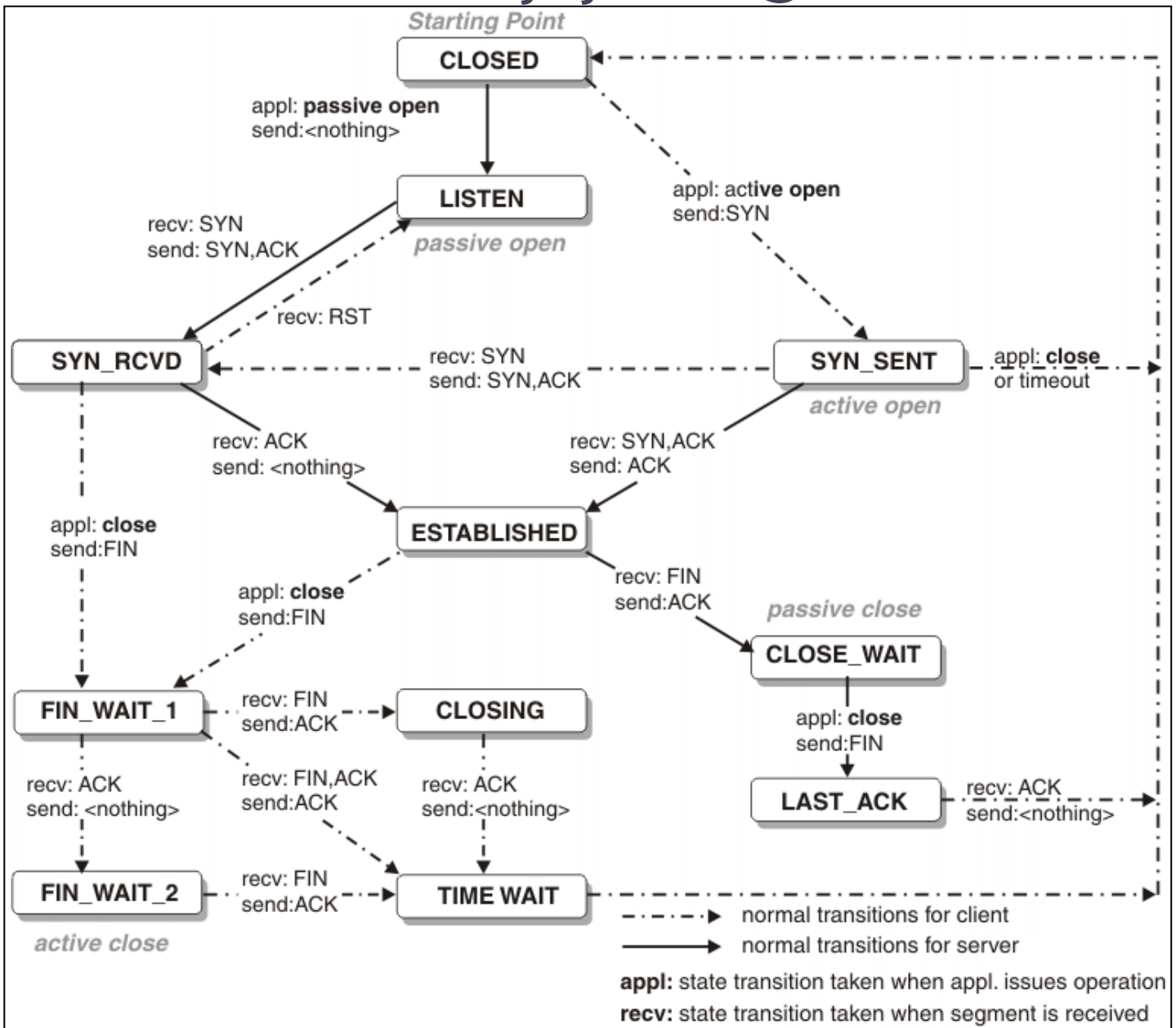
TCP'de bağlantı kurulumu **3 yönlü el sıkışma** (*three-way handshake*) tekniği ile sağlanır.



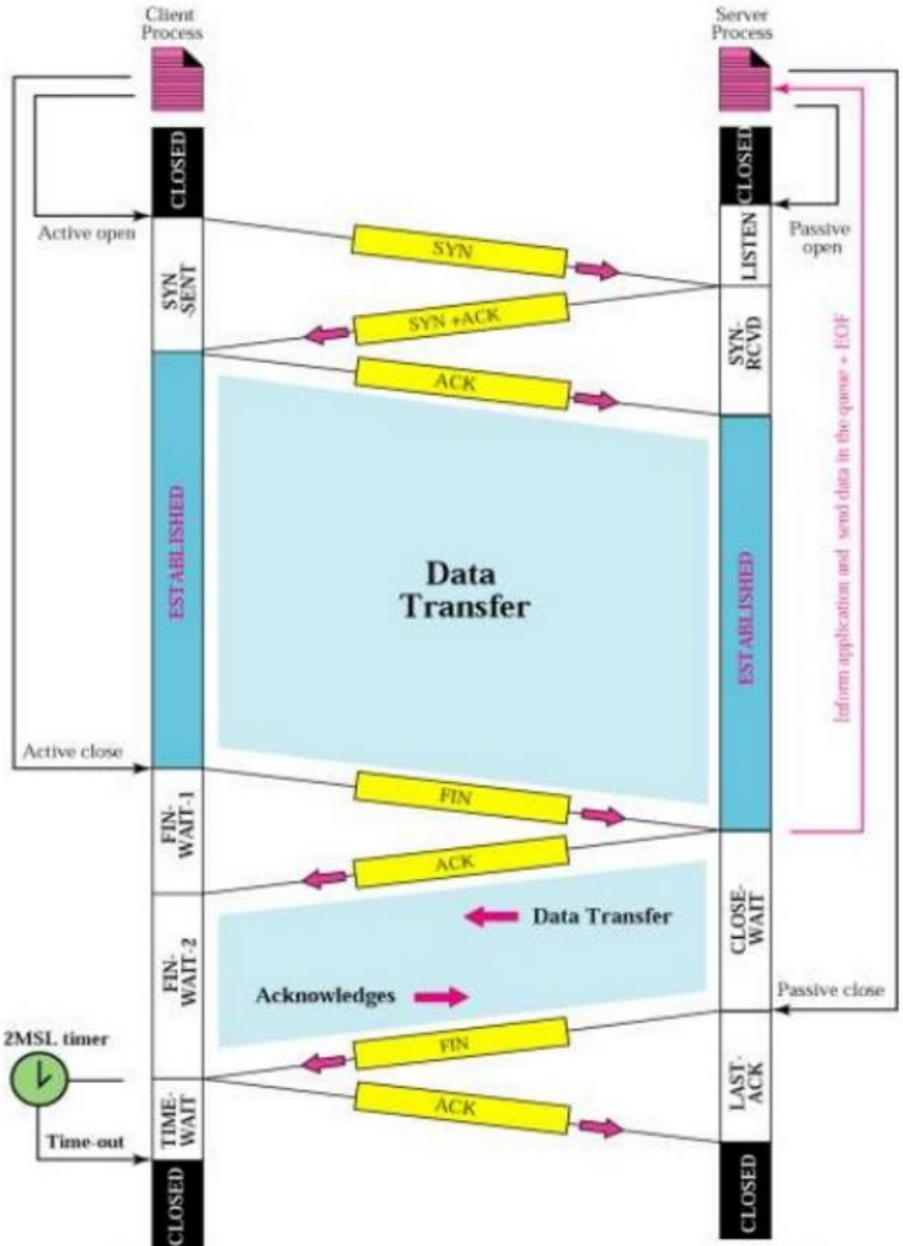
TCP – Veri aktarımı



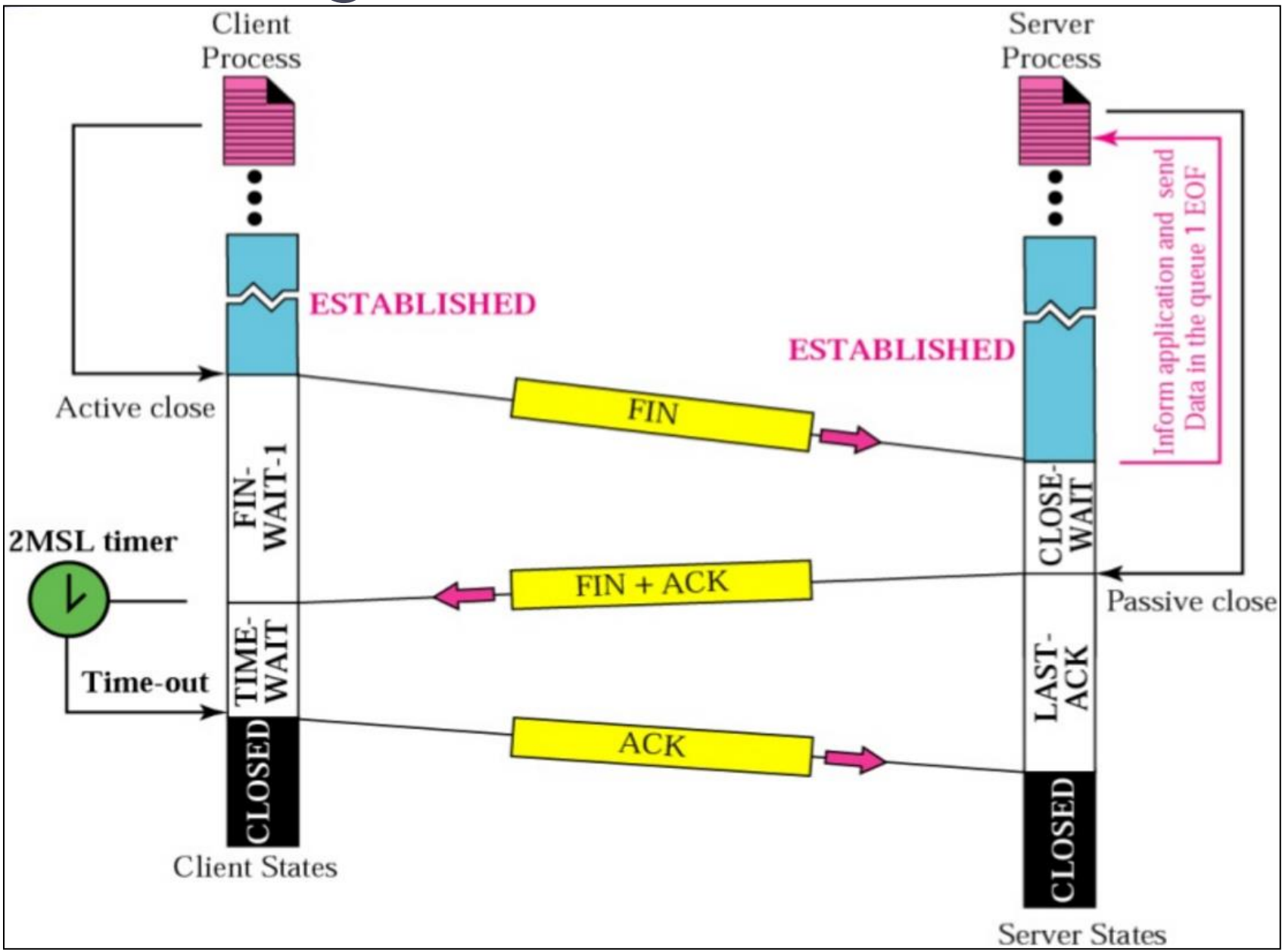
TCP – Durum Geçiş Diagramı



TCP – Normal senaryo



TCP – Bağlantının sonlandırılması 1



TCP – Bağlantının sonlandırılması

TCP'de bağlantı sonlandırılması;

- ESTABLISHED→FIN WAIT 1→FIN WAIT 2→TIME WAIT→CLOSED
- ESTABLISHED→CLOSE WAIT→LAST ACK→CLOSED
- ESTABLISHED→FIN WAIT 1→CLOSING→TIME WAIT→CLOSED

Internet'te TIME_WAIT süresi 2*120 sn. (*2 MSL-Maximum segment lifetime*) olarak kullanılır.

TCP – Zaman Aşımı

Timeout – zaman aşımı: TCP zaman aşımı süresinin belirlenmesi önemlidir. RTT'den büyük olmalıdır ancak çok büyük olmamalıdır. Kısa olursa gereksiz yeniden iletime, çok büyük olursa segment kayıplarına geç reaksiyona neden olur.

SampleRTT = Gönderilen segment ile ACK alımı arasındaki geçen süre. Bir anda bir RTT ölçümü yapılabilir ve fazla oranda değişkenlik gösterebilir. Bu nedenle tahminsel/ortalama bir RTT değeri hesaplanır.

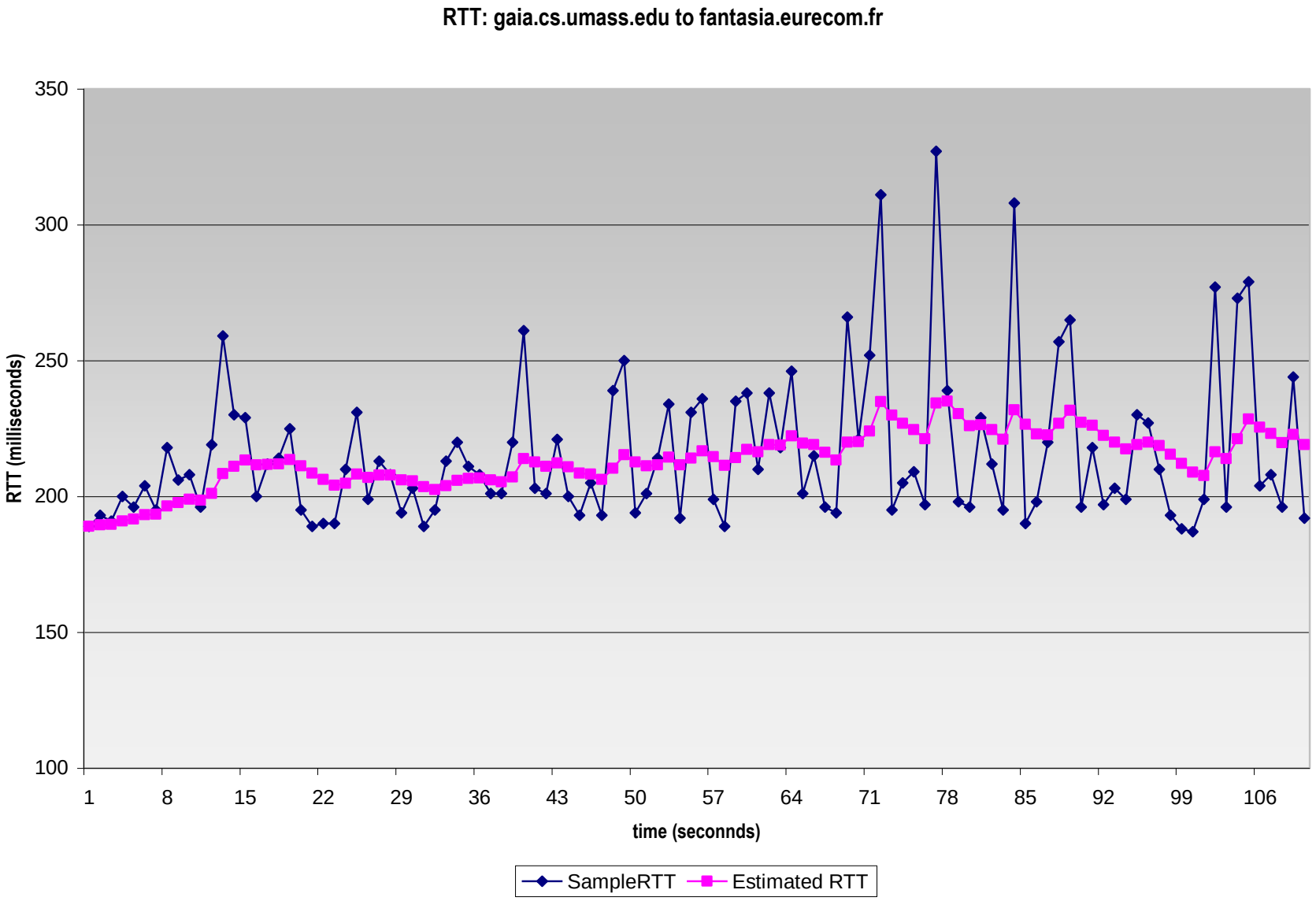
EstimatedRTT: SampleRTT değerleri çok değişkendir. Bu nedenle, bir önceki tahmini RTT değerini kullanan ortalama bir tahmini RTT değeri hesaplanır.

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

NOT: $\alpha = 0.125$ (1/8) [RFC 6298].

TimeOut: TimeOut = 2×EstimatedRTT ile hesaplanır.

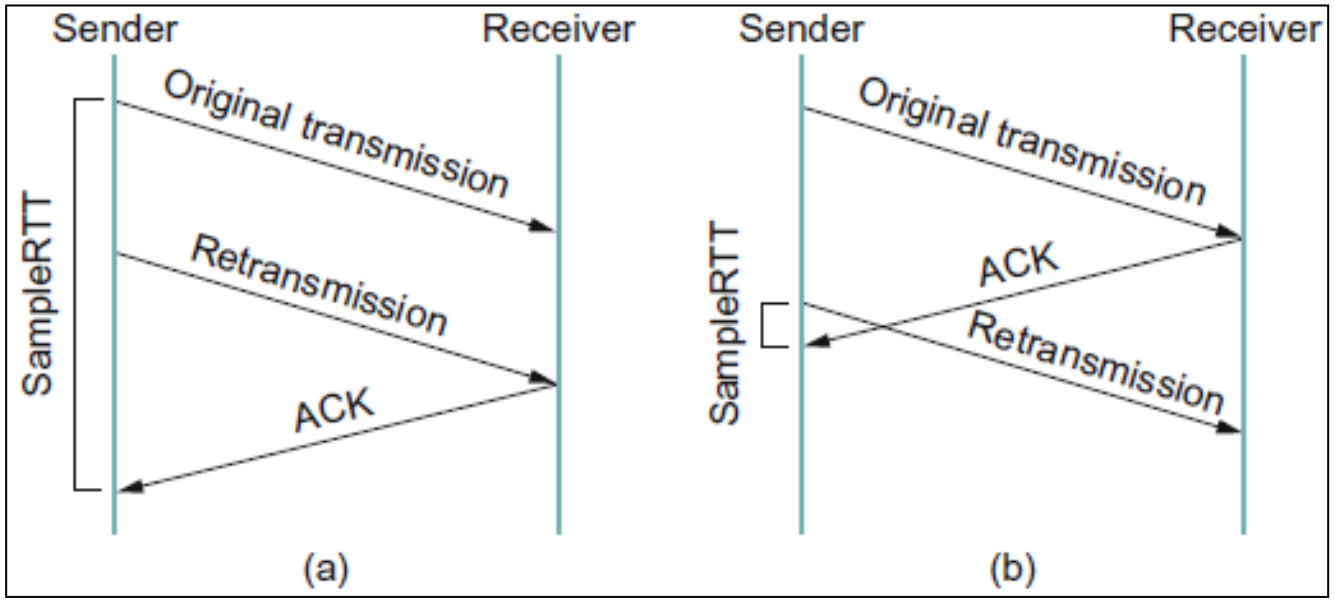
TCP – Zaman Aşımı



TCP – Zaman Aşımı

Karn/Partridge Algoritması: SampleRTT şekildeki gibi, a kabul edilirse çok büyük, b kabul edilirse çok küçük.

Bu algoritma RTT ölçümünü segment tekrar iletime geçtiği zaman durdurur. Bu algoritmaya göre sampleRTT ölçümü segment sadece bir kez gönderildiği zaman yapılır.



TCP – Zaman Aşımı

Jacobson/Karels Algoritması: EstimatedRTT de ki değişimin güvenlilirliğini belirlemek gerekir. Bu algoritmaya göre bir önceki EstimatedRTT değerinin değişiminin belirlenmesi gerekir. Uç değişimlerin etkisi azaltılır.

$$Deviation = (1 - \beta) * Deviation + \beta * |SampleRTT - Estimated\ RTT|$$

$\beta = 0.25$ olarak kullanılır.

$$\textbf{\textit{TimeOut}} = EstimatedRTT + 4 * Deviation$$

olarak hesaplanır.

TCP – Zaman Aşımı

RTT Örnek: Bir TCP bağlantının SampleRTT değeri 1.5 ms hesaplansın. Timeout (RTO) ve EstimatedRTT değerlerini hesaplayın.

SampleRTT = 1.5

EstimatedRTT = (1-α) * EstimatedRTT+ α*SampleRTT (α=0.125)
EstimatedRTT = 0.875*1.5+ (0.125*1.5)
EstimatedRTT = 1.5

Deviation = (1 - β) * Deviation + β * |SampleRTT -Estimated RTT| (β=0.25)
Deviation = 0 .75 * 1 + 0.25 * (1.5-1.5)
Deviation = 0.75

Timeout (RTO) = EstimatedRTT + 4 * Deviation
Timeout (RTO) = 1.5 + 4*0.75
Timeout (RTO) = 4,5 ms

TCP – Zaman Aşımı

RTT Örnek: Önceki örnek için yeni SampleRTT değeri 2.5 ms hesaplansın. Timeout (RTO) ve EstimatedRTT değerlerini hesaplayın.

SampleRTT = 2.5

EstimatedRTT = (1-α) * EstimatedRTT+ α*SampleRTT (α=0.125)
EstimatedRTT = 0.875*1.5+ 0.125*2.5
EstimatedRTT = 1.625

Deviation = (1 - β) * Deviation + β * |SampleRTT -Estimated RTT| (β=0.25)
Deviation = 0 .75 * 0.75 + 0.25 * | 0,875 |
Deviation = 0.78125

Timeout (RTO) = EstimatedRTT + 4 * Deviation
Timeout (RTO) = 1.625 + 4* 0.78125
Timeout (RTO) = 4,75 ms

TCP – Akış denetimi-Sliding window

TCP, Kayan pencere yöntemini kullanarak;

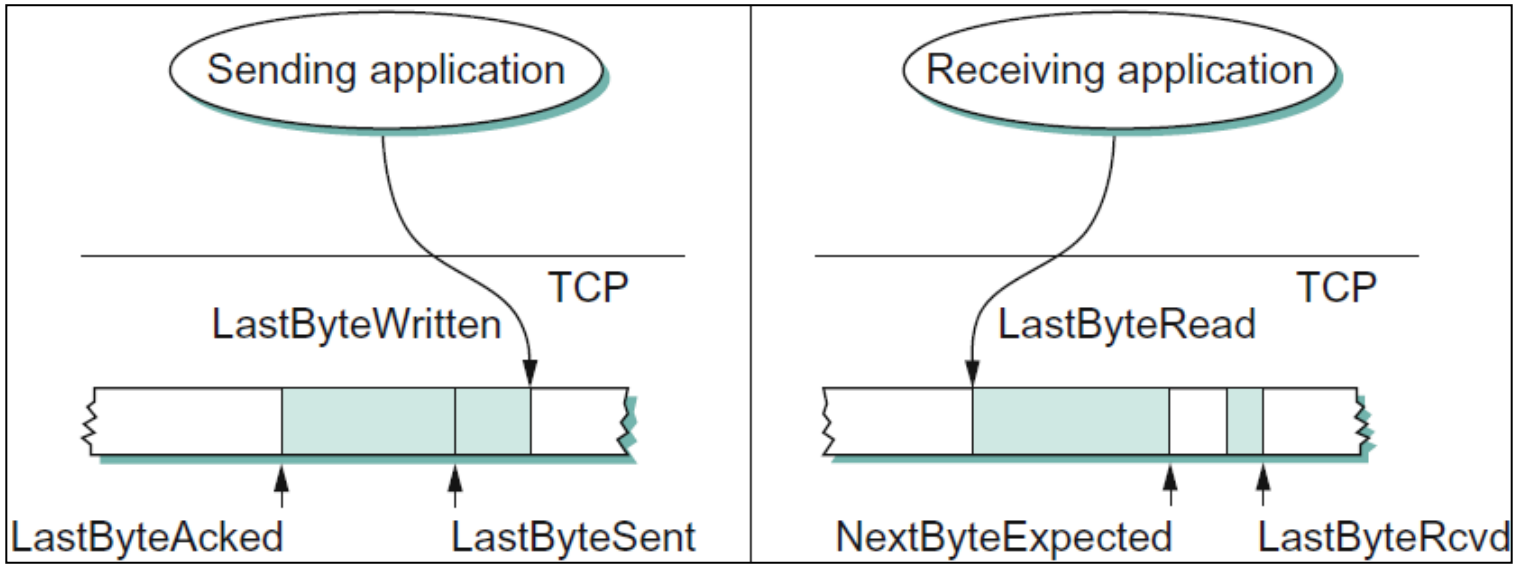
- Güvenilir veri dağıtım garantisi
- Doğru sıra ile veri dağıtım garantisi
- Alıcı ve gönderici arasında akış kontrolü sağlar.

TCP, standart kayan pencere yönteminde tanımlanan pencere boyutunu, TCP segmentinde tanımlı *AdvertisedWindow* alanı ile dinamikleştirerek akış denetimi yapmaktadır.

AdvertisedWindow: Alıcıdan onay beklemeden gönderilebilecek byte boyutu. (Kayan pencere algoritmasındaki pencere boyutu)

NOT: Alıcı tamponundan daha büyük byte gönderimini engellemek için.

TCP – Akış denetimi-Sliding window



LastByteAcked: Onaylanan (ACK) byte
LastByteSend: Gönderilen son byte
LastByteWritten: Hazır byte'lar

LastByteRead:Onaylanan (ACK) byte
NextByteExpected: Beklenen byte
LastByteRcvd: Alınan son byte

MaxRcvBuffer ≥ LastByteRcvd−LastByteRead
AdvertisedWindow = MaxRcvBuffer−((NextByteExpected−1)−LastByteRead)
AdvertisedWindow alıcı taraf pencere boyutu olarak ifade edilir (RWND).

TCP – Akış denetimi-Sliding window

Gönderici maksimum segment boyutu (Sender's Maximum Segment Size - SMSS);

SMSS = L2'de MTU boyutu olarak işlenir

CongestionWindow = Tıkanıklık pencere boyutu

If SMSS > 2190 bytes:

CongestionWindow = 2 * SMSS bytes

If (SMSS > 1095 bytes) and (SMSS <= 2190 bytes):

CongestionWindow = 3 * SMSS bytes

If SMSS <= 1095 bytes:

CongestionWindow = 4 * SMSS bytes

İnternette yaygın olarak SMSS, 1500 – 20 (IP) – 20 (TCP)= 1460 byte

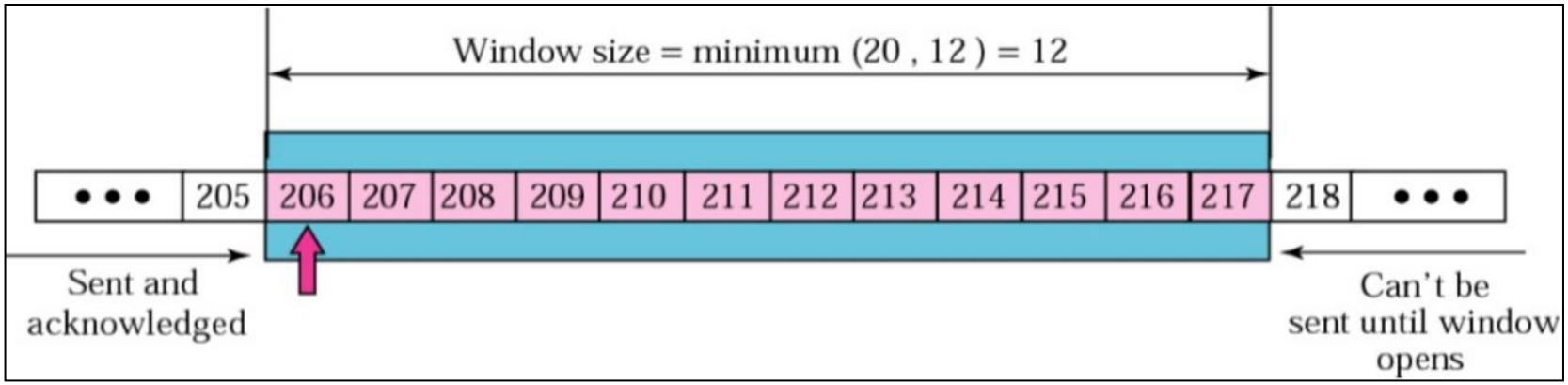
CongestionWindow = 3*SMSS = 4380 bytes olarak kullanılır.

WindowSize = minimum (CongestionWindow**,** AdvertisedWindow**)**

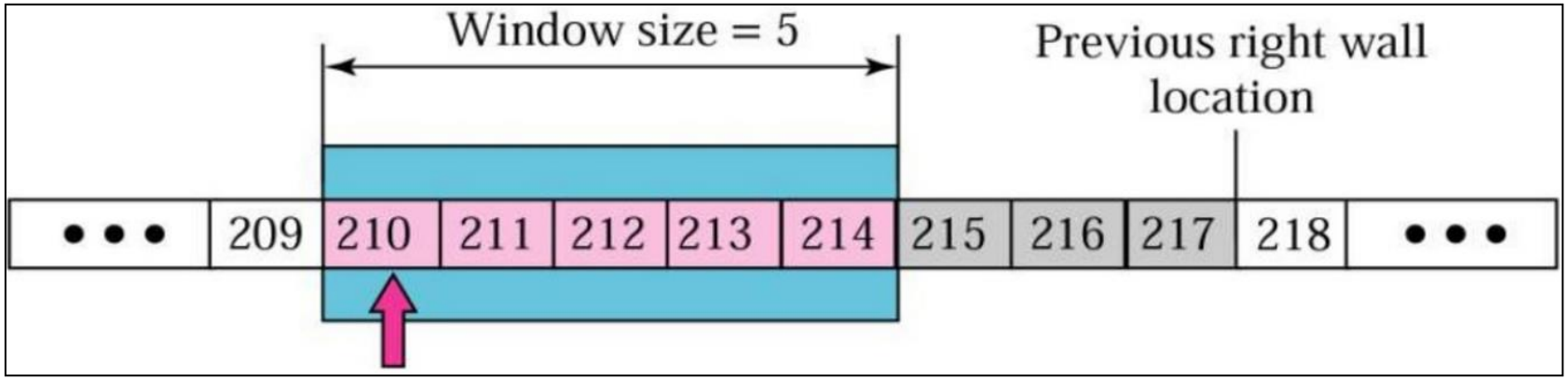
Düşük olan pencere boyutu olarak kullanılır.

TCP – Akış denetimi-Sliding window

Soru: Gönderici ack değeri 210 ve AdvertisedWindow değeri 5 olan olan bir paket alıyor (CongestionWindow =20). Düğüm, 206,207,208 ve 209. byte'ları gönderiyor. Yeni pencere nasıl olur?



Cevap: Yeni WindowSize = minumum(CongestionWindow, AdvertisedWindow) olduğundan 5 olur. Bu nedenle gönderici 215-217 nolu byte'ları göndermez.



TCP – Tıkanıklık Denetimi

Gönderen düğüm tarafından, iletim ortamının kullanılabilirlik durumunu belirlemek yada ardıl olarak gönderebileceği segment sayısını belirlemeye çalışmaktır.

TCP'de **tıkanıklık kontrolü** kısaca, her bir kaynağın (gönderici), ağ kapasitesini ne kadar kullanabileceğini belirlemesidir.

TCP, ağ kapasitesini belirlemek için ACK paketlerini kullanır. İletim onayı olarak gelen ACK paketleri ağın yeni iletimler için güvenli (tıkanıksız) olduğunu söylemektedir.

Temel Terimler

RTT (round trip time): Bir segment gönderim anı ile ACK alımı arasındaki süre.

MSS (maximum segment size): Bir segmentin alacağı en fazla boyut (MTU-kapsüller)

CongestionWindow (cwnd): Her TCP bağlantı için kaynak(gönderen) tarafından belirlenen değişken.

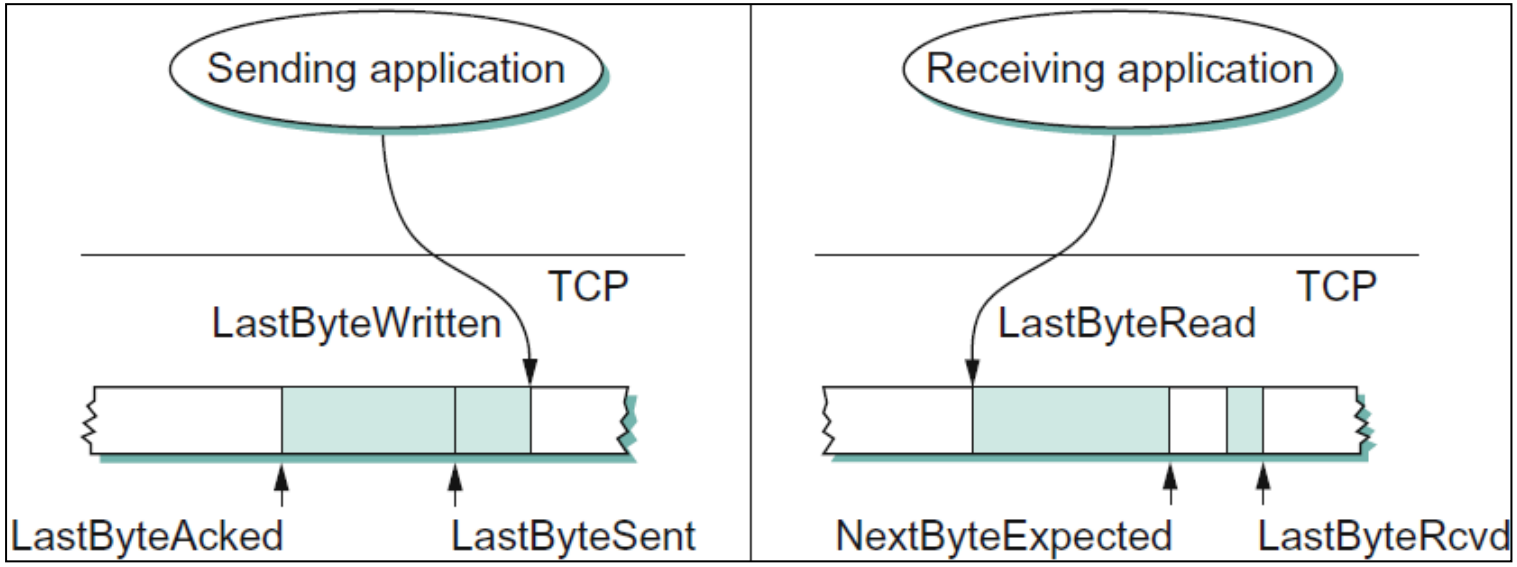
AdvertisedWindow (rwnd): Hedef(alıcı) buffer durumuna göre belirlenen değişken.

ssthresh (slow start threshold): Tıkanıklıktan kaçınma eşik değeri (başlangıçta cwnd)

TCP – Tıkanıklık Denetimi

TCP'de, gönderen düğüm tıkanıklık kontrolü için CongestionWindow ve AdvertisedWindow pencere boyutlarının küçük olanını alır ve Etkin pencere boyutunu hesaplar.

MaxWindow = min (CongestionWindow , AdvertisedWindow)
EffectiveWindow = MaxWindow – (LastByteSent -LastByteAcked)



TCP – Tıkanıklık Denetimi

AIMD (Additive Increase / Multiplicative Decrease):

Ağda mevcut kapasiteyi etkin kullanabilmek için tıkanıklık penceresi artırılır. Her iletilen frame için gelen ACK sonrasında TCP, CongestionWindow değerini bir artırır. Bu durum **Additive Increase (AI)** olarak tanımlanır. Uygulamada bu durum aşağıdaki gibi gerçekleşir.

increment = MSS x (MSS /cwnd)

cwnd = cwnd + increment

Bir paket tıkanıklık nedeniyle iletilemediğinde, yada ACK alımı zaman aşımına uğradığında, gönderici iletim hızını azaltır. Bu azaltım, CongestionWindow değeri mevcut iletim hızının yarısı şeklinde gerçekleşir ki bu kavram **Multiplicative Decrease (MD)** olarak ifade edilir. Uygulamada CongestionWindow değeri 1 MSS'ye indirilir.

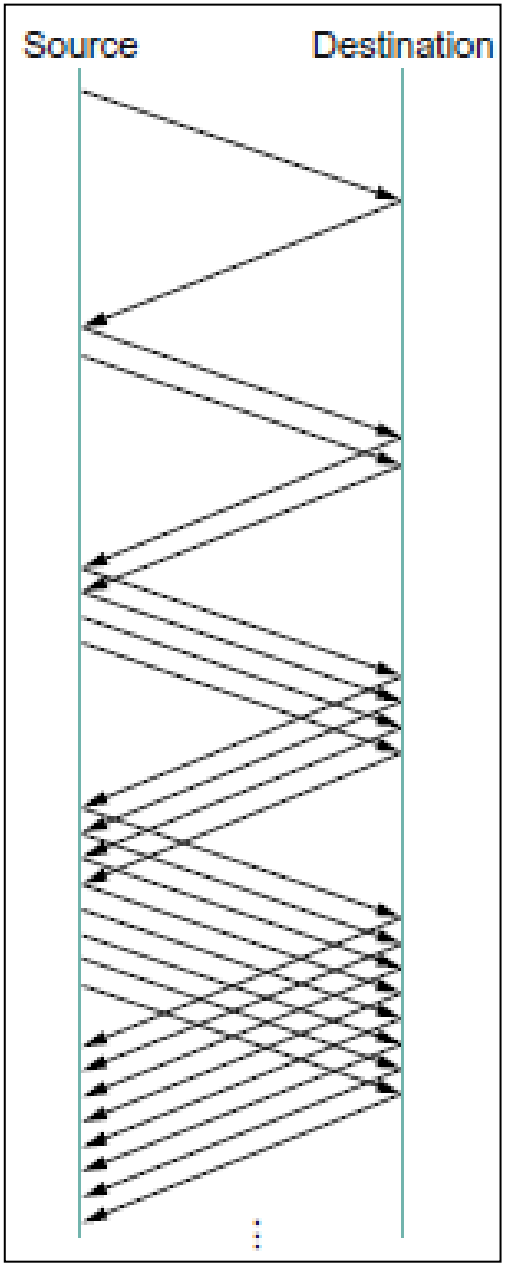
TCP – Tıkanıklık Denetimi

Slow Start:

Additive Increase (AI) yöntemi başlangıçta ağın mevcut kapasitesine ulaşmak için yavaş bir geçişe neden olduğu için TCP, ismi ile pek uyuşmasada **slow start** olarak isimlendirilen mevcut ağ kapsitesine çok daha hızlı ulaşmayı sağlayan bir teknik kullanmaktadır.

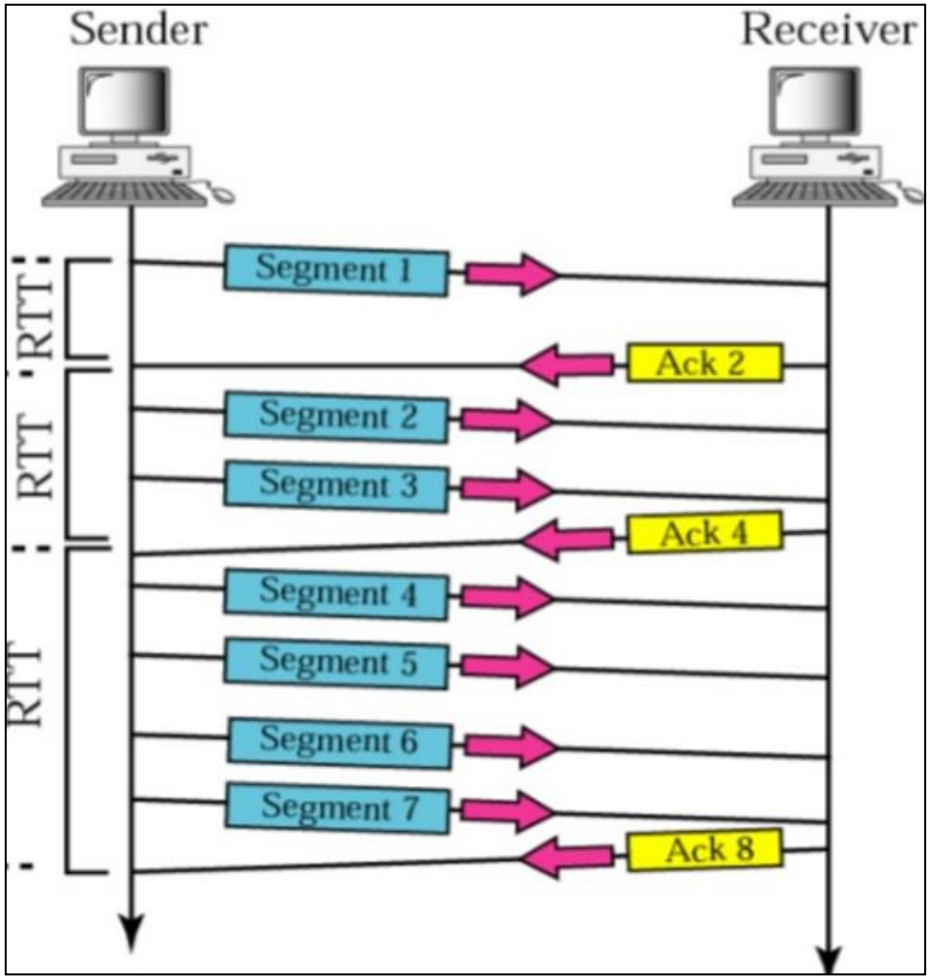
TCP Tahoe ile TCP'ye ilave edilmiştir ve AI'nın lineer artışı yerine üstel artış getirerek ağın kapasitesine ulaşmayı hızlandırmıştır.

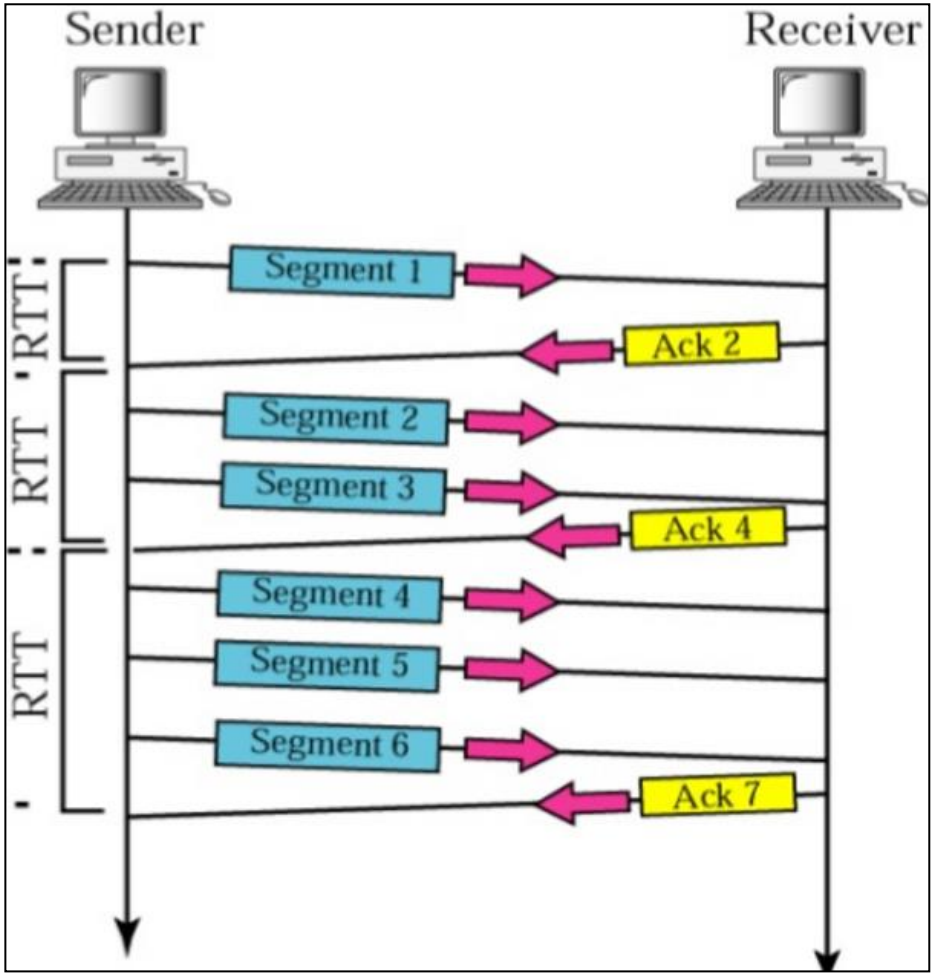
Yani yavaş başlangıç (slow start), TCP'deki cold start problemini çözmektedir.



TCP – Tıkanıklık Denetimi

Slow Start / Additive Increase (AI)

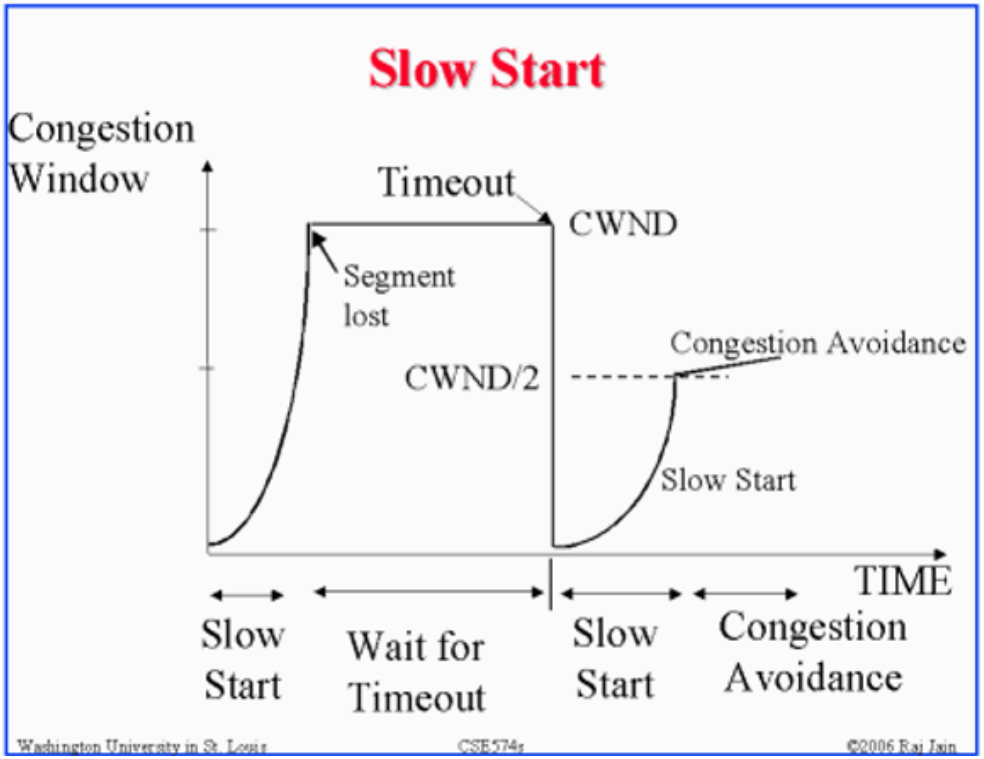




TCP – Tıkanıklık Denetimi

congestion threshold - Slow Start threshold (ssthresh):

Yavaş başlangıç problemini çözmek için geliştirilen **Slow Start**, hızla CongestionWindow değerine ulaşmayı hedeflemektedir. Bu değere ulaşıldığında artırımın **Additive Increase (AI)** olarak lineer devam etmesi istenir. Bu eşik değeri, **congestion threshold** yada **ssthresh** olarak isimlendirilir. **Ssthresh** değeri tıkanıklık algılandığında ağın o anki CongestionWindow değerinin yarısı olarak güncellenir.

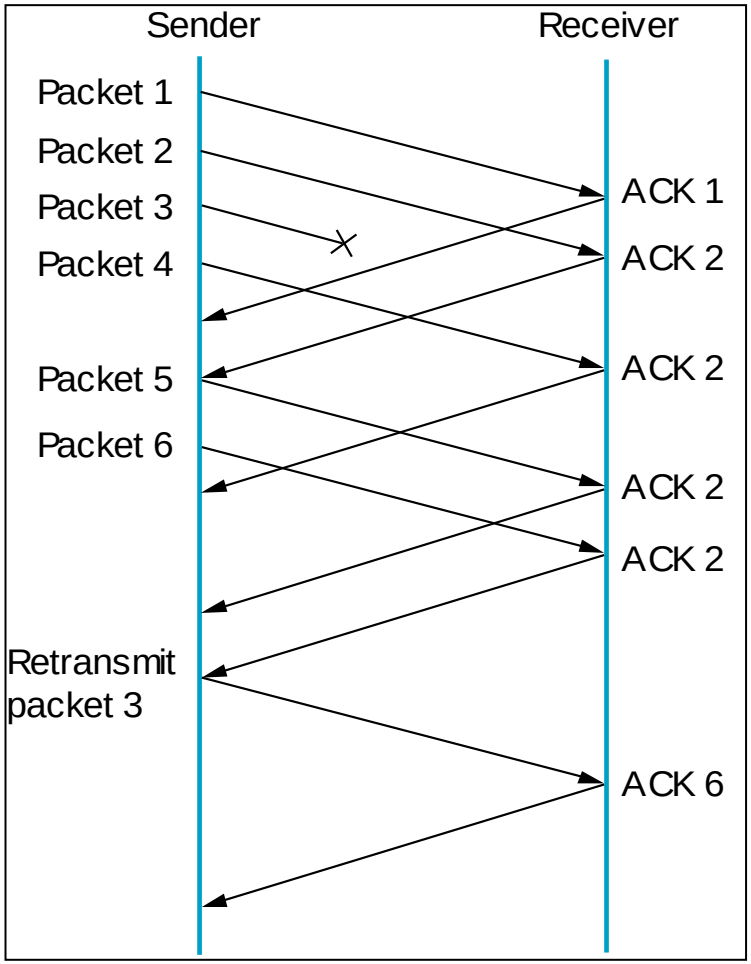


TCP – Tıkanıklık Denetimi

Fast Retransmit:

Zaman aşımı süresinin beklenilmesi bağlantının kopması gibi çok daha kötü sonuçlar doğurmaktadır. **Fast Retransmit,** normal zaman aşımı mekanizmasını beklemek yerine paket kayıplarını daha erken tespit ederek tıkanıklığı sezen bir tekniktir.

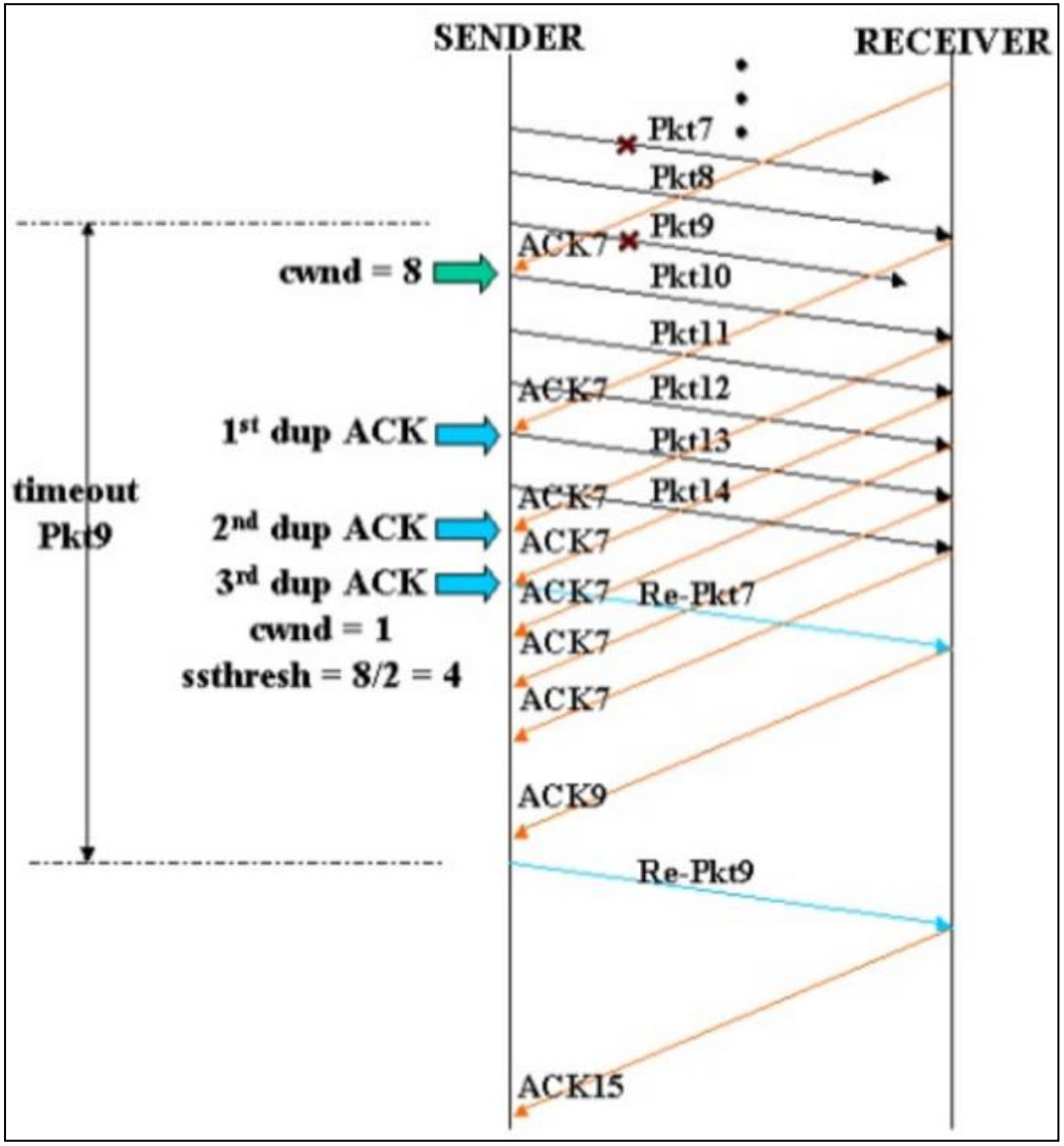
Bu teknik, 3 kez aynı ACK alındığında, ilgili paketi yeniden gönderir. **TCP Tahoe** ile TCP'ye eklenmiştir.



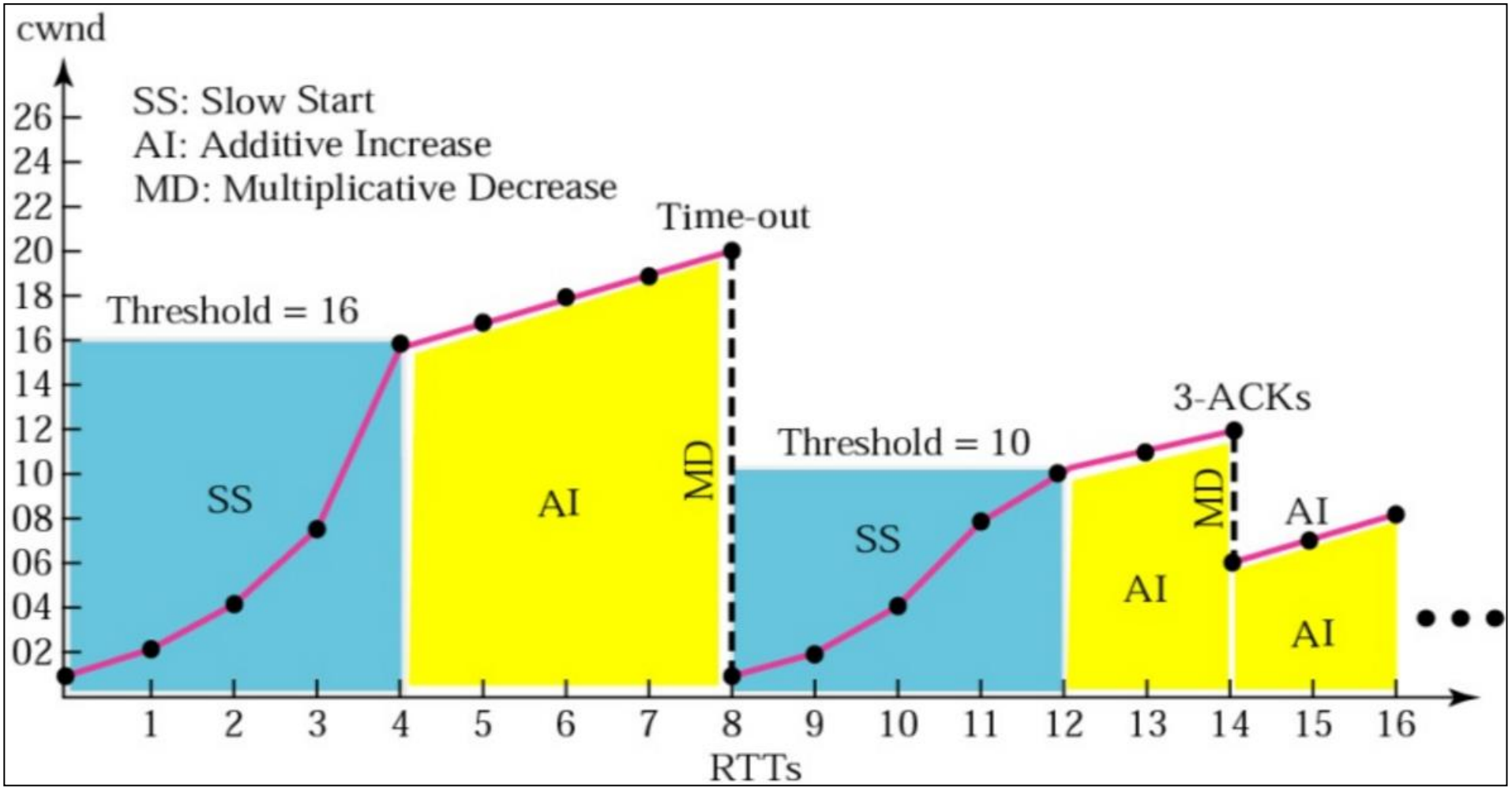
TCP – Tıkanıklık Denetimi

Fast Recovery:

Fast Retransmit durumu tespit edildiğinde (3 kez aynı ACK geldiğinde) **ssthresh** ve cwnd değerini yarıya indirir. **TCP Reno** ile TCP'ye eklenmiştir.



TCP – Tıkanıklık Denetimi



RTP – Gerçek zamanlı aktarım protokolü

IP ağlarında ölçeklenebilirlik, hata ve tıkanıklık kontrolü gibi ihtiyaçlar, gerçek zamanlı aktarım protokolünün doğmasına yol açmıştır.

RTP, UDP aktarım protokülü üzerinden gerçek zamanlı multimedia veri akışını yönetmeyi amaçlamaktadır.

Protokol tasarımcılarının temel amacı güvenilmez bir aktarım katmanı üzerinden sağlam ve gerçek zamanlı multimedia taşınımı için bir mekanizma kurmak olmuştur. Bu noktada temel felsefe uygulama katmanında çerçeveleme ve uçtan-uca presibidir.

Bu prensibe göre ağ yolu boyunca varolan sistemler verinin doğru iletimi için sorumluluk almaları gerekmez. Bu sistemler güvenilir olmayabilir. Veri iletiminin uç noktaları, alt sistemlerin yardımı olmaksızın teslim sürecinin güvenilirliğini sağlamak için uygulama tasarımcısı adına önemli çalışmalar yapmasını gerektirir.

RTP, UDP aktarım protokolü üzerinden gerçek zamanlı multimedya veri aktarımını sağlamayı amaçlamaktadır.

RTP – Gerçek zamanlı aktarım protokolü

RTP paket yapısı

0-1	2	3	4-7	8	9-15	16-31
Ver.	P	X	CC	M	PT	16 bit sıra numarası
32 bit zaman damgası						
32 bit senkronizasyon kaynak kimliği (SSRC)						
32 bit sağlayıcı kaynak kimliği (CSRC)						
Ekstra başlıklar (gerekiyorsa)						
RTP yükü (payload) (…)						

RTP – Gerçek zamanlı aktarım protokolü

1. IP ağlarında iletilen datagram paketleri farklı yönlendiriciler üzerinden iletim sırası değişerek alıcaya ulaşabileceğinden dolayı veri paketlerinin alıcı tarafından doğru sıra ile sıralanabilmesi için sıra numarası sunar.

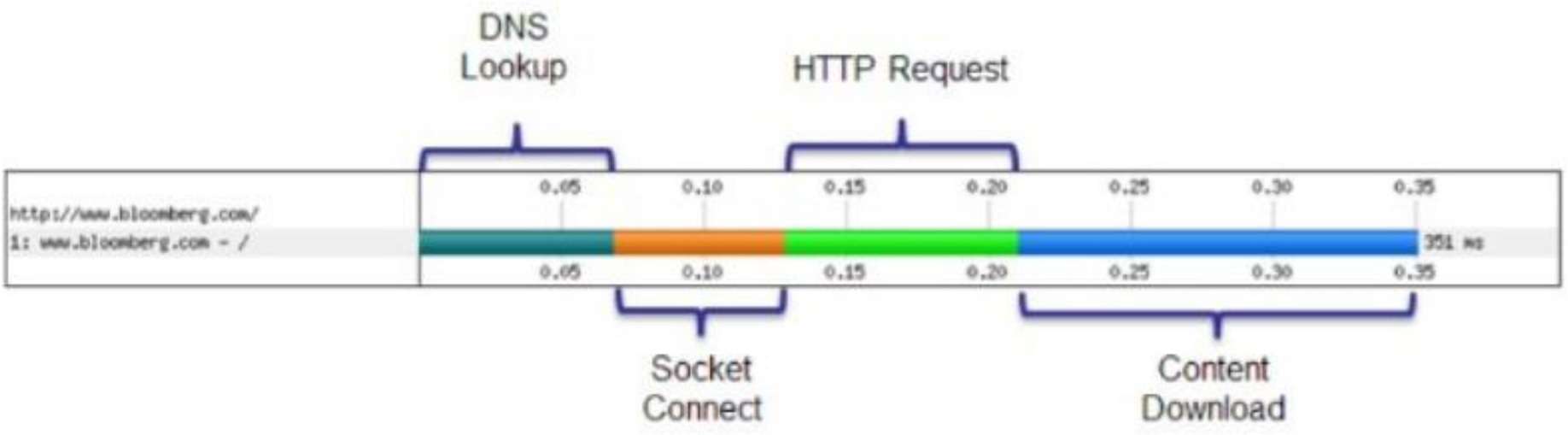
2. Alıcı tarafından veri kaynağının ve yükün tanımlanması için yük içeriğinin türü ve kaynağı tanımlanır.

3. Gerçek zamanlı multimedia içeriğin iletiminde alıcının iletilen verileri doğru sırada alması yeterli olmamaktadır. Sıralamanın yanında zamanlama ve senkronizasyon gereklidir. Örneğin görüntü ve ses verilerinin doğru zamanlama ile senkronize edilmesi gerekir. RTP zamanlama ve senkronizasyon işlevlerini yerine getirebilme işlevselliğine sahiptir.

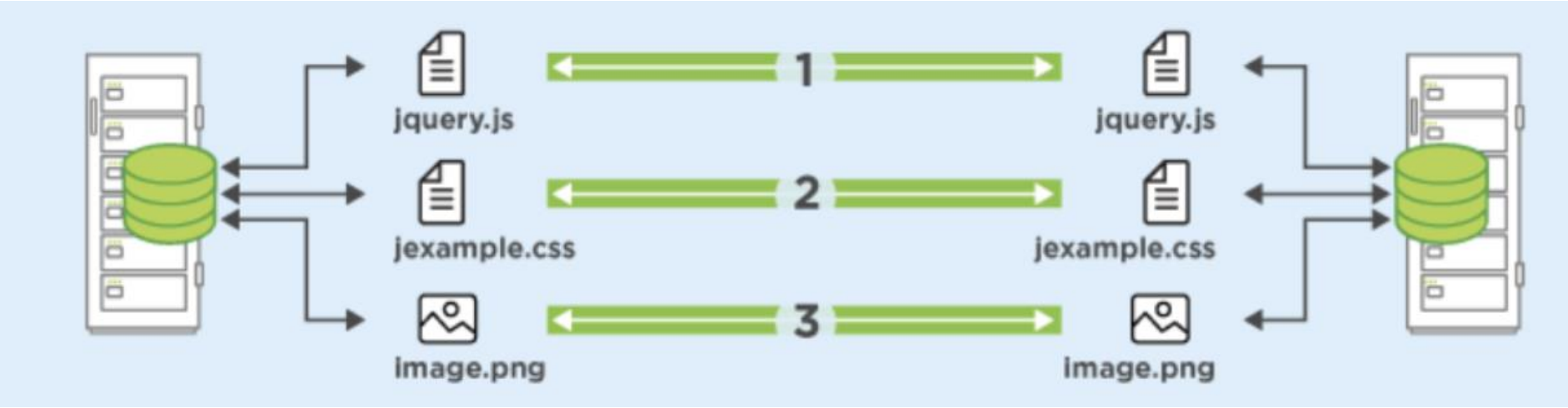
4. RTP, aktarımların izlenebilmesine imkan sunmaktadır. İletilen verinin dağıtım verimliliğini ve kalitesini izleyebilme ve gönderen tarafa sağladığı geribildirim olanakları ile ölçeklenebilirliği ve servis kalitesini sağlamaktadır.

5. RTP, bir RTP oturumu üzerinden farklı kaynaklardan oluşan trafiği tek bir akış halinde birleştirerek, heterojen bir trafik sunabilmektedir.

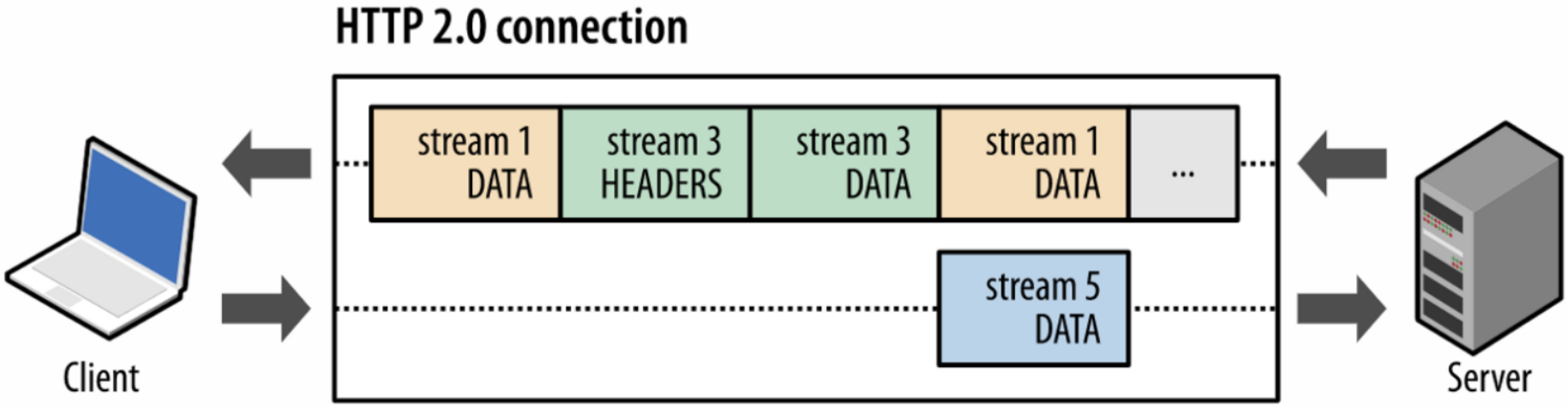
HTTP 1.1 Head-of-Line Blocking



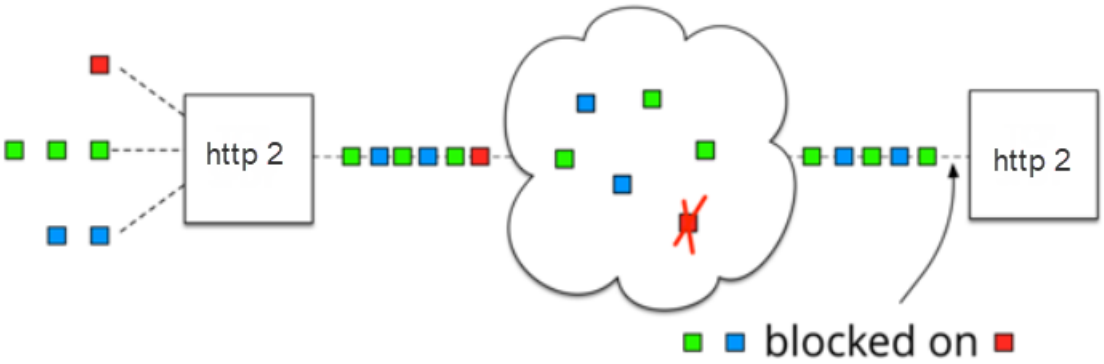
HTTP 1.1 – Multiple Con



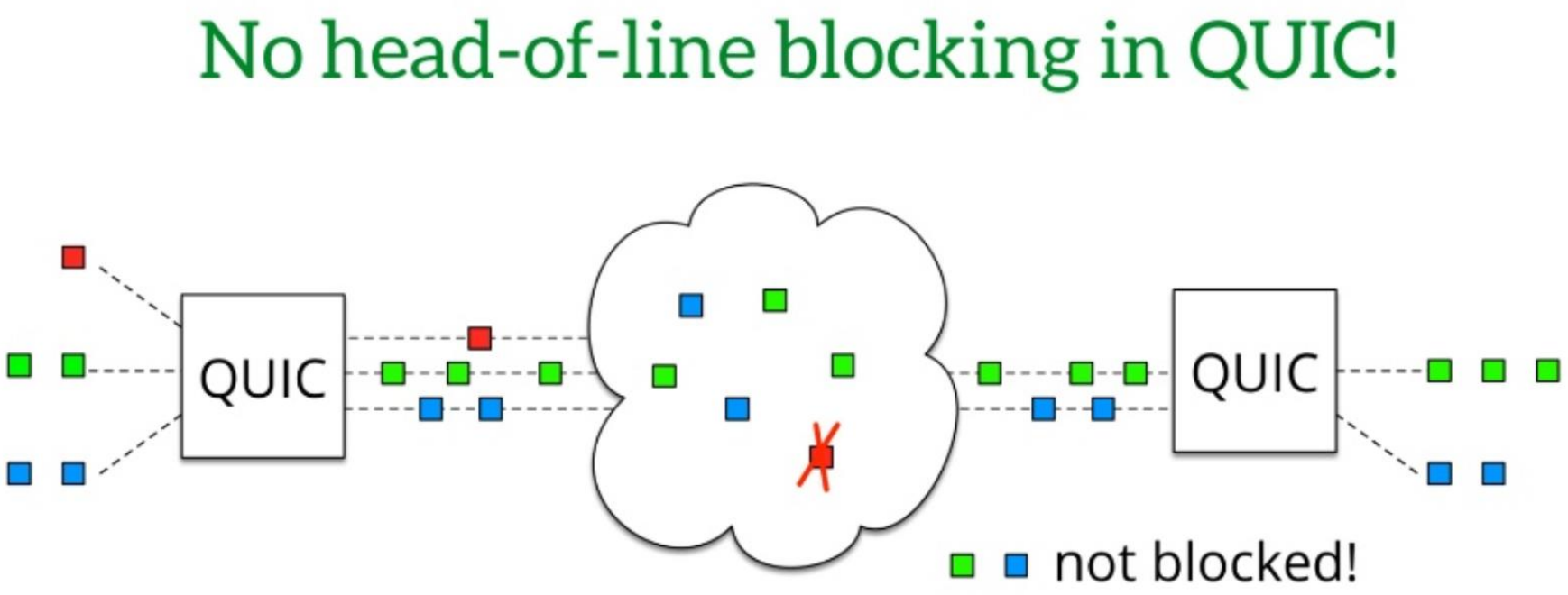
HTTP 2.0 – Multiplexing Stream



- Frame yapısını önerdi.
  - Headers frame,
  - Data frame
- Frame'ler çoklu stream desteği ile tek TCP kanalından iletildi



QUIC



RTCP – RT aktarım kontrol protokolü

RTP, bahsedilen fonksiyonları yerine getirmek için gerçek zamanlı aktarım kontrol protokolüne (RTCP) ihtiyaç duyar.

RTCP, RTP için tasarlanmış ilave bir protokoldür.

RTCP iki temel işleve sahiptir. Bunlardan birincisi, multimedia içerik dağıtım kalitesi hakkında geri bildirim sağlamaktır.

İkinci işlevi ise ses ve video senkronizayonunu sağlamak için RTP akışları için zaman damgalarını eşlemektir.