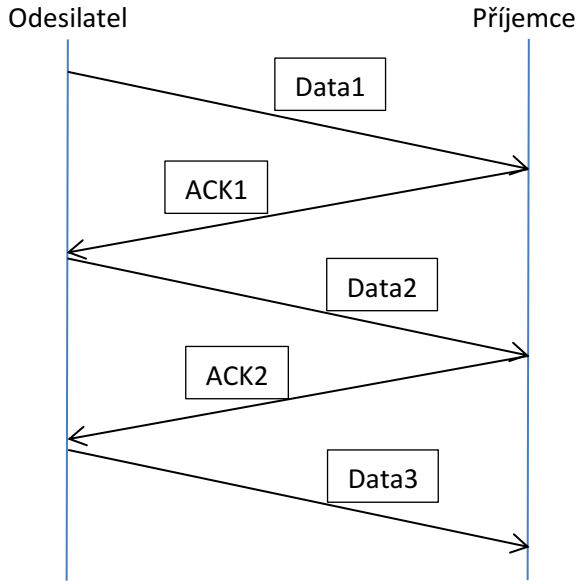
Metoda *Stop and Wait ARQ* (*Automatic Repeat Request*)

Je třeba splnit následující požadavky:

1. Každý odeslaný datový paket obsahuje kontrolní součet (např. CRC), aby příjemce mohl detekovat chyby v přijatých datech
2. Každý odeslaný datový paket obsahuje pořadové číslo, aby příjemce mohl jednoznačně detekovat tentýž datový paket, přijatý dvakrát (a nepřijal tedy duplikát jako další blok dat)
3. Každé potvrzení odeslané příjemcem odesilateli ( ACK - *Acknowledge*) obsahuje pořadové číslo potvrzovaného datového paketu

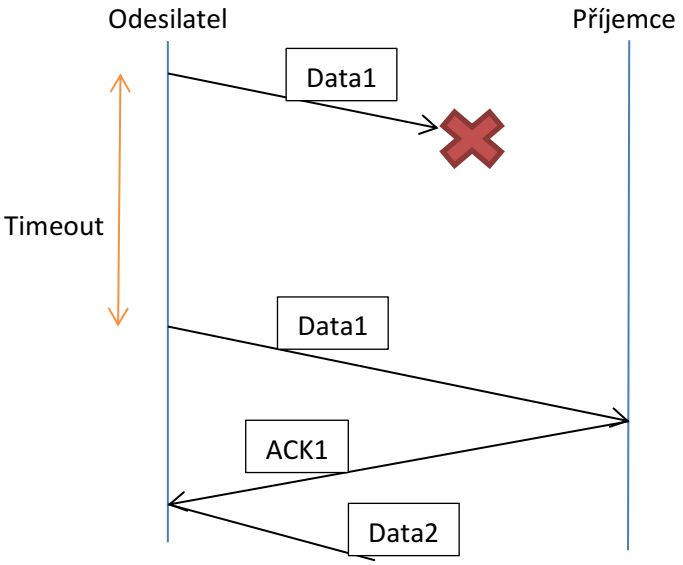
Komunikace pak probíhá následovně:

1. Bezchybný provoz



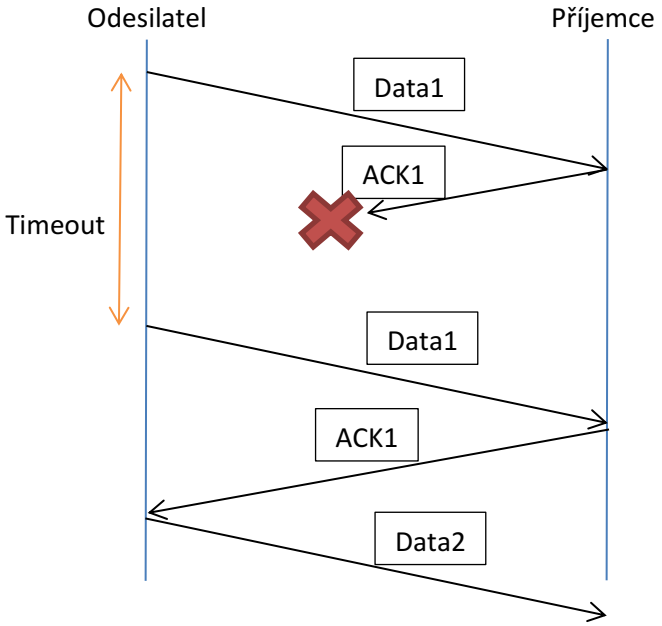

atd.

2. Pokud je v datovém paketu příjemcem detekována chyba nebo se datový paket cestou ztratí



S odeslání paketu startuje odesilatel měření časového intervalu (Timeout), v němž je očekáván příjem potvrzení. Pokud do dané doby potvrzení nedorazí, je stejný datový paket odeslán znovu.

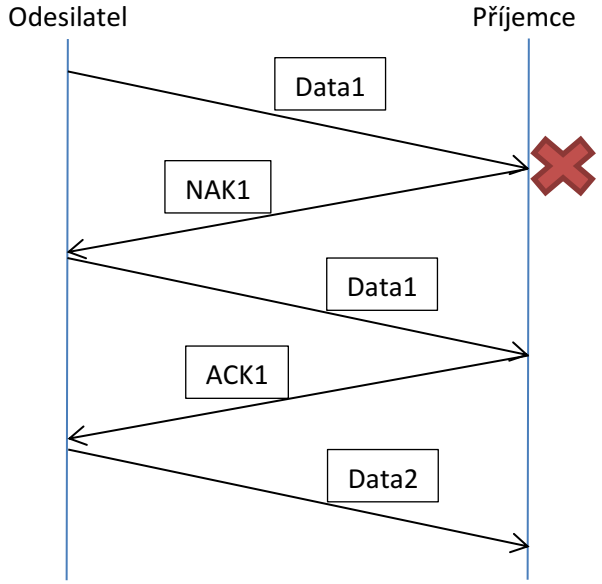
3. Může ale nastat i situace, kdy se cestou zpět ztratí potvrzení (nebo se poškodí vlivem rušení, takže ho odesilatel nerozpozná)




Všimněte si, že díky číslování paketů je příjemce při opakovaném příjmu paketu Data1 schopen detekovat, že tento paket již má přijatý. Obsah paketu tedy může ignorovat, ale musí znovu poslat potvrzení odesilateli, aby ten mohl pokračovat v odesílání dalších paketů.

Zkuste si sami odvodit, jak velký rozsah pořadových čísel potřebujete pro číslování paketů během přenosu. Na první pohled to vypadá, že pokud se přenos skládá z 1000 paketů, potřebujeme číslovat do 1000. Ale pozor – číslování používáme proto, abychom rozpoznali opakovaný příjem již správně přijatého paketu. Kolikabitové číslo tedy (pro dosažení tohoto cíle) potřebujeme? Začněte od nejmenšího a uvidíte, že to nebudete řešit dlouho ☺.

4. Rozšíření o implementaci negativního potvrzení (NAK – *Negative Acknowledge*)



Zde nastala situace, kdy příjemce detekoval chybu v přijatých datech. Ve standardní implementaci metody Stop and Wait nepošle žádné potvrzení a nastalou situaci vyřeší Timeout na straně odesilatele. Ten je ale typicky nastaven na poměrně dlouhý časový interval a komunikace se tak na tuto dobu zcela zastaví. Rozšíření základní metody o negativní potvrzení tuto časovou prodlevu zkracuje (odesilatel se o chybě dozví dříve, než na základě Timeoutu).

Poznámka k bonusové části úlohy

Cílem je vytvořit model této komunikace v situaci, kdy nedochází k chybám, a využít ho k předpovědi, jaká bude (při vámi zvolené rychlosti na fyzické vrstvě a velikosti datových a potvrzovacích paketů) závislost rychlosti přenosu dat na zpoždění mezi komunikujícími uzly. Tuto závislost lze vyjádřit analyticky jednoduchým vzorcem – nebojte se se do toho pustit. Následně si předpovědi poskytované vaším modelem můžete prakticky ověřit s využitím programu NetDerper, který do vaší komunikace vnese definované zpoždění.

Pro ty z vás, které modelování zaujme, mám ještě další námět (nad rámec bonusu) – zkuste do modelu zahrnout i vliv chybovosti přenosu (vyjádřený jako pravděpodobnost chyby či ztráty datového paketu a/nebo potvrzení). I zde vám program NetDerper poskytne možnost ověřit správnost modelu.