Bu projede UDP alt yapısı kullanılarak, reliable bir transfer metodu yazılmıştır.

Bunun gerçekleştirebilmesi amacıyla UDP kullanılarak Stop&Wait metodu implement edilmiştir.

Gönderilen paketler 3 ana kısımdan oluşmaktadır (Packet offset | data checksum | data). Bu 3 ana kısmı birbirinden ayırmak amacıyla aralarda delimiter olarak '$' karakteri kullanılmıştır.

Ayrıca gönderici programda belirlenen buffer değeri, alıcı programında belirlenen buffer değerinden daha küçüktür. Bunun nedeni packet'lara eklenen header boyutunun dinamik olmasıdır. Bu nedenle gonderici buffer'ı için bu dinamikliği kaldırabilecek bir değer olan 2000, alıcı içinse 2048 seçilmiştir.

Bizden istenilen programların gerçekleştirilmesi için gerekli olup olmadığını bilemiyoruz ancak biz, gelen datanın bozulup bozulmadığının anlaşılabilmesi için bir de checksum özelliği ekledik. Packet header'ına gönderilecek verinin checksum değeri ekleniyor, alıcı da aldığı verinin checksum değerini hesaplayarak gelen değer ile karşılaştırıyor. Bunun güzel bir özellik olduğunu düşünüyoruz.

---

Stop&Wait metodunu programımızda şu şekilde implement ettik:

Gönderici için 4 farklı ACK durumu belirlendi. Bunlar:

1 = OK (Packet basarili bir sekilde alindi)
2 = Packet Corrupted (Alinan packet gonderilen ile ayni degil)
3 = Packet Wrong Order (Packet dogru sirada alinmadi)
None = Timeout (2 farkli durum mevcut olabilir. Ya alici taraf
packeti belirlenen timeout degeri icerisinde almamistir,
ya da alici tarafin gonderdigi ack bilgisi kaybolmus/
gecikmistir. Bu nedenle de cagrilan fonksiyona None bilgisi
gonderilir.)

Karşı tarafa bir veri gönderildikten sonra gönderici, 'Wait ACK' durumuna geçer ve alıcıdan bir ACK bekler. Eğer aldığı ACK 1 ise, bir sonraki packet gönderilir. Eğer 1'den farklı bir değer ise aynı data tekrar gönderilmeye çalışılır.

Her aktarımdan sonra bu Wait ACK durumuna geçilir.

---

Alıcı program ise şu şekilde çalışmaktadır:

Alınan packet, aralara konulmuş olan '$' işareti yardımıyla 3 ayrı parçaya ayrılır. Bunlar offset, checksum ve data dır.

Alınan verinin ilk olarak doğru sırada olup olmadığı kontrol edilir. Bu işlem için internal bir değişken tutulur ve bu değişken, gelen offset ile karşılaştırılır. Eğer sıra doğruysa paketin bozulup bozulmadığı checksum kullanılarak kontrol edilir. Pakette bozulma yoksa ACK olarak 1 değeri karşı tarafa gönderilir.

Eğer paket yanlış sıradaysa öncelikle duplicate durumu kontrol edilir. Duplicate varsa, gelen veri dosyaya eklenmeden pas geçilir. Eğer duplicate yoksa, sıra yanlıştır bu nedenle ilgili ACK değeri karşı tarafa iletilir.

---

İlk projede yazmış olduğumuz programların ve yeni programın zaman ölçümleri şu şekildedir:
(~10MB lık pdf dosyası üzerinde test yapılmıştır, veriler saniye cinsindendir)

TCP = [7.3, 8.9, 9.1, 8.9, 9.6, 9.5, 7.7, 8.3, 7.9, 8.6]
- Min: 7.3
- Max: 9.6
- Avg: 8.58

UDP = [4.8, 5.0, 4.9, 4.7, 4.9, 5.2, 5.3, 4.9, 5.5, 4.6]
- Min: 4.6
- Max: 5.5
- Avg: 4.98

Yeni Protokol = [23.3, 26.0, 22.6, 36.6, 37.1, 22.4, 23.0, 25.9, 26.3, 24.2]
- Min: 22.4
- Max: 37.1
- Avg: 26.74

İlk projemizin sonucunda, raporunda belirttiğimiz üzere TCP hatasız bir şekilde transfer yaparken UDP bazen dosyayı eksik olarak aktarıyordu. Bu nedenle TCP güvenilirken UDP güvenilir bir transfer protokolü değildi. Hız olarak TCP, UDP ‘den yavaş kalıyordu ki bu da hız ve güvenilirliğin bir trade-off olduğunu göstermekteydi.

Bizim implement etmiş olduğumuz protokol de güvenilir bir protokoldür. Yapılan testler neticesinde herhangi bir aktarım sorunu gözlemlenmemiştir. Dolayısıyla TCP gibi güvenilir olduğu söylenebilir.

Ancak test sonuçları, oldukça yavaş bir protokol olduğunu da göstermektedir. Yazmış olduğumuz program, TCP’den ~3 kat, UDP’den ise ~5 kat yavaş çalışmaktadır.

Alicinin ve gonderici farkli OS'ler uzerinde calisiyor ise, gonderilen byte ve alinan byte sayilari farkli gozukebilir. Ornegin her 2 program da Windows uzerinde calistirilirken byte gosterimlerinde herhangi bir sorun gozukmez iken, Gondericinin Windows alicinin ise Linux oldugu denemelerde gonderici 2037 byte gonderirken alici ayni verinin 8032 byte olarak alindigini gosterebiliyor. Saniyorum bu durum encode/decode islemi yuzunden olusabiliyor. Ancak bu durum dosyanin alinmasini kesinlikle etkilemiyor. Dosya basarili bir sekilde karsi tarafa iletilebiliyor.