۱- الف)

در اتصال TCP بین دو میزبان، شماره پورت‌های میزبان ارسال کننده و گیرنده در هر دو سمت تعیین می‌شود. در واقع، هر بسته TCP دارای اطلاعاتی در سربرگ خود است که شامل شماره پورت مبدا (ارسال کننده) و شماره پورت مقصد (گیرنده) است. این اطلاعات در فرآیند ساختار دهی و ارسال بسته‌های TCP توسط سیستم عامل میزبان ارسال کننده قرار می‌گیرد و در هنگام دریافت بسته‌ها توسط سیستم عامل میزبان گیرنده، این اطلاعات از سربرگ بسته استخراج و استفاده می‌شود تا بسته به درستی روت شود و به برنامه مورد نظر تحویل داده شود. بنابراین، شماره پورت مبدا در اتصال TCP بین دو میزبان به راحتی قابل شناسایی است.

شبکه های TCP/IP از پورت ۳۳۶۰ استفاده میکنند. برای ارسال هر نوع داده هم پورت های مختلفی داریم.

Application	Port number
FTP- transfer files between a client and a server	20 -21
Secure Shell	22
SMTP- email	25
IMAP	993
DNS	53
HTTP- internet	80
Network Time Protocol	123
Border Gateway Protocol (BGP)	179
HTTPS	443
Internet Security Association and Key Management Protocol	500
Remote Desktop Protocol	3389
WEB SERVER	8080

ب)

یکی از دلایل استفاده از UDP به جای TCP برای انتقال تصویر و فایل های صوتی این است که latency UDP کمتری نسبت به TCP دارد. latency تأخیر بین ارسال داده و دریافت آن است و می تواند عامل مهمی در برنامه های بلادرنگ مانند کنفرانس صوتی و ویدیویی باشد. از آنجایی که UDP سربار مکانیزم های قابلیت اطمینان مانند کنترل جریان و ارسال مجدد را ندارد، می تواند داده ها را با تاخیر کمتری نسبت به TCP ارسال کند.

دلیل دیگر این است که UDP سرعت انتقال سریع‌تری را برای فایل‌های بزرگ‌تر فراهم می‌کند. هنگام استفاده از TCP، میزبان فرستنده منتظر تایید هر بسته ای است که ارسال می کند قبل از ارسال بسته بعدی. این فرآیند می‌تواند سرعت انتقال فایل‌های حجیم را کاهش دهد که می‌تواند هنگام ارسال فایل‌های تصویری یا صوتی بزرگ در زمان واقعی مشکل ایجاد کند. از طرف دیگر UDP این محدودیت را ندارد، بنابراین می تواند فایل های حجیم را با سرعت بیشتری انتقال دهد.

UDP پروتکل ساده تری از TCP است و برای پیاده سازی به منابع کمتری نیاز دارد. این بدان معنی است که می تواند انتخاب کارآمدتری برای برنامه هایی باشد که به قابلیت اطمینان و ویژگی های بررسی خطا TCP نیاز ندارند.

UDP از انتقال چندپخشی و پخش پشتیبانی می کند که می تواند برای انتقال فایل های تصویری و صوتی به چندین گیرنده به طور همزمان مفید باشد. از طرف دیگر، TCP برای انتقال نقطه به نقطه بین دو میزبان طراحی شده است.

با این حال، توجه به این نکته مهم است که UDP هیچ گونه ضمانت بررسی خطا یا قابلیت اطمینان ارائه نمی دهد، بنابراین خطر از بین رفتن یا خراب شدن داده ها هنگام استفاده از UDP وجود دارد. بنابراین، هنگام انتقال فایل های تصویری یا صوتی از طریق UDP، استفاده از مکانیسم های تشخیص و تصحیح خطا در لایه برنامه برای اطمینان از یکپارچگی داده ها مهم است.

پ)

کاملا بستگی به برنامه دارد. اگر دریافت همه پکت ها در برنامه حائز اهمیت باشد نمیتوانیم از این پروتکل استفاده کنیم مثل برنامه هایی که دانلود می کنند. اما اگر این مسئله چندان اهمیت نداشته باشد و سرعت ارسال و دریافت مهم تر باشد میتانیم از این پروتکل استفاده کنیم. حتی برای بهینه کردن و بالا بردن دقت دریفت میتوان از سایر مکانیزم ها برای بررسی دریافت کامل فایل استفاده نمود. مثلا در برنامه های سوشال مدیا که قابلیت تماس صوتی دارند سرعت ارتباط مهم است و میتوان از UDP استفاده کرد یا مثلا در استریم های گیم هر ثانیه برای گیمر ها مهم است و نمیتوانند منتظر تایید دریافت و ارسال دیتا بمانند و میتوان از UDPاستفاده نمود.

ت)

پروتکل RDT یا Reliable Data Transfer، یک پروتکل ارتباطی است که به منظور انتقال داده های قابل اعتماد بین دو ماشین مجازی یا دستگاه ارتباطی استفاده می شود. در این پروتکل، اعداد توالی و تایمر به منظور تضمین تحویل داده های دقیق و به موقع به مقصد استفاده می شوند.

علت استفاده از اعداد توالی در پروتکل RDT این است که در ارتباطات شبکه، پیام ها به ترتیبی که فرستنده آنها را ارسال می کند، دریافت نمی شوند. به عبارت دیگر، پیام ها ممکن است به ترتیب نامناسبی دریافت شوند. با استفاده از اعداد توالی، فرستنده می تواند به درستی مشخص کند کدام داده باید به مقصد ارسال شود و مقصد هم می تواند به درستی ترتیب این داده ها را بازسازی کند.

علاوه بر اعداد توالی، تایمر نیز به منظور تضمین تحویل داده های به موقع استفاده می شود. تایمر به عنوان یک ساعت در نظر گرفته می شود که به طور مداوم شمارش می شود و زمانی که یک داده ارسال می شود، تایمر شروع به شمارش می کند. اگر در زمان تعیین شده توسط تایمر، پاسخی از مقصد دریافت نشد، فرستنده فرض می کند که داده قبلی از دست رفته و دوباره آن را ارسال می کند. این کار به تضمین تحویل داده های به موقع و جلوگیری از میس شدن داده ها کمک می کند.

ث)

مکانیسم congestion control درTCP پهنای باند را به طور عادلانه بین اتصالات تخصیص می دهد.

الگوریتم کنترل تراکم TCP برای جلوگیری از تراکم شبکه با تنظیم پویا نرخ ارسال بر اساس شرایط شبکه طراحی شده است. TCP از مکانیزمی به نام " congestion window " (CWND) برای کنترل مقدار داده ای که می تواند قبل از دریافت تایید ارسال شود، استفاده می کند. اندازه CWND به صورت پویا بر اساس سطح تراکم شبکه تنظیم می شود.

با فرض اینکه دو اتصال اندازه congestion window یکسانی داشته باشند، هر اتصال سهم مساوی از پهنای باند موجود را دریافت خواهد کرد. بنابراین، هر اتصال TCP سهمی معادل M/2 bps خواهد داشت. با این حال، اگر یکی از اتصالات congestion window کمتری داشته باشد، در مقایسه با اتصال دیگر، سهم کمتری از پهنای باند موجود را دریافت می کند.

شایان ذکر است که پهنای باند واقعی تخصیص داده شده به هر اتصال ممکن است در طول زمان به دلیل ماهیت پویا کنترل تراکم TCP متفاوت باشد.

۲- الف)

بله، این امکان برای کاربر A وجود دارد که حتی اگر هیچ داده ای را با سایر کاربران به اشتراک نگذارد، فایل کامل را از طریق بیت تورنت دریافت کند. با این حال، عموماً این یک عمل غیراخلاقی در نظر گرفته می‌شود و با اصول اشتراک‌گذاری peer to peer که BitTorrent مبتنی بر آن است، مغایرت دارد.

در BitTorrent، یک فایل به قطعات کوچک زیادی تقسیم می‌شود و هر قطعه در میان peer های مختلف به اشتراک گذاشته می‌شود. هنگامی که یک همتا به swarm می پیوندد و شروع به دانلود یک فایل می کند، به peer های دیگر متصل می شود و قطعاتی را که ندارد درخواست می کند. با پیشرفت دانلود، این peer بخشی از گروه می شود و قطعاتی را که دانلود کرده است با سایر همتایان به اشتراک می گذارد.

هنگامی که یک peer در ازدحام هیچ داده ای را به اشتراک نمی گذارد، به عنوان "زالو" شناخته می شود. این بدان معناست که peer فقط داده‌ها را از peer های دیگر دریافت می‌کند اما به اشتراک گذاری کلی فایل کمکی نمی‌کند. در حالی که ممکن است یک زالو فایل کامل را دریافت کند در صورتی که

همتایان که فایل کامل را دانلود کرده اند و به طور فعال آن را به اشتراک بگذارند، این یک مدل پایدار برای به اشتراک گذاری فایل نیست. اگر همه همتایان در گروه زالو بودند، کسی نبود که فایل را به اشتراک بگذارد و دانلود ممکن نبود.

علاوه بر این، بسیاری از مشتریان BitTorrent اقداماتی را برای جلوگیری از این روند در نظر گرفته اند. به عنوان مثال، برخی از مشتریان ممکن است سرعت دانلود یک peer را که به طور فعال داده‌ها را با دیگران به اشتراک نمی‌گذارد، محدود کنند، یا peerهایی را که در ازدحام شبکه مشارکت نمی‌کنند، به‌کلی منع کنند.

ب)

بله، استفاده از بیش از یک کامپیوتر (یا دستگاه) برای دانلود یک فایل از طریق BitTorrent به طور بالقوه می تواند سرعت دریافت فایل را افزایش دهد، بسته به عوامل مختلفی مانند تعداد Seeder در ازدحام، سرعت اتصال هر دستگاه، تنظیمات مشتری BitTorrent و .... وقتی چندین دستگاه در حال دانلود یک فایل هستند، بخشی از یک گروه می شوند و می توانند قطعاتی از فایل را با یکدیگر به اشتراک بگذارند. این بدان معنی است که هر دستگاه به طور بالقوه می تواند قطعات فایل را از چندین peer به طور همزمان دریافت کند که می تواند سرعت کلی دانلود را افزایش دهد.

با این حال، توجه به این نکته مهم است که افزایش سرعت تضمینی نیست و به عوامل مختلفی بستگی دارد، از جمله تعداد seeder در ازدحام، سرعت دانلود هر دستگاه و سلامت کلی ازدحام. اگر Seeder کافی وجود نداشته باشد، یا اگر سرعت دانلود یک یا چند دستگاه پایین باشد، ممکن است منجر به افزایش قابل توجهی در سرعت دانلود نشود.

علاوه بر این، استفاده از چندین دستگاه برای دانلود یک فایل ممکن است همیشه مجاز نباشد، زیرا برخی از ردیاب‌های BitTorrent ممکن است قوانینی علیه استفاده از چندین دستگاه برای دانلود همزمان یک فایل داشته باشند.

بله، استفاده از بیش از یک کامپیوتر (یا دستگاه) برای دانلود یک فایل از طریق BitTorrent به طور بالقوه می تواند سرعت دریافت فایل را افزایش دهد، بسته به عوامل مختلفی مانند تعداد Seders در ازدحام، سرعت اتصال هر دستگاه و ... تنظیمات مشتری BitTorrent

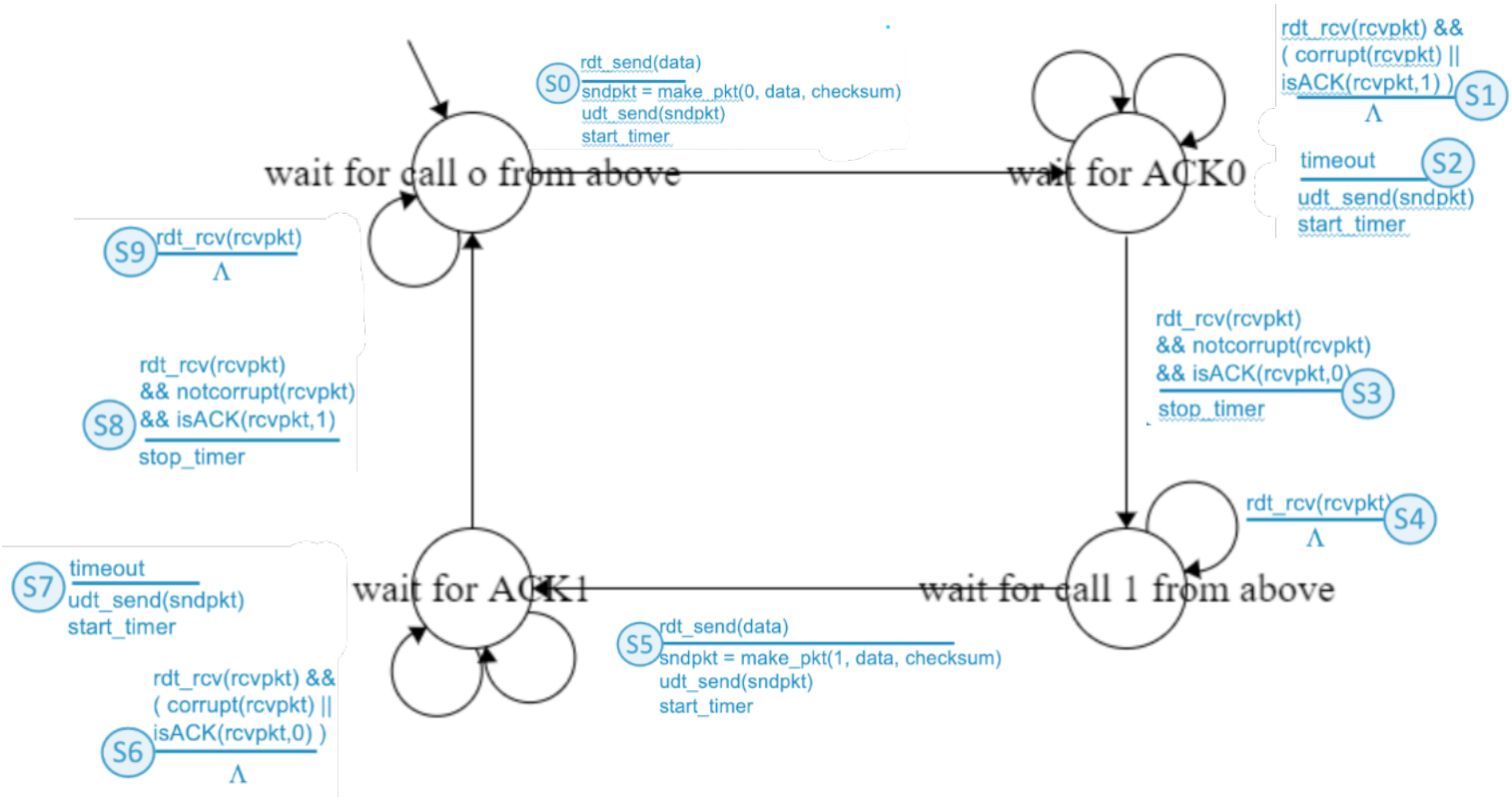
وقتی چندین دستگاه در حال دانلود یک فایل هستند، بخشی از یک گروه می شوند و می توانند قطعاتی از فایل را با یکدیگر به اشتراک بگذارند. این بدان معنی است که هر دستگاه به طور بالقوه می تواند قطعات فایل را از چندین همتا به طور همزمان دریافت کند که می تواند سرعت کلی دانلود را افزایش دهد.

با این حال، توجه به این نکته مهم است که افزایش سرعت تضمینی نیست و به عوامل مختلفی بستگی دارد، از جمله تعداد بذرها در ازدحام، سرعت دانلود هر دستگاه و سلامت کلی ازدحام. اگر Seder کافی وجود نداشته باشد، یا اگر سرعت دانلود یک یا چند دستگاه پایین باشد، ممکن است منجر به افزایش قابل توجهی در سرعت دانلود نشود.

علاوه بر این، استفاده از چندین دستگاه برای دانلود یک فایل ممکن است همیشه مجاز یا اخلاقی نباشد، زیرا برخی از ردیاب‌های BitTorrent ممکن است قوانینی علیه استفاده از چندین دستگاه برای دانلود همزمان یک فایل داشته باشند.

۳-

ماشین حالت فرستنده:



0: حالت اولیه FSM فرستنده، جایی که منتظر می ماند تا داده ها از لایه بالایی منتقل شوند. هنگامی که داده ها در دسترس هستند، به حالت بعدی منتقل می شوند.

1: فرستنده FSM پس از دریافت داده از لایه بالایی وارد این حالت می شود. سپس یک شماره توالی به داده ها اضافه می کند، یک بسته ایجاد می کند و آن را به گیرنده می فرستد. همچنین یک تایمر را شروع می کند تا منتظر یک ACK (تأیید) از گیرنده باشد.

2: اگر فرستنده قبل از انقضای زمان سنج ACK دریافت نکند، بسته را دوباره ارسال می کند و تایمر را ریست می کند. در حالی که منتظر رسیدن ACK است، به این حالت منتقل می شود.

3: هنگامی که فرستنده یک ACK برای بسته ارسالی دریافت می کند، به این حالت انتقال می یابد. تایمر را متوقف می کند و منتظر بسته داده بعدی از لایه بالایی است.

4: اگر فرستنده یک NAK (تأیید منفی) از گیرنده دریافت کند که نشان می دهد گیرنده خطایی را در بسته دریافتی تشخیص داده است، فرستنده به این حالت منتقل می شود. سپس بسته را با شماره دنباله درخواستی مجددا ارسال می کند.

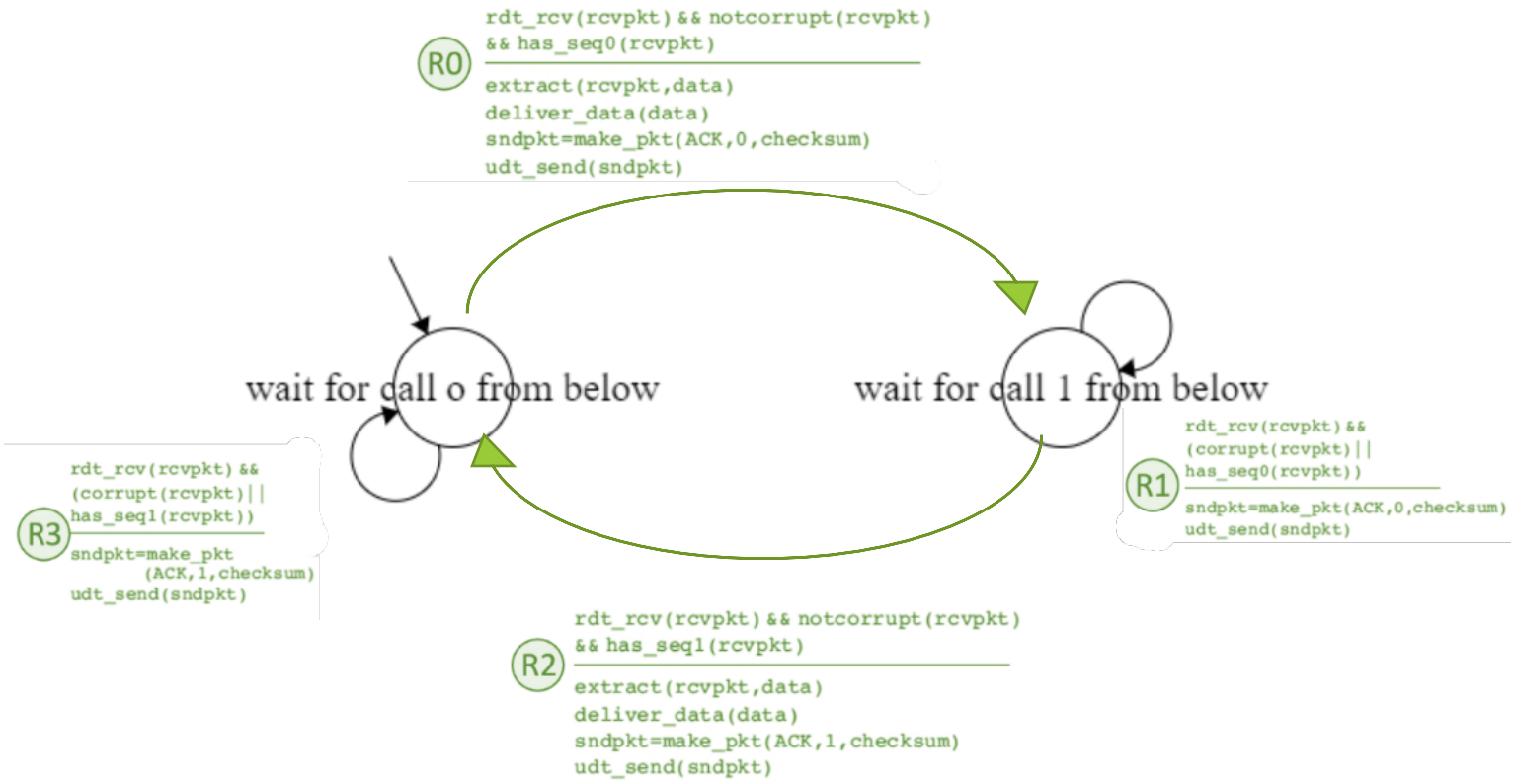
5: اگر فرستنده یک ACK برای بسته ای با شماره دنباله ای بزرگتر از شماره دنباله فعلی دریافت کند، به این حالت انتقال می یابد. شماره دنباله فعلی را به شماره دنباله دریافتی به روز می کند و منتظر بسته بعدی از لایه بالایی است.

6: فرستنده FSM زمانی وارد این حالت می شود که پنجره پر باشد، یعنی چندین بسته را بدون دریافت ACK برای گیرنده به گیرنده ارسال کرده است. قبل از ارسال بسته های بیشتر منتظر می ماند تا ACK برسد.

7: اگر فرستنده یک بسته خراب از گیرنده دریافت کند، به این حالت منتقل می شود. بسته را دور انداخته و منتظر بسته بعدی از گیرنده می ماند.

8: اگر فرستنده یک ACK تکراری از گیرنده دریافت کند که نشان می دهد گیرنده یک بسته را چندین بار دریافت کرده است، به این حالت انتقال می یابد. بسته را با شماره دنباله درخواستی مجددا ارسال می کند.

ماشین حالت گیرنده:



0: حالت اولیه گیرنده FSM، جایی که منتظر می ماند تا یک بسته از فرستنده برسد. پس از دریافت بسته، به حالت بعدی منتقل می شود.

1: در این حالت گیرنده شماره توالی بسته دریافتی را بررسی می کند. اگر شماره توالی مورد انتظار باشد، یک ACK برای فرستنده ارسال می کند و داده ها را به لایه بالایی ارسال می کند. اگر شماره توالی کمتر از عدد مورد انتظار باشد، بسته را دور انداخته و یک ACK تکراری برای فرستنده ارسال می کند. اگر شماره دنباله بزرگتر از عدد مورد انتظار باشد، بسته را دور می اندازد و یک ACK با شماره دنباله مورد انتظار برای فرستنده ارسال می کند.

2: گیرنده زمانی وارد این حالت می شود که بسته ای با شماره ترتیب نامرتب دریافت می کند. یک ACK با شماره دنباله مورد انتظار برای فرستنده ارسال می کند و منتظر بسته بعدی می ماند.

3: اگر گیرنده یک بسته خراب را شناسایی کند، به این حالت منتقل می شود. بسته را دور می اندازد و یک NAK برای فرستنده ارسال می کند که درخواست ارسال مجدد بسته را دارد.

ABP:

پروتکل بیت متناوب (ABP) یک پروتکل ساده کنترل جریان است که برای انتقال مطمئن داده ها از طریق یک کانال یا شبکه نویزی استفاده می شود. بر اساس استفاده از دو مقدار بیت متناوب، معمولاً ۰ و ۱، برای کنترل جریان داده بین فرستنده و گیرنده است.

عملکرد ABP را می توان به صورت زیر توضیح داد:

فرستنده یک بسته داده با مقدار بیت متناوب ۰ یا ۱ ارسال می کند.

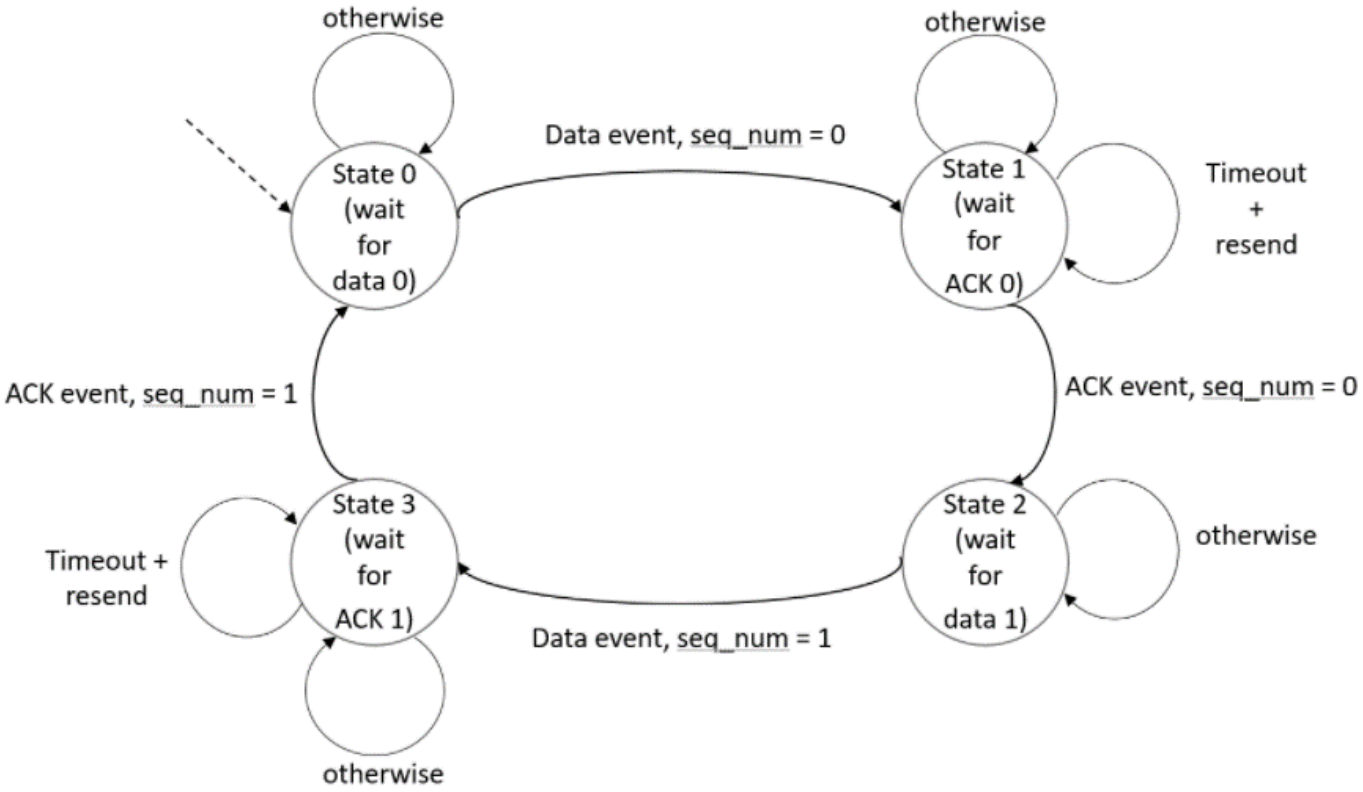
گیرنده بسته را دریافت می کند و مقدار بیت متناوب را بررسی می کند. اگر بیت دریافتی مقدار مورد انتظار باشد، گیرنده یک بسته تایید (ACK) را برای فرستنده ارسال می کند.

اگر بیت دریافتی مقدار مورد انتظار نباشد، گیرنده یک بسته تایید منفی (NAK) برای فرستنده ارسال می کند و از فرستنده درخواست می کند که بسته را مجددا ارسال کند.

با دریافت بسته ACK یا NAK، فرستنده مقدار بیت متناوب را افزایش می دهد و بسته بعدی داده را ارسال می کند.

این فرآیند تا زمانی ادامه می یابد که همه بسته ها با موفقیت ارسال و تایید شوند.

استفاده از مقادیر بیت متناوب در ABP به فرستنده و گیرنده اجازه می دهد تا بسته های گم شده یا خراب را شناسایی و بازیابی کند. اگر بسته ای در حین انتقال گم شود یا خراب شود، گیرنده بسته ACK را ارسال نمی کند و باعث می شود فرستنده بسته را با همان مقدار بیت متناوب دوباره ارسال کند. این ارسال مجدد، تحویل مطمئن داده ها به گیرنده را تضمین می کند.



هنگامی که داده ها از لایه بالا در دسترس هستند، فرستنده یک بسته (با داده به عنوان بار) با شماره دنباله ۰ ارسال می کند. همچنین یک تایمر را شروع می کند.

هنگامی که یک ACK با شماره دنباله ۰ دریافت می کند، تایمر را متوقف می کند و منتظر فراخوانی دیگری از بالا می ماند.

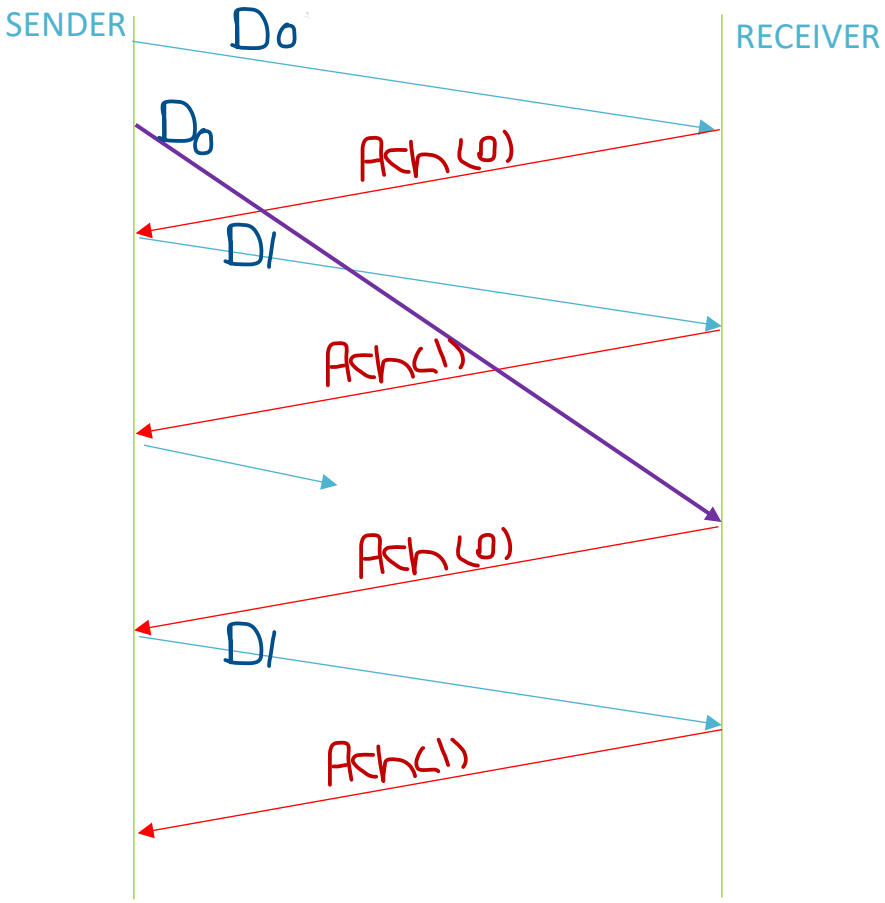
هنگامی که داده ها دوباره در دسترس قرار می گیرند، فرستنده بسته ای با دنباله شماره ۱ می فرستد و دوباره یک تایمر را شروع می کند.

هنگامی که یک ACK با دنباله شماره ۱ دریافت می کند، تایمر را متوقف می کند و منتظر فراخوانی دیگری از بالا می ماند.

هر زمان که مهلت زمانی وجود داشته باشد، فرستنده آخرین بسته خود را دوباره ارسال می کند.

هر زمان که بسته ای را به طور غیرمنتظره دریافت می کند (مثلاً یک ACK اشتباه)، آن را نادیده می گیرد.

۴-



تبادل پیام:

فرستنده داده‌های DATA0 را به گیرنده می‌فرستد و انتظار دارد از گیرنده تأیید شود. بنابراین، فرستنده برای مدت زمان مشخصی منتظر تایید است. از آنجایی که فرستنده تاییدیه را از گیرنده دریافت نمی کند، فرض می کند که داده DATA0 توسط گیرنده دریافت نمی شود. بنابراین، فرستنده دوباره DATA0 را ارسال می کند. پس از ارسال اطلاعات DATA0 برای بار دوم، فرستنده تاییدیه ACK(۰) را از گیرنده دریافت می کند، داده DATA1 را ارسال می کند. گیرنده داده DATA1 را دریافت می کند و ACK(۱) را برای فرستنده ارسال می کند. گیرنده داده های ارسالی مجدد DATA0 را دریافت می کند و با این فرض که تایید قبلی توسط فرستنده دریافت نشده است، مجدداً تأیید ACK(۰) را برای فرستنده ارسال می کند. بنابراین، نسخه قدیمی داده DATA0 توسط گیرنده تایید می شود. فرستنده انتقال داده DATA1 را تکرار می کند و گیرنده تایید ACK(۱) را برای فرستنده ارسال می کند. نسخه جدید داده DATA0 دریافتی توسط گیرنده با نسخه قدیمی داده DATA0 به دلیل مرتب سازی مجدد جایگزین می شود. بنابراین، پروتکل به خوبی کار نمی کند و تبادل پیام نامناسب است. داده ها همانطور که در نظر گرفته شده منتقل نمی شوند.

۵-

پروتکل Stop-and-Wit یک مکانیسم ساده و آسان برای انتقال داده های قابل اعتماد از طریق یک کانال غیر قابل اعتماد است. با این حال، به دلیل این واقعیت که فرستنده باید قبل از ارسال بسته بعدی منتظر تأیید (ACK) از گیرنده باشد، از low channel utilization رنج می برد. این می تواند منجر به تاخیرهای قابل توجهی در انتقال داده شود، به خصوص اگر کانال دارای تاخیر زیاد یا پهنای باند کم باشد.

در موردی که گیرنده یک ACK ارسال می کند حتی اگر فایل را به طور کامل دریافت نکرده باشد، این به طور بالقوه می تواند عملکرد کانال را از نظر کاهش تاخیرها بهبود بخشد. با این حال، این به قیمت کاهش قابلیت اطمینان انتقال است. در یک پروتکل انتقال داده قابل اعتماد مانند RDT 3.0، گیرنده باید قبل از ارسال ACK به فرستنده اطمینان حاصل کند که فایل کامل را دریافت کرده است. این مهم است تا اطمینان حاصل شود که فرستنده قبل از اینکه گیرنده بسته فعلی را پردازش کند، به سمت بسته بعدی حرکت نمی کند، که می تواند منجر به خراب شدن یا از دست رفتن داده شود.

اگر گیرنده قبل از دریافت فایل کامل، یک ACK ارسال کند، می‌تواند منجر به انتقال داده‌های بیشتری از طریق کانال توسط فرستنده شود که به طور بالقوه می‌تواند منجر به خراب شدن یا از دست رفتن داده‌ها شود. بنابراین، ارسال ACK قبل از دریافت فایل کامل در یک پروتکل انتقال داده قابل اعتماد مانند RDT 3.0 توصیه نمی شود.

برای بهبود عملکرد کانال در یک پروتکل توقف و انتظار، می توان از تکنیک های دیگری مانند خط لوله یا تکرار انتخابی استفاده کرد که به فرستنده اجازه می دهد چندین بسته را بدون انتظار ACK از گیرنده ارسال کند. این تکنیک ها می توانند به طور قابل توجهی استفاده از کانال را افزایش داده و تاخیرها را کاهش دهند، در حالی که همچنان قابلیت اطمینان انتقال را حفظ می کنند.

-۶

از آنجایی که کانال A-to-B می تواند پیام های درخواستی را از دست بدهد، A باید مهلت زمانی داشته باشد و پیام های درخواستی خود را مجددا ارسال کند (تا بتواند پس از از دست دادن بازیابی شود). از آنجایی که تاخیرهای کانال متغیر و نامعلوم هستند، ممکن است A درخواست های تکراری ارسال کند (یعنی یک پیام درخواستی را که قبلاً توسط B دریافت شده است دوباره ارسال کند چون تاییدیه نگرفته است). برای شناسایی پیام های درخواست تکراری، پروتکل از اعداد ترتیبی استفاده می کند. یک شماره توالی ۱ بیتی برای نوع stop-and-wait پروتکل request/response کافی است.

A دارای ۴ حالت است:

۱- “Wait for Request 0 from above.”در اینجا درخواست کننده منتظر تماسی از بالا برای درخواست واحد داده است. هنگامی که درخواستی را از بالا دریافت می کند، یک پیام درخواست R0 را به B ارسال می کند، تایمر را شروع می کند و به حالت "Wait for D0" تغییر می کند. هنگامی که این استیت است، A هر چیزی را که از B دریافت می کند نادیده می گیرد.

۲- “Wait for D0” در اینجا درخواست کننده منتظر یک پیام داده D0 از B است. یک تایمر همیشه در این حالت در حال اجرا است. اگر تایمر منقضی شود، A پیام R0 دیگری ارسال می کند، تایمر را مجددا راه اندازی می کند و در این حالت باقی می ماند. اگر یک پیام D0 از B دریافت شود، A زمان را متوقف می کند و به استیت “Wait for Request 1 from above” می رود. اگر A در این حالت یک پیام داده D1 دریافت کند، نادیده گرفته می شود.

۳- “Wait for Request 1 from above” در اینجا درخواست کننده دوباره منتظر تماسی از بالا برای درخواست واحد داده است. هنگامی که درخواستی را از بالا دریافت می کند، یک پیام درخواست R1 را به B ارسال می کند، تایمر را راه اندازی می کند و به حالت "Wait for D1" منتقل می شود. زمانی که A در حالت “”Wait for Request 1 from above است، A هر چیزی را که از B دریافت می کند نادیده می گیرد.

۴- "Wait for D1" در اینجا درخواست کننده منتظر یک پیام داده D1 از B است. یک تایمر همیشه در این حالت در حال اجرا است. اگر تایمر منقضی شود، A پیام R1 دیگری می فرستد، تایمر را مجددا راه اندازی می کند و در این حالت باقی می ماند. اگر یک پیام D1 از B دریافت شود، A تایمر را متوقف می کند و به حالت "Wait for Request 0 from above" منتقل می شود. اگر A در این حالت یک پیام داده D0 دریافت کند، نادیده گرفته می شود.

B (تامین کننده داده) تنها دو حالت دارد:

۱- "Send D0"در این حالت، B با ارسال D0 به پیام های R0 دریافتی پاسخ می دهد و سپس در این حالت باقی می ماند. اگر B یک پیام R1 دریافت کند، می داند که پیام D0 آن به درستی دریافت شده است. بنابراین داده‌های D0 را دور می‌اندازد (از آنجایی که از طرف دیگر دریافت شده است) و سپس به حالت "Send D1" منتقل می‌شود، جایی که از D1 برای ارسال قطعه داده درخواستی بعدی استفاده می‌کند.

۲- "Send D1" در این حالت، B با ارسال D1 به پیام های R1 دریافتی پاسخ می دهد و سپس در این حالت باقی می ماند. اگر B یک پیام R1 دریافت کند، می داند که پیام D1 آن به درستی دریافت شده است و بنابراین به حالت "Send D1" منتقل می شود.