Soket numaralarıyla iletim sağlanır: iletim katmanı bu iiş görür.

Udp best affort çalışıyor. Paketler kaybolursa bir şekilde telafi etmeya çaışıyordu

Tcp daha güvenli

Udp kullanan protokoller: snmp dns

Güvenilir veri transferi ağ ile ilgili en önemli 10 konudan biridir.

Rdt1.0 da güvenilir kanal kullanılır. **Unrelaible\_chanel()**e uğramaz

Rdt2.0: paket bozulduysa NAK gönderir paket bozulmadıysa iletir ve karşı tarafa acknowlement(ack) iletir.Sürekli beklemeler olduğu için performansı düşük

Rdt2.1 ile bu sorunun çözümü sıra numarsı eklenir. Ack bozulursa ek tedbir alınmış olur. 1, 2, 3, 4 şeklinde daireler vardı. Ack gelir sorun çıkmazsa 1. Paketi gönderir.

2.1 de karmaşıklık vardı. 4 ayrı durum içinde kod yazmamız gerektiğinden karmaşık bir işlem oluşur.

2.2 : karmaşıklık bir nebze aşılmıştır. 2.1e göre Nak kullanılmamıştır. Zaman burada kötü kullanılmıştır. Timer yok

Rdt 3.0 ile Bekleme süresi hesaba katılmıştır. Timer eklenmiştir. Verimsizdir. RTT aralığında bir tane paket gönderiliyor. Verimsizdir. Çözüm önerileri geliştirilmiştir. Piplening birden fazla dosyayı aynanda gönderilmek fkri ortaya atılmış, zamandan tasarruf edilmiştir. Pakaet kaybolunca gidince, dönünce durumlarını incelediğimiz olay..Selective repeat vb

Piplined protocols: Tek bir hat üzerine birden çok verinin gönderilmesi.

TCP fast retransmint: triplle ack gelirse (aynı paekte ayıt), o paketi hemen geri gönderiyordu

Relaibledata transfer karşı tarafa sıkıntısız bir şekilde veriyi iletilmeini sağlar. Sonlu durum daireleri ile çözüyorduk (udt\_send(),extract(packet,data)).

Acknowladgements ile sağlanır -> ben bu gönderilen paketi aldım bilgisini bize verir.

İki yönlü ack handshaking sıkıntılı. Üç yönlü anlatıldı.

Sıkışıklık kontrolündeki temel sorun: bir ağda çok fazla kaynak var. Hepsini göndermeye çalışınca bir sıkışıklık ortaya çıkar. **Bu ağ ile ilgili en önemli 10 problemden biridir.**

**Paketleri**  zamanından iletemezse routerlerdeki bufferlar dolar, sistem yavaşlar. Buffer sonsuz olsa sorun olmaz.

Buffer olayları:

TCP congestion control(sıkışıklık kontrolü):

Tcp nin sıkışıklığı azalmak için yöntemi: toplamsal artırım çarpımsal azaltım. **Kaşıkla verdim kepçeyle geri aldım 😊. Paket kaybı sinyali gelene kadar y yi arttırır. Geldiğinde ise yarıya düşürür.**

**TCP de bu** olaylar nasıl gerçekleşiyor?

Tcp slowstart: Clientler anlaştıktan sonra paket gider

Cwnd(cong. Window ) = 1mss

32 den 16 ya düşürdüysek tcpyi, triple acknowledgment mı timeout mu sebep olur?

Triple ack olur.(

32 den 1 e düşseydi time out olurdu(thoma)

TCP:detecting, reacting to loss

Eğer tcpde kayıp varsa-> geribildirim yok, timeout

Timeout olursa segment daha az olması için pencere boyutunu küçültür.(geçen haftaki animasyon)

Sstresh(eşik seviyesi) sınırından sonra cwnd birer birer artar.

Grafikte 12 yi geçtiğinde triple ack gelirse 6 y düşürür. Tcp Reno gelirse triple veya timeout olması önemli olmadan 1e gönderir

Multimedia izleyen kullanıcı, Bantı genişliğini sömürürür.

Oturum 2:

NETWORK LAYER:

Uç birim iletim yapmak isterse gerçekleştirir. Routerler arasında dolaşıp, iletilmesi gereken uç yöne bağlanmaya çalışır. Eğer gittiği router gitmek istediği routerde değilse, router diğer routerlere ileterek, hedef bilgisayarı ve routeri bulur ve iletir.

Network layerin 2 önemli görevi yani fonksyonu vardır.

Routerlerdeki ilgili yönlendirme tablosuna bakarak diğer routere iletim sağlanır. Kontrol tablosunu doldurma işi control plane katmanı ile gerçekleştirlir.

Tcp taho(tcp ssthresh) : üçlü ack ta gelse timeoutta gelse hemen 1 e düşürür.