۱) لایهٔ انتقال[۱] در شبکهٔ اینترنت

بگونه‌ای که در فصل قبل اشاره شد پروتکل IP وظیفهٔ هدایت و مسیریابی بسته های اطلاعاتی را از یک ماشین میزبان به ماشینی دیگر برعهده دارد و مشکلاتی که در طی مسیر ممکن است برای یک بستهٔ IP اتفاق بیفتد ، توسط این لایه قابل حل نیست.

وظیفهٔ لایهٔ انتقال در شبکه ، " فراهم آوردن خدمات سازماندهی شده ، مطمئن و مبتنی بر اصول سیستم عامل ، برای برنامه های کاربردی در لایهٔ بالاتر است ، بگونه‌ای که مشکلات و ناکارآمدی لایهٔ IP جبران و ترمیم شود ".

در مقام مقایسه ، وظیفه‌ای را که لایهٔ انتقال بر عهده دارد ، می‌توان با وظایفی که "سیستم مدیریت فایل"[۲] به عنوان بخشی از سیستم عامل بر عهده دارد ، قیاس کرد. سیستم مدیریت فایل از یک طرف با ابزارهای ذخیره سازی اطلاعات که ذاتاً سخت افزاری ، متنوع و ناهمگون هستند ، سر و کار دارد و از طرف دیگر با برنامه های کاربردی در ارتباط است که برای ذخیره و بازیابی اطلاعات فقط مفهومی به نام فایل ، در اختیار دارد و از دید برنامه نویس نوع ابزار و چگونگی و محل فیزیکی ذخیرهٔ داده هایش مهم نیست ، بلکه فقط عملیات لازم را برنامه ریزی میکند. از دیدگاهِ ابزارهای ذخیره و بازیابی اطلاعات ، چیزی به نام فایل ، درایوهای منطقی (مجازی) و جدول FAT[۳] معنایی ندارد ، بلکه این ابزار میتوانند یک بلوک داده را با اندازهٔ ثابت ، تحویل گرفته و بر روی محل مشخصی از فضای فیزیکی ذخیره سازی اطلاعات بنویسند (یا بخوانند) . سیستم مدیریت فایل که بین این ابزار فیزیکی و برنامه های کاربردی قرار میگیرد از یک ابزار فیزیکیِ خام ، یکپارچه و پیچیده ، خدماتی را در قالب مفهوم فایل به برنامه های کاربردی ارائه میکند که کاملاً قابل اعتماد ، شفاف ، ساده و عاری از هرگونه پیچیدگی سخت افزاری است. سیستم مدیریت فایل برای ارائهٔ چنین خدماتی باید جداول FAT ، جدول درایوهای منطقی[۴] ، سیستم فهرست فایلها[۵] و ... را ایجاد و سازماندهی نماید. تنها کاری که برنامه نویس برای بهره گیری از خدمات سیستم فایل باید انجام بدهد آنست که فایلی را بگشاید و تقاضای خواندن از آن یا نوشتن در آن را بدهد. پیچیدگی هایی که در این بین وجود دارد توسط مدیر فایل حلّ و فصل می‌شود.

وظیفهٔ لایه انتقال همین مفهوم را دنبال میکند یعنی : " بهره گیری از خدمات لایهٔ IP که سریع و ساده و در عین حال غیرمطمئن و ناکارآمد است و ارائهٔ خدماتی مطمئن ، ساختاریافته

---

۱ Transport Layer
۲ File Management System
۳ File Allocation Table
۴ Partition Table
۵ Root Directory

و شفاف به برنامه های کاربردی در لایهٔ بالاتر ، به گونه‌ای کـه برنامـه نویـس از درگـیری بـا جزییات زیرشبکه و مشکلات کانالهای انتقال و مسایلی از این قبیل به دور باشد".

برای تشریح وظایف لایهٔ انتقال باید کاستی های لایهٔ IP را بررسی کرده و سپس روشـی را که لایهٔ انتقال برای جبران آنها برگزیده است ، توضیح بدهیم. دقت کنید که منشأ کاستی هـای لایهٔ IP ، ذاتِ کانالهای انتقال و مشکلات فیزیکی در زیرشبکهٔ ارتباطی است. عمدهٔ این کاستی ها عبارتند از:

- تضمینی وجود ندارد وقتی بسته‌ای برای یک ماشین مقصد ارسـال میشـود آن ماشـین آمـادهٔ دریافت آن بسته باشد و بتواند آنرا دریافت کند.
- تضمینی وجود ندارد وقتی چند بستهٔ متوالی برای یک ماشین ارسال می‌شود به همان ترتیبـی که بر روی شبکه ارسال شده‌اند ، در مقصد دریافت شوند.
- تضمینی وجود ندارد که وقتی بسته‌ای برای یک مقصد ارسال می‌شود ، به دلیل دیـر رسـیدن مجدداً ارسال نشود و در چنین حالتی ممکن است بسته‌ای به اشتباه دو بار[1] در مقصد دریـافت شود. لایهٔ IP قادر نیست تمایزی بین دو بستهٔ عین هم ، که یکی از آنها زائد است قائل شـود و هر دو را تحویل ماشین مقصد می‌دهد.
- لایهٔ IP هیچ وظیفه‌ای در قبال توزیع بسته ها بین پروسه های مختلفی که بر روی یک ماشین واحد اجرا شده‌اند ندارد. در یک محیط "چند کاربره"[2] یا "چند وظیفه‌ای"[3] ممکن است چندین پروسهٔ متفاوت تقاضای ارسال یا دریافت داده داشته باشند. حال فرض کنید بسته‌ای به لایـهٔ IP از یک ماشین واحد ، تحویل داده شود. داده های درون این بسته متعلق به کدامین پروسـهٔ در حال اجرا روی آن ماشین است؟ از دیدگاه لایهٔ IP مفهومی به نام "پروسه های متفاوتِ در حال اجرا" ، رسمیّت و هویّت ندارد.
- لایهٔ IP هیچ وظیفه‌ای در قبال تنظیم سرعت تحویل بسته ها به یک ماشین ندارد. مثلاً ممکـن است یک ماشین با سرعت بسیار زیاد بسته هایی را تولید کرده و تحویـل لایـهٔ IP بدهـد ولـی ماشین مقصد قادر نباشد بسته ها را با این سرعت دریافت کند و بسته هـا در مقصـد بـه دلیـل عدم توانایی در دریافت ، از بین بروند.

در لایهٔ انتقال دو پروتکل به نامهای TCP[4] و UDP[1] تعریف شده‌اند که ابتدا پروتکـل TCP را که تمام کاستی های عنوان شده را جبران کرده معرفـی می‌کنیـم و نـهایتاً بـه پروتکـل UDP و مشخصات آن خواهیم پرداخت.

---

1 Duplication Problem
2 Multi User
3 Muti Task
4 Transmission Control Protocol

۲) راهکارهای پروتکل TCP برای جبران کاستی های لایهٔ IP

در این بخش مفهوم عملیاتی که پروتکل TCP برای جبران کاستیهای لایهٔ IP انجام میدهد ، بررسی می‌شود و سپس جزئیات این عملیات را در بخشهای آتی ارائه می‌دهیم.

اولین کاستی در لایهٔ IP ، عدم تضمین در آماده بودن و توانایی دریافت داده ها توسط ماشین مقصد ، عنوان شد. در پروتکل TCP راهکاری ساده وکارآمد برای این مشکل اتخاذ شده است: " برقراری یک ارتباط و اقدام به هماهنگی بین مبدأ و مقصد ، قبل از ارسال هرگونه داده".

برای تشریح این راه حل ، فرض کنید پروسهٔ A تمایل داشته باشد برای پروسهٔ B بر روی یک ماشین مشخص ، داده هایی را ارسال کند؛ قبل از اقدام به ارسال داده به صورت زیر عمل می‌کند:

الف) A یک بستهٔ خاص را به عنوان درخواست برای ارتباط ، به آدرس ماشین B می‌فرستد و منتظر می‌ماند.

ب) B درخواست ارتباط را دریافت کرده و بر حسب شرایط ، آمادگی یا عدم آمادگی خود را به A اعلام مینماید. (ممکن است B اصلاً وجود خارجی نداشته باشد و طبعاً هیچ پاسخی بر نمیگردد.)

ج) در صورتی که A در یک مهلت زمان مشخص ، پاسخ مثبت مبنی بر آماده بودن B دریافت نماید میتواند به ارسال داده ها اقدام نماید.

به پروتکلهایی که قبل از مبادلهٔ داده ها سعی در برقراری یک ارتباط و ایجاد هماهنگی قبلی می‌نمایند پروتکلهای "اتصال گرا"[۲] گفته میشود. در این پروتکلها خاتمهٔ مبادلهٔ داده ها نیز بایستی در یک روند هماهنگ و با اطلاع قبلی انجام شود:

الف) A خاتمهٔ ارسال داده های خود را اعلام می‌کند ولی باید منتظر بماند و به دریافت داده های ارسالی از طرف B ادامه بدهد تا آ نکه B نیز اعلام ختم ارتباط را تایید کند.

ب) B نیز اعلام ختم ارتباط کرده و ارتباط ، در یک روند هماهنگ خاتمه می‌یابد.

البته ممکن است اتفاقاتی در خلال مبادلهٔ داده ها رخ بدهد که توسط لایهٔ TCP قابل حل نباشد (مثل خرابی خط یا خاتمه ناهنگام یکی از برنامه های A یا B توسط سیستم عامل در اثر

---

[۱] User Datagram Protocol
[۲] Connectuion Oriented

بروز یک مشکل داخلی[۱]، یا هر اتفاق دیگر) این مشکلات در پروتکل TCP توسط زمان سنج های خاصّی کشف و مدیریت می‌شوند که در بخشی مجزا به آنها خواهیم پرداخت.

معضلات بعدی در لایهٔ IP تضمین به ترتیب رسیدن داده ها و صحّت آنهاست. حلّ این مسایل چندان مشکل نیست. مجدداً فرض کنید پروسهٔ A تمایل داشته باشد برای پروسهٔ B بر روی یک ماشین مشخص ، داده هایی را ارسال کند و قبل از اقدام به ارسال داده ها یک ارتباط موفق برقرار کرده باشد. برای تضمین صحّت و ترتیب داده ها روند زیر قابل انجام است:

الف) A بخشی از داده هایی که باید ارسال شوند را در قالب یک بسته سازماندهی کرده و در سرآیند آن یک "شمارهٔ ترتیب"[۲] تنظیم می‌نماید؛ سپس ضمن نگهداری آن بسته درون یک بافر ، آن را جهت هدایت به سمت مقصد ، تحویل لایهٔ IP می‌دهد و یک "زمان سنج" تنظیم مینماید. همچنین برای نظارت بر خطاهای احتمالی یک کد ۱۶ بیتی کشف خطا در سرآیند بسته قرار می‌دهد.

ب) در صورتی که B بستهٔ ارسالی از A را سالم دریافت کرد ، یک "پیغام تصدیق" که اختصاراً Ack نامیده میشود برای A پس می‌فرستد.[۳]

ج) اگر A در زمان مقرّر پیغام Ack را دریافت کرد ، بافر مربوط به آن بسته را آزاد کرده و اقدام به ادامهٔ ارسال داده ها به همین روال مینماید. اگر به دلیل خرابی داده ها (یا خرابی پیغام Ack در مسیر برگشت) در مهلت مقرر پیغام تصدیق دریافت نشود ، بسته بافرشده از نو ارسال میشود.[۴]

با قرار دادن شمارهٔ ترتیب برای داده ها می‌توان تضمین کرد که جریان داده ها به ترتیب می‌رسند و به هر دلیلی اگر بسته‌ای دو بار دریافت شوند ، با مقایسهٔ شماره های ترتیب ، یکی از آنها دور انداخته می‌شود.

با تنظیم یک کد ۱۶ بیتی کشف خطا در مبدأ و بررسی مجدّد آن در مقصد ، می‌توان از صحت داده ها نیز مطمئن شد. جزئیات این عملیات با تشریح پروتکل TCP مشخص خواهد شد.

---

[۱] Bug

[۲] Sequence Number

[۳] ارسال Ack معمولا بصورت مجزا ارسال نمیشود بلکه ضمیمهٔ اطلاعاتی میشود که قرار است در پاسخ ، ارسال شود ، مگر آنکه داده ای برای ارسال وجود نداشته باشد؛ به این روش Piggy Backing گفته میشود.

[۴] به پروتکلهایی که فقط در هنگام دریافت صحیح داده ها پیغام Ack برمیگردانند و در صورت دریافت بستهٔ خراب ساکت میمانند ، پروتکلهای Positive Acknowledgement with Retransmission- PAR- گفته میشود.

در پروتکل TCP برای به رسمیّت شناختن پروسه های مختلفی که بر روی یک ماشین در حال اجرا هستند راه حل زیر ارائه شده است:

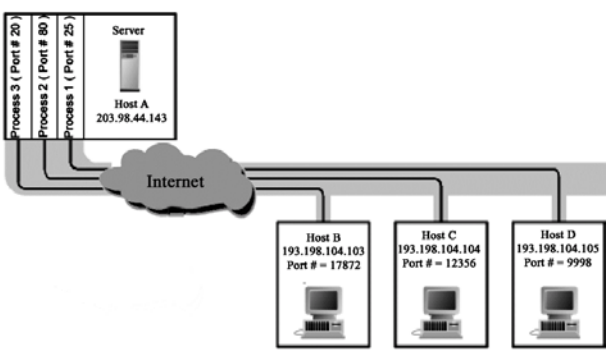
هر پروسه برای تقاضای برقراری یک ارتباط با پروسه‌ای دیگر روی شبکه ، یک شمارهٔ شناسایی برای خود برمی‌گزیند. به این شمارهٔ شناسایی "آدرس پورت"[1] گفته می‌شود. در سرآیند بسته‌ای که توسط پروتکل TCP سازماندهی می‌شود آدرس پورت پروسهٔ فرستنده و آدرس پورت پروسهٔ گیرندهٔ آن درج می‌شود. یکتا بودن شماره های پورت که به پروسه ها رسمیت و هویت می‌بخشد ، توسط پروتکل TCP به عنوان جزئی از سیستم عامل نظارت خواهد شد. سیستم عامل جدولی را نگهداری میکند که شماره شناسایی تقاضا دهندهٔ ارتباط در آن وجود دارد.

به شکل (۱-۵) دقت کنید. آدرس IP ، یک ماشین یکتا را در کل شبکه مشخص می‌نماید؛ شمارهٔ پورت نیز از بین پروسه های اجرا شده بر روی آن ماشین ، یکی از آنها را به عنوان مبدأ (یا مقصد) تعیین می‌کند. بنابراین زوجِ آدرس IP و آدرس پورت می‌تواند یک پروسهٔ یکتا و واحد را بر روی هر ماشین در دنیا مشخص نماید. در ادبیات شبکه به این زوج آدرس ، "آدرس سوکت" گفته می‌شود:

(IP Address **:** Port Number) = Socket Address

مثال : 193.142.22.121**:**80

(البته اصطلاح "آدرس سوکت" نباید با مفهوم "برنامه نویسی سوکت" اشتباه شود ، آنرا در فصلی جداگانه به تفصیل بررسی خواهیم کرد.)



شکل (۱-۵) آدرس دهی پروسه ها بوسیلهٔ شماره پورت روی یک ماشین واحد

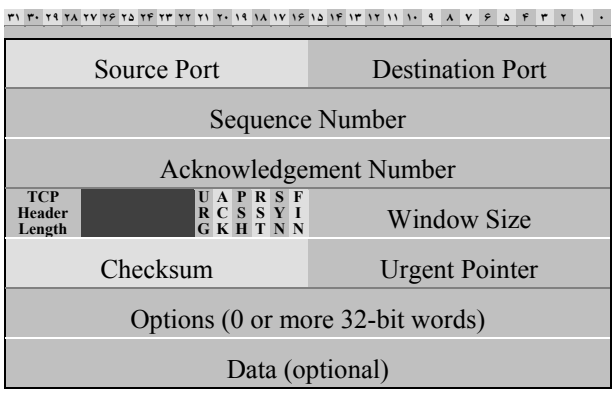
[1] Port Number

برای حل مسئلهٔ هماهنگی سرعت ارسال و دریافت در پروتکل TCP الگوریتمی پویا برای تنظیم مجموعهٔ زمان سنجهایی که در این رابطه انجام وظیفه می‌نمایند بکار گرفته شده است که در بخشی مجزا تشریح خواهد شد.

قبل از وارد شدن به جزئیات پروتکل TCP بهتر است ساختار بسته ای را که این پروتکل برای تحویل به لایهٔ IP تنظیم و سازماندهی میکند ، مورد بررسی قرار بدهیم چرا که بسیاری از مسائل با بررسی ساختار این بسته آشکار خواهد شد. (بسته‌ای که در لایهٔ انتقال تولید و تنظیم می‌شود ، "قطعهٔ TCP"[۱] یا TPDU[۲] نام دارد ، که به اختصار به آن بستهٔ TCP خواهیم گفت.)

۳) ساختار بسته های پروتکل TCP

در این بخش یک دید کلی از پروتکل TCP ارائه می‌نمائیم و ساختار سرآیند بسته ها را در این پروتکل ، توضیح خواهیم داد. در شکل (۲-۵) ساختار یک بستهٔ TCP به تصویر کشیده شده است.



شکل (۲-۵) ساختار یک بستهٔ TCP

[۱] TCP Segment
[۲] Transport Protocol Data Unit

♦ **فیلد Source Port** : در این فیلد یک شمارهٔ ۱۶ بیتی بعنوان آدرس پورت پروسهٔ مبدأ که این بسته را جهت ارسال ، تولید کرده ، قرار خواهد گرفت.

♦ **فیلد Destination Port** :در این فیلد ، آدرس پورتِ پروسهٔ مقصد که آنرا تحویل خواهد گرفت ، تعیین خواهد شد .

همانگونه که در بخش قبلی اشاره شد این دو آدرس مشخص می‌کنند که این بستهٔ از چه برنامهٔ کاربردی در لایه بالاتر تولید و باید به چه برنامه‌ای در ماشین مقصد تحویل داده شود. برخی از پروسه های کاربردی و استاندارد دارای شماره پورت استاندارد و جهانی هستند؛ مثلاً سرویس دهنده پست الکترونیکی دارای شماره پورت ۲۵ است. به جدول شماره پورتهای استاندارد در انتهای این فصل نگاهی بیندازید.

♦ **فیلد Sequence Number** : این فیلد سی و دو بیتی ، شمارهٔ ترتیب آخرین بایتی را که در "فیلد داده" از بستهٔ جاری قرار دارد ، نشان می‌دهد.

در پروتکل TCP شمارهٔ ترتیب ، بر حسب شمارهٔ آخرین بایتی است که در بستهٔ جاری قرار گرفته و ارسال شده است. بعنوان مثال اگر در این فیلد عددی معادل ۱۹۳٤۱ قرار بگیرد به این معناست که داده ها تا بایت شمارهٔ ۱۹۳٤۱ درون فیلد داده قرار دارد . دقت کنید که این عدد بمعنای آن نیست که به تعداد ۱۹۳٤۱ بایت ، درون قسمت داده قرار دارد ، بلکه همیشه به شمارهٔ ترتیب آخرینِ بایت داده ، اشاره مینماید. یعنی ممکن است که کلاً درون فیلد داده فقط یک بایت قرار داشته باشد در حالی که در فیلد شمارهٔ ترتیب عدد ۱۹۳٤۱ قرار داشته باشد. دقّت شود که شمارهٔ ترتیبِ اولین بایت ، از صفر شروع نمی‌شود بلکه از یک عدد تصادفی که در هنگام برقراریِ ارتباط به اطلاع طرفین میرسد ، شروع خواهد شد.

♦ **فیلدAcknawledgement Number** : این فیلد ۳۲ بیتی نیز شمارهٔ ترتیب بایتی که فرستندهٔ بسته منتظرِ دریافت آن است را تعیین می‌کند. بعنوان مثال اگر دراین فیلد عددی معادل ۳٤۲۳۱۰ قرار گرفته باشد بدین معناست که از رشته داده ها (که مشخص نیست چند بایت است) تا شماره ۳٤۲۳۱۰ صحیح و کامل دریافت شده است و منتظر بایتهای از ۳٤۲۳۱۱ به بعد می‌باشد .

♦ **فیلد TCP Header Length** : عددی که در این فیلد قرار می‌گیرد ، طول سرآیند بستهٔ TCP را بر مبنای کلمات ۳۲ بیتی تعیین می‌کند . بعنوان مثال اگر در این فیلد عدد ۷ قرار بگیرد طول سرآیند مقدار ۲۸=۷×٤ بایت خواهد بود. (این فیلد کلاً چهار بیتی است) دقّت کنید که قسمت ثابت و اجباری در یک بستهٔ TCP حداقل ۲۰ بایت است ولی در فیلد اختیاری Options می‌تواند اطلاعاتی قرار و بنابراین گیرندهٔ یک

بستهٔ TCP باید بتواند مرز بین سرآیندِ بسته و قسمت داده را تشخیص بدهد . پس عددی که در این فیلد قرار می‌گیرد می‌تواند بعنوان یک "اشاره گر"[1] ، محل شروع داده ها را در یک بستهٔ TCP تعیین کند (توجه دارید که مبنای این عدد کلمات ۳۲ بیتی (چهار بایتی) هستند)

- **٦ بیت بلا استفاده** : پس از فیلد TCP Header Length شش بیت بلا استفاده رها شده است که شاید برای استفاده در آینده رزرو شده‌اند .
- **بیتهای Flag** : شش بیت بعدی در بستهٔ TCP هر کدام نقش یک بیت پرچم را که معنا و کاربرد مختلفی دارند را بازی میکنند.

URG	ACK	PSH	RST	SYN	FIN

تک تک این بیتها و معنای آنها را به ترتیب بررسی می‌کنیم:

- **بیت URG** : در صورتی که این بیت مقدار ۱ داشته باشد ، معین می‌کند که در فیلد Urgent Pointer که در ادامه معرفی خواهد شد مقداری قابل استناد و معتبر قرار دارد و بایستی مورد پردازش قرار گیرد . درصورتی که این بیت صفر باشد فیلد Urgent Pointer شامل مقدار معتبر و قابل استنادی نیست و از آن چشمپوشی می‌شود.
- **بیت ACK** : اگر در این بیت مقدار ۱ قرار گرفته باشد ، نشان می‌دهد که عددی که در فیلد Acknowledgement Number قرار گرفته است ، دارای مقداری معتبر و قابل استناد است . بیتACK و بیت SYN نقش دیگری نیز دارند که در ادامه بدان اشاره خواهد شد.
- **بیت PSH**[2] : اگر در این بیت مقدار ۱ قرار گرفته باشد فرستندهٔ اطلاعات از گیرنده تقاضا می‌کند که داده های موجود در این بسته را بافر نکند و در اسرع وقت آنرا جهت پردازشهای بعدی تحویل برنامهٔ کاربردی صاحب آن بدهد. این عمل گاهی برای برنامه هائی مشابه Telnet ضروری است؛ بعنوان مثال فرض کنید یک کاربر با کامپیوتر شخصی خود از نوع سازگار با IBM به کامپیوتری در فاصله هزاران کیلومتری خود وصل شده و تصمیم دارد از طریق یک محیط شبیه سازی شده با کامپیوتر سرویس دهنده ارتباط برقرار کرده و دستورات سیستم عامل UNIX را تمرین نماید. فرض کنید کاربر دستور ls (معادل dir در DOS) را اجرا می‌کند و بالطبع توقع دارد پس از ارسال این دو کاراکتر و تحویل آن به برنامه Telnet سریعاً پاسخ لازم را روی صفحهٔ نمایشش ببیند ولی نرم افزار TCP

---

[1] Pointer
[2] Push

معمولاً در هنگام دریافت بسته های کوچک ، آنها را بافر کرده تا وقتی حجم بافر به اندازهٔ مشخصی پر شد ، آنرا یکجا تحویل برنامه کاربردی بدهد. در چنین حالتی فرستندهٔ بسته با ۱ کردن بیت PSH ، از گیرندهٔ آن می‌خواهد که آنرا بافر نکند.

- **بیت RST** : اگر در این بیت مقدار ۱ قرار بگیرد ارتباط بصورت یکطرفه و ناتمام قطع خواهد شد[۱]، بدین معنا که به هر دلیلی (اعم از نقص سخت افزاری یا نرم افزاری) اشکالی بوجود آمده که یکی از طرفین ارتباط مجبور به خاتمه ارتباط فعلی شده است. همچنین بیت RST می‌تواند بعنوان علامت عدم پذیرش برقراری ارتباط بکار برود.اگر یکی از طرفین ارتباط یک بسته دریافت کند که در آن بیت RST مقدار۱ داشته باشد ، ارتباط بصورت ناهماهنگ و نامتعادل ، قطع خواهد شد.
- **بیت SYN** : این بیت نقش اساسی در برقراری یک ارتباط بازی می‌کند . برقراریِ یک ارتباطِ TCP از روند زیر تبعیت میکند:

الف) شروع کننده ارتباط یک بستهٔ TCP بدون هیچگونه داده و با تنظیم بیتهای (ACK=0, SYN=1) ، برای طرف مقابل ارسال می‌کند . در حقیقت ارسال چنین بسته ای به معنای "تقاضای برقراری ارتباط"[۲] تلقی می‌شود.

ب) در پاسخ به درخواست ارتباط ، در صورتیکه طرف مقابل به برقراری ارتباط تمایل داشته باشد بسته ای بر می‌گرداند که در آن بیت SYN=1 و بیت ACK=1 است . این بسته نقش "پذیرش یک ارتباط"[۳] را بازی می‌کند.

برقراری ارتباط را بیشتر توضیح خواهیم داد.

- **بیت FIN** : اگر یکی از طرفین ارتباط ، داده دیگری برای ارسال نداشته باشد در هنگام ارسال آخرین بستهٔ خود این بیت را ۱ می‌کند و در حقیقت ارسال اطلاعات خودش را یکطرفه قطع می‌کند. در این حالت اگر چه ارسال اطلاعات قطع شده و لیکن طرف مقابل هنوز ممکن است به ارسال اطلاعات مشغول باشد. زمانی ارتباط کاملاً خاتمه می‌یابد که طرف مقابل نیز در یک بسته با ۱ کردن بیت FIN ، ارسال اطلاعات را خاتمه بدهد.

♦ **فیلد Windows Size** : مقدار قرار گرفته در این فیلد مشخص می‌کند که فضای بافرِ گیرنده چند بایت دیگر ظرفیت خالی دارد. یعنی به طرف مقابل اعلام می‌کند که مجاز است از بایت با شمارهٔ ترتیبی که در فیلد Acknowledgement مشخص شده

---

[۱] Abnormaly Ended
[۲] Connection Request
[۳] Connection Accept

است، حداکثر به اندازهٔ مقداری که در این فیلد درج شده ، ارسال داشته باشد و در غیر اینصورت فضای کافی برای دریافت داده ها وجود نداشته و ناگزیر دور ریخته خواهد شد. اگر مقدار این فیلد صفر باشد بدین معناست که بافر گیرنده تماماً پر شده است و امکان دریافت داده های بعدی وجود ندارد و پروسهٔ فرستنده متوقف خواهد شد؛ در این مورد نیز بیشتر توضیح خواهیم داد.

♦ **فیلد Checksum** : در این فیلد ۱٦ بیتی ، کد کشف خطا قرار می‌گیرد. طریقه محاسبه این کد اندکی متفاوت از روشهای معمول است:

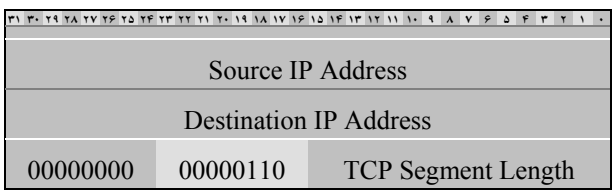
الف) کلّ بستهٔ TCP شامل قسمت سرآیند بسته و قسمت داده ، در قالب کلمات ۱٦ بیتی در نظر گرفته می‌شود. (منهای قسمت Checkum)

ب) یک "سرآیند فرضی"[۱] که در بستهٔ TCP وجود ندارد ، با قالب زیر ساخته شده و بصورت کلمات ۱٦ بیتی در نظر گرفته می‌شود.

ج) تمامی کلمات در "مبنای مکمل ۱"[۲] با هم جمع شده و سپس عدد بدست آمده در مبنای مکمل ۱ منفی می‌شود. این عدد نهایتاً در فیلد Checksum قرار می‌گیرد.

اگر در حین ارسال داده ها خطائی بروز نکند ، پس از دریافت بسته در مقصد ، جمع کل کلمات ۱٦ بیتی موجود در یک بستهٔ TCP به انضمام "سرآیند فرضی" بایستی صفر شود ، در غیر اینصورت داده های غیرمعتبر و خراب هستند.

ساختار "سرآیند فرضی" که بصورت ساختگی تولید می‌شود بصورت زیر است:



همانگونه که دیده می‌شود این سرآیند ساختگی شامل فیلدهای زیر است:

- ۳۲ بیت آدرس IP مربوط به ماشین مبدأ
- ۳۲ بیت آدرس IP مربوط به ماشین مقصد
- یک فیلد هشت بیتی کاملاً صفر
- فیلد هشت بیتی پروتکل که برای پروتکل TCP یقیناً مقدار ٦ دارد.

---

[۱] Pseudo Header
[۲] 1's Compelement

- فیلد TCP Segment Length که در آن طول کل بستهٔ TCP مشخص می‌شود.

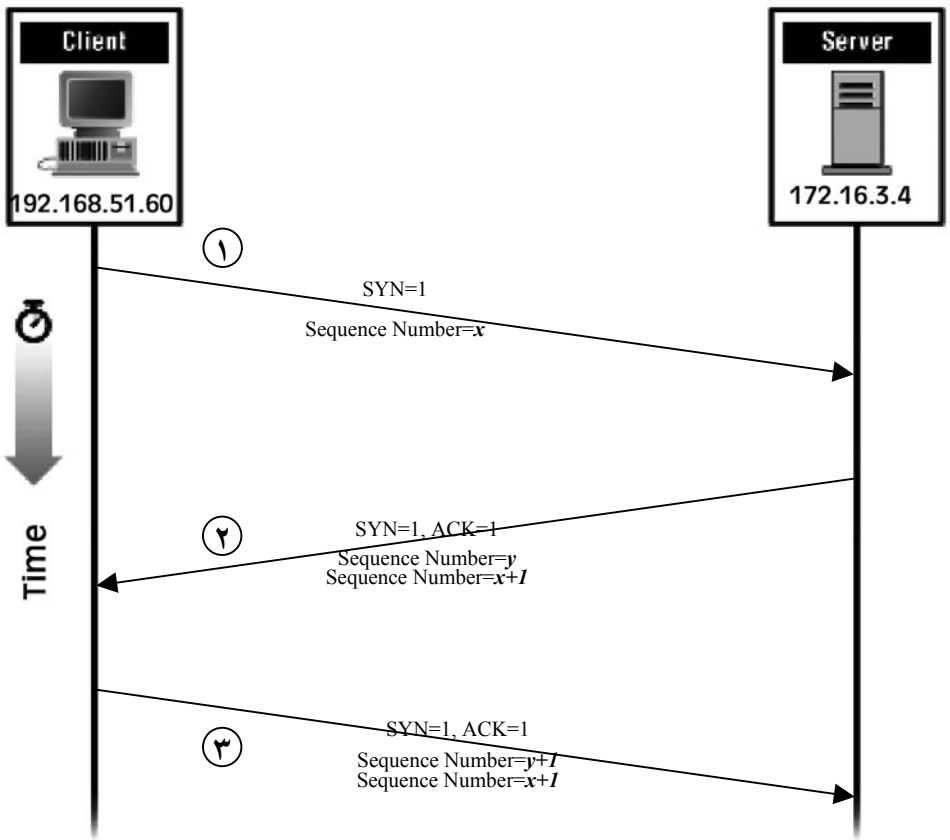
♦ **فیلد Urgent Pointer** : در این فیلد یک عدد بعنوان اشاره گر قرار می‌گیرد که موقعیت داده های اضطراری را درون بستهٔ TCP معین می‌کند . این داده ها ، زمانی اتفاق می‌افتند و ارسال می‌شوند که عملی شبیه وقوع وقفه ها در هنگام اجرای یک برنامهٔ کاربردی رخ بدهد. بدون آنکه ارتباط قطع شود داده های لازم در همین بستهٔ جاری ارسال خواهد شد. دقّت کنید که داده های اضطراری توسط برنامهٔ کاربردی در لایهٔ بالاتر پردازش خواهد شد و برای پروتکل TCP کاربردی ندارد.

♦ **فیلد Options** : در این فیلد اختیاری است و مقداری نظیر حداکثر طول بستهٔ TCP در آن قرار میگیرد. برای آنکه طول بسته ضریبی از ٤ باقی بماند از این فیلد با کدهای بی ارزش استفاده می‌شود. گزینهٔ خاص دیگری در این فیلد تعریف نشده است.

۴) روش برقراری ارتباط در پروتکل TCP

برای برقراری ارتباط در پروتکل TCP از روش "دست تکانی سه مرحله ای" استفاده می‌شود . البته برقراری ارتباط منوط به این قضیه است که طرفین ارتباط آمادهٔ برقراری یک ارتباط باشند یعنی یکطرف که فعلاً آنرا سرویس دهنده می‌نامیم برای برقراری ارتباط از طریق توابع سیستمی ()listen و ()accept اعلام آمادگی کرده باشد و طرف مقابل نیز یعنی مشتری با فراخوانی تابع سیستمی ()connect و تعیین آدرس IP و آدرس پورت پروسهٔ مقصد ، تمایل خود را برای ارتباط ، ابراز نماید. این توابع در فصل برنامه نویسی تحت شبکه بطور مبسوط توضیح داده خواهد شد. در چنین حالتی بین طرفین اتفاقات ۳ مرحله ای زیر خواهد افتاد (در حالت طبیعی).
در شکل (۳-۵) این مراحل به تصویر کشیده شده است.

در مرحله اوّل ، از طرف شروع کنندهٔ ارتباط ، یک بستهٔ TCP (خالی از داده) ارسال خواهد شد که در آن بیت SYN=1 و بیت ACK=0 است و درون فیلد شماره ترتیب عدد x قرار داده شده که در آن x یک عدد تصادفی است. در حقیقت با این شماره به طرف مقابل اطلاع داده می‌شود که ترتیب داده های ارسالی از شماره $x+1$ شروع می‌شود . در پروتکل TCP شماره ترتیب ۳۲ بیتی است لذا برای پیشگیری از مشکلات احتمالی ناشی از مساوی بودن شمارهٔ ترتیب بسته های ارسالی ، داده ها از شماره ۰ شروع نمی‌شوند ، بلکه از یک عدد تصادفی (که بصورت خودکار تولید می‌شود) ، شروع می‌گردد و در همان مرحلهٔ اول ، این



شکل (۳-۵) مراحل دست تکانی سه مرحله ای برای برقراری ارتباط در پروتکل TCP

شمارهٔ ترتیب به طرف مقابل اعلام خواهد شد. بعنوان مثال اگر SEQ=145500 باشد بدین معناست که داده هائی که قرار است ارسال شوند شماره ترتیب آنها از ۱٤٥٥۰۱ آغاز خواهد شد. طرف مقابل حتماً باید از این موضوع باخبر باشد.

در مرحلهٔدوم ، طرف مقابل با دریافت بسته ای با مشخصات فوق الذکر اگر تمایل به برقراری ارتباط نداشته باشد با ارسال یک بستهٔ خالی که در آن بیت RST به ۱ تنظیم شده ، این تقاضا را ردّ می‌کند ولی اگر متمایل به برقراری ارتباط بود یک بستهٔ خالی از داده با مشخصات زیر تولید می‌کند:

- بیت SYN را یک می‌کند.
- بیت ACK را یک می‌کند.

- مقدار فیلد Acknowledgement Number را $x+1$ قرار می‌دهد.
- مقدار فیلد Sequence Number را مقدار تصادفی y قرار می‌دهد.

در این مرحله که به معنای پذیرش ارتباط است طرف مقابل با قرار دادن مقدار فیلد Ack=$x+1$ نشان می‌دهد که شماره ترتیب x را پذیرفته و منتظرداده ها از شماره ترتیب $x+1$ به بعد است . در ضمن خودش عدد تصادفی y را در فیلد .Seq.No قرار می‌دهد و به طرف مقابل اعلام می‌کند که شماره ترتیب داده های ارسالی از y خواهد بود.

در مرحلهٔ سوّم ، شروع کننده ارتباط با قرار دادن مقادیر زیر شروع ارتباط را تصدیق می‌کند :

- بیت SYN را یک می‌کند.
- بیت ACK را یک می‌کند.
- فیلد Seq. No.=x+1 را قرار می‌دهد.
- فیلد Ack را y+1 قرار می‌دهد.

با قرار دادن Seq.No.=$x+1$ و Ack=$y+1$ شروع کنندهٔ ارتباط اعلام می‌کند که بر روی پارامترهای شماره ترتیب توافق شده است و او پذیرفته که داده های طرف مقابل را از شماره $y+1$ بپذیرد. پس از این مرحله ارسال و دریافت داده ها توسط طرفین تا هنگامی که ارتباط با اطلاع طرفین خاتمه داده نشده است آزاد است.

برای خاتمه ارتباط روند زیر صورت می‌گیرد:

طرفی که داده هایش برای ارسال تمام شده است یک بستهٔ TCP ارسال می‌نماید که در سرآیند آن بیت FIN را یک قرار داده است. طرف مقابل این درخواست را دریافت می‌کند و با ختم یک طرفهٔ آن موافقت می‌کند. ولی چون ارتباط بصورت یکطرفه ختم می‌شود طرف مقابل می‌تواند تا جائیکه داده دارد ، آنها را ارسال کند و نهایتاً در آخرین بسته ، بیت FIN را یک بگذارد تا پس از تصدیقِ آن ، ارتباط به صورت دو طرفه ختم شود.

نکته ای که وجود دارد آنست که اگر یکی از طرفین ارتباط در اثر بروز مشکلی سخت افزاری یا نرم افزاری ارتباط را بدون هماهنگی قطع کند حق ندارد تا ۱۲۰ ثانیه به ارتباط مجدد با همان پروسه اقدام کند و این نکته ناشی از آن است که مطمئن

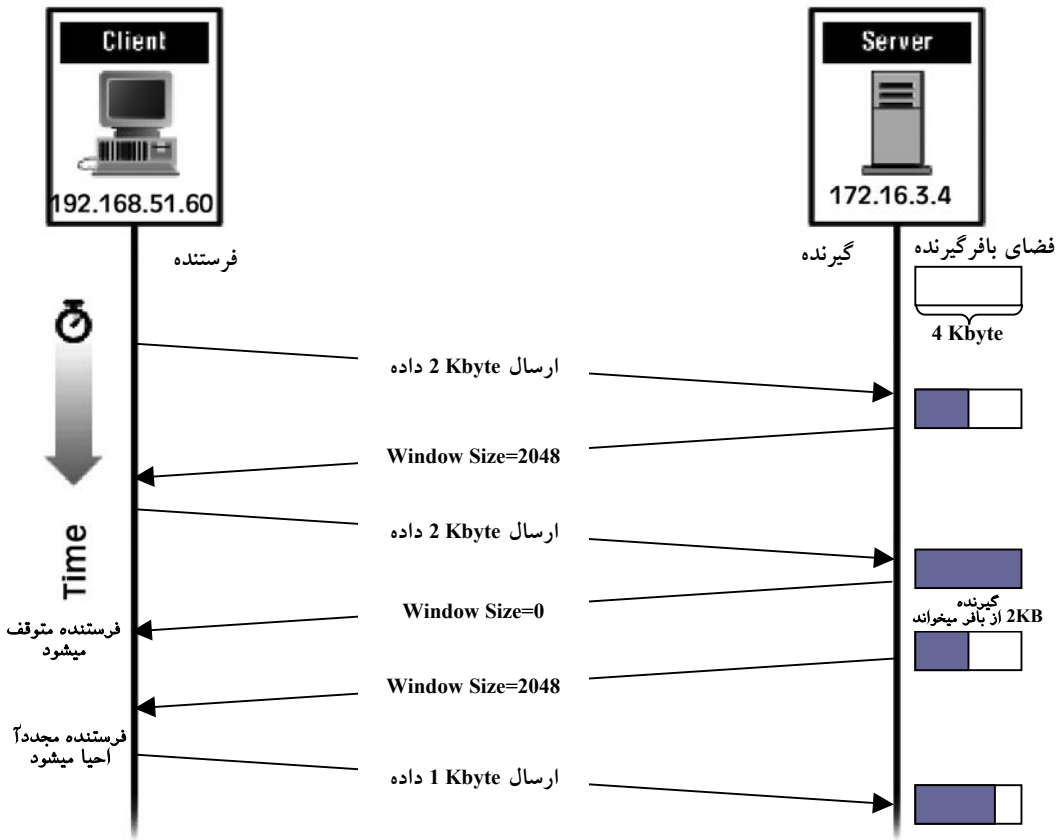
باشد بسته هـای قبلـی کـه ارسـال کـرده یـا آنکـه برایـش ارسـال شـده از زیرشـبکه حـذف شـده‌اند .

۵) کنترل جریان در پروتکل TCP

در اینجـا بـد نیسـت کـه اندکـی در مـورد نقـش فیلـد Window size بحـث کنیـم . همانگونـه کـه قبـلاً اشـاره شـد در پروتکـل TCP بـرای کنـترل جریـان داده هـا از بـافر اسـتفاده می‌شـود و داده هـا قبـل از ارسـال بـه برنامـه کـاربردی لایـه بـالاتر بـافر شـده و بصـورت دسـته ای تحویـل خواهـد شـد و گاهـاً ممکـن اسـت برنامـهٔ کـاربردی اقـدام بـه دریافت داده های بافر شـدهٔ خـود در مـهلت مقـرر نکـرده و بـافر پـر شـود. در ایـن حـالت گیرنده دیگر قادر بـه دریـافت و ذخـیرهٔ داده هـا در بـافرش نخواهـد بـود ، بـهمین دلیـل در هر بستهٔ TCP کـه بـه طـرف دیگـر ارسـال می‌شـود حجـم فضـای آزاد بـافر ، در ایـن فیلـد اعـلام خواهـد شـد. بعنـوان مثـال اگـر در یـک بسـتهٔ دریـافتی مقـدار فیلـد Window Size مقدار ٤٠٩٦ باشد بدیـن معناسـت کـه از کـل فضـای بـافر موجـود ، فعـلاً چـهار کیلوبـایت از آن خالی است. برای روشن شـدن قضیـه بـه شـکل (۵-۵) توجـه کنیـد.

نام متغیر	توضیح
	متغیرهای نظارت بر ارسال داده ها
SND.UNA	شمارهٔ ترتیب آخرین بسته ای که ارسال شده ولی هنوز پیغام Ack آن برنگشته است.
SND.NXT	شمارهٔ ترتیب آخرین بایت که داده ها از آن شماره به بعد در بستهٔ بعدی که باید ارسال شود.
SND.WND	میزان فضای آزاد در بافر ارسال
SND.UP	شمارهٔ ترتیب آخرین داده های اضطراری که تحویل برنامهٔ کاربردی شده است.
SND .WL1	
SND.WL2	
SND.PUSH	شمارهٔ ترتیب آخرین داده هایی که باید آنی به برنامهٔ کاربردی گسیل (Push) شود.
SND.ISS	مقدار اولیهٔ شمارندهٔ ترتیب داده های دریافتی که در حین ارتباط بر روی آن توافق می‌شود.
	متغیرهای نظارت بر دریافت داده ها
RCV.NXT	شمارهٔ ترتیب آخرین بایت در بستهٔ بعدی که از آن شماره به بعد انتظار دریافت آنرا دارد.
RCV.WND	میزان فضای آزاد در بافر دریافت
RCV.UP	شمارهٔ ترتیب آخرین داده های اضطراری که برای برنامهٔ طرف مقابل ارسال شده است.
RCV.IRS	مقدار اولیهٔ شمارندهٔ ترتیب داده های ارسالی که در حین ارتباط بر روی آن توافق می‌شود.

جدول (٤-۵) برخی از متغیرهای ساختمان دادهٔ TCB



شکل (۵-۵) مثالی از روند کنترل جریان در پروتکل TCP

در این پروتکل به ازای هر ارتباط TCP که موفقیت آمیز برقرار شود ، یک "ساختمان دادهٔ" خاص برای آن ایجاد خواهد شد که اطلاعاتی از آخرین وضعیت ارسال یا دریافت جریان داده ها در آن نگهداری می‌شود. این ساختمان داده ، "بلوک نظارت بر انتقال"[1] یا اختصاراً TCB نامیده می‌شود. برخی از متغیرهای تعریف شده درون ساختمان دادهٔ TCB در جدول (٤-۵) معرفی شده است.

عملکرد این متغیرها با تعریفی که از فیلدهای یک بستهٔ TCP داشتیم واضح و مشخص است.

[1] Transmission Control Block

۶) زمان سنجها در پروتکل TCP[۱]

عملکرد صحیح پروتکل TCP وابستگی شدیدی به استفادهٔ درست و منطقی از زمان سنجها دارد. در این بخش مهمترین زمان سنجهای بکار رفته در این پروتکل را بررسی می‌نماییم :

- Retransmission Timer : به گونه ای که اشاره شد پس از برقراری یک ارتباط ، وقتی بسته ای برای پروسهٔ مقصد ارسال میشود ، ضمن نگهداری موقت آن در یک بافر ، برای آن یک زمان سنج تنظیم و فعّال میشود و اگر در مهلت مقرّر پیغام دریافت آن (Ack) نرسید ، آن بسته از نو برای مقصد ارسال خواهد شد. این زمان سنج که اختصاراً **RT** نامیده میشود به یک مقدار پیش فرض ، مقداردهی میشود و شروع به شمارش معکوس زمان می‌نماید؛ هرگاه مقدار آن زمان سنج به صفر برسد ولی پیغام دریافت بسته برنگردد ، "رخداد انقضای زمان تکرار"[۲] حادث شده و پروسهٔ TCP را وادار به ارسال مجدد آن بسته می‌کند و مراحل قبلی از نو تکرار می‌شود.

عملکرد این زمان سنج بسیار ساده است ولی مسئلهٔ بغرنج در شبکه آنست که : اولاً پیش فرض این زمان سنج چه مقداری باشد؟ ثانیاً عمل ارسال مجدّد یک بسته چند بار تکرار شود؟
در شبکه های محلی سریع ، زمان رفت یک بسته و برگشت پیغام دریافتِ آن ، حدود چند هزارم ثانیه طول خواهد کشید در حالی که در شبکهٔ WAN این زمان رفت و برگشت میتواند تا چندین ثانیه طول بکشد.
اگر قرار باشد زمان پیش فرض زمان سنج RT به مقداری کم تنظیم شود ، آنگاه وقتی مقصد روی یک شبکهٔ راه دور واقع است ، قبل از آنکه بسته بتواند به مقصد برسد ، مهلت این زمان سنج منقضی شده و بسته مجدداً ارسال میشود و این کار برای هر بسته بطور متوالی تکرار می‌شود و ترافیک زائد و بیهوده ای را به شبکه تحمیل میکند.
از طرف دیگر اگر قرار باشد زمان پیش فرض این زمان سنج با مقداری بزرگ تنظیم شود در شبکه های محلی و سریع ، هنگام بروز یک خطا تاخیر زیادی بوجود خواهد آمد.
بهترین راه تنظیم زمان سنج استفاده از روشهای وفقی و پویا است چراکه راندمان پروتکل TCP به شدت به آن وابسته است. الگوریتم پویایی که معمولاً در پیاده سازی TCP بکارگرفته میشود در زیر معرفی شده است[۳] :

---

[۱] TCP Timers
[۲] Retransmission Timeout Event
[۳] این الگوریتم در سال ۱۹۸۸ توسط Jacobson معرفی شد.

الف) وقتی یک ارتباط TCP برقرار میشود یک متغیر حافظه به نـام RTT متنـاظر بـا آن ایجـاد شده و به یک مقدار پیش فرض مقداردهی میشود. این مقدار کـه حدکـثر زمـان انتظـار بـرای برگشت پیغام دریافت بسته محسوب میشود ممکن است اصلاً بهینه و مناسب نباشد.
ب) به ازای هر بسته که ارسال میشود یک زمان سنج متناظر با آن بسته تنظیم می‌شود که زمان رفت بسته و برگشت پیغام دریافت آنرا اندازه میگیرد؛ فرض کنید برای یـک بسـته ایـن زمـان مقدار M محاسبه شده باشد.
ج) مقدار پیش فرض زمان سنج طبق رابطهٔ زیر بهنگام میشود:

$$RTT_{new}=RTT_{old}+4*D_{new}$$

$$D_{new}=\alpha.D_{old}+(1-\alpha).(RTT_{old}-M)$$

$\alpha=7/8$ ، مقدار اولیهٔ D می‌تواند صفر باشد.

تبصره : مقدار جدید برای زمان سنج فقط به شرطی اعمال می‌شود که ارسال بستهٔ TCP تکـرار نشده باشد و فقط یکبار ارسال شده باشد[۱].

- **Keep-Alive Timer** : ممکن است طرفین یک ارتباط به هر دلیلی ارسال اطلاعات را موقتاً متوقف کنند و هیچ دادهای مبادله نشود ، هرچند ارتباط TCP فعّال و باز باشد. از سوی دیگـر ممکن است یکی از طرفین به دلیلی مثل خرابی سخت افزاری یا نرم افـزاری ، بـدون اطـلاع ، ارتباط را رها کرده باشد. برای تمایز بین این دو حالت ، فرستندهٔ اطلاعات بـا اسـتفاده از ایـن زمان سنج در بازههای زمانی منظم یک بستهٔ TCP که خالی از هرگونـه دادهای می‌باشـد بـرای مقصد ارسال می‌شود و در صورتی که پیغام دریافت آن باز گشـت ، نشـان دهنـدهٔ آنسـت کـه ارتباط TCP فعال و باز است؛ در غیر این صورت ارتباط TCP بصورت یکطرفه قطـع شـده و تمام بافرها و فضای ایجاد شده آزاد می‌شوند. زمان پیش فرض این زمان سنج مقداری بین ۵ تا ٤٥ ثانیه می‌باشد.
- **Persistence Timer** : در پروتکل TCP وقتی یکی از طرفین ارتباط ، مقدار فضای بـافر آزاد خود را در فیلد Window Size صفر اعلام کند ، ناگزیر پروسهٔ طـرف مقـابل متوقـف (بلوکـه) خواهد شد. در چنین حالتی پس از آنکه مقداری از فضـای بـافر پـر شـده تخلیـه شـد ، ایـن موضوع باید به طرف مقابل گزارش شود تا سیستم عامل ، پروسهٔ بلوکه شـده را احیـا کـرده و ادامهٔ ارسال ممکن باشد. در غیر اینصورت "بن بست"[۲] و تاخیر بینهایت بـرای پروسـه بوجـود خواهد آمد.

---

[۱] Karn's Algorithm
[۲] Deadlock

با استفاده از این زمان سنج پس از آزاد شدن فضای بافر ، در فواصل زمانی منظم یک بستهٔ TCP برای پروسهٔ بلوکه شده ارسال می‌شود تا ضمن آگاهی از آخرین وضعیت فضای بافر پروسه بتواند احیا شود.

- **Quiet Timer** : ممکن است یک ارتباط TCP بسته شود ولی هنوز بستههایی سرگردان بر روی شبکه وجود داشته باشد که پس از بسته شدن ارتباط TCP به مقصد برسند ، لذا در این پروتکل پس از بسته شدن یک ارتباط با شمارهٔ پورت خاص ، بقیهٔ پروسه ها حق استفاده از شمارهٔ پورتی که اخیراً بسته شده را ندارند. مقدار پیش فرض این زمان سنج دقیقاً دو برابر مقدار پیش فرض زمان حیات بستهٔ IP بر حسب ثانیه است. (چیزی بین ۳۰ تا ۱۲۰ ثانیه)
- **Idle Timer** : این زمان سنج برای آن است که اگر تلاش برای تکرار ارسال یک بسته بیش از حد متعارف انجام شود ارتباط TCP را بصورت یکطرفه رها و قطع نماید. مقدار معمول این زمان سنج ۳٦۰ ثانیه (٦ دقیقه) است.

۷) پروتکل UDP

پروتکل TCP پروتکلی "اتصالگرا" است و لزوم برقراری یک ارتباط قبل از هرگونه مبادلهٔ داده ، می‌تواند بین چند میلی ثانیه (برای شبکه های محلی سریع) تا چندین ثانیه (برای شبکه های WAN) طول بکشد؛ در ضمن تامل برای بازگشت پیغامهای Ack ، یک پروسهٔ کاربردی را با تاخیر مواجه خواهد کرد. برای برخی از کاربردها این زمان قابل تحمل نیست و سرعت در رسیدن یک بسته به مقصد ، ضروریتر از پرداختن به مسائلی از قبیل بررسی شمارهٔ ترتیب و ارسال پیغامهای کنترلی محسوب میشود. (کاربردهایی مثل سیستم DNS یا TFTP که در بخشهای آتی بررسی میشوند.)

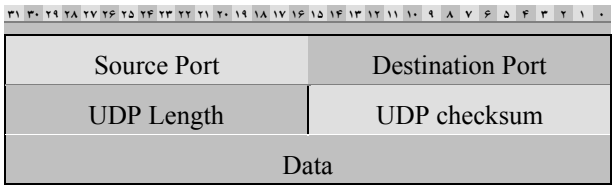
در لایهٔ انتقال از مدل TCP/IP برای چنین کاربردهایی یک پروتکل ساده و سریع به نام UDP معرفی شده است که به صورت ذاتی "بدون اتصال"[۱] است ، یعنی بدون هیچ اطلاعی از سرنوشتی که در انتظار یک بسته است ، به سمت مقصد ارسال میشود. هرگونه اطلاعی از رسیدن یا نرسیدن داده ها باید در لایه بالاتر بررسی و مدیریت شود.

پروتکل UDP ، تمام کاستی های لایهٔ IP را دارد (به غیر از نظارت بر خطای کانال که میتواند وجود داشته باشد) و تنها ارمغان این پروتکل برای پروسه ها سرعت ارسال و کم شدن تأخیرات ناشی از نظارت بر جریان بسته هاست.

---

[۱] Connectionless

در ادامه ساختار بسیار سادهٔ یک بستهٔ UDP را بررسی میکنیم. در شکل (٦-٥) ساختار یک بستهٔ UDP به تصویر کشیده شده است.



شکل (٦-٥) ساختار یک بستهٔ UDP

- **فیلد Source Port** : در این فیلد ، یک شمارهٔ ١٦ بیتی بعنوان آدرس پورت پروسهٔ مبدأ که این بسته را جهت ارسال ، تولید کرده ، قرار خواهد گرفت.
- **فیلد Destination Port** :در این فیلد ، آدرس پورتِ پروسهٔ مقصد که آنرا تحویل خواهد گرفت ، تعیین خواهد شد .

همانگونه که در بخش قبلی اشاره شد این دو آدرس مشخص می‌کنند که این بسته از کدام برنامهٔ کاربردی در لایه بالاتر تولید و باید به چه برنامه ای در ماشین مقصد تحویل داده شود.

- **فیلد UDP Length** : در این فیلد طول بستهٔ UDP برحسب بایت ، شامل سرآیند و داده ها ، درج میشود.
- **فیلد UDP Checksum** : در این فیلدِ ١٦ بیتی کد کشف خطا درج میشود. روش محاسبهٔ این کد دقیقاً همانند روشی است که در پروتکل TCP معرفی شد. تنها تفاوت در آنست که بکارگیری این فیلد اختیاری است و در صورت عدم نیاز به آن ، تمام بیتهای آن به صفر تنظیم می‌شود. (برای کاربردهایی مثل ارسال دیجیتال صدا یا تصویر)

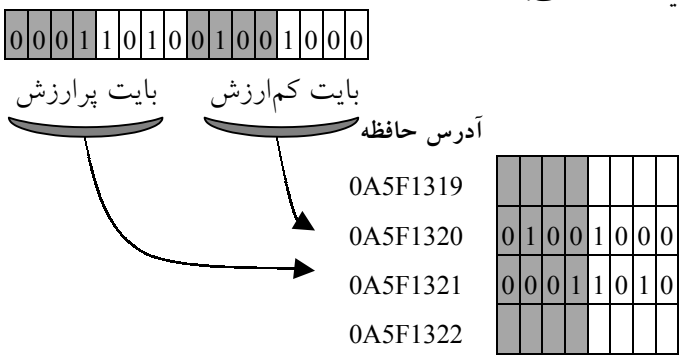
مناسبترین کاربرد پروتکل UDP برای پروسه هایی است که عملیاتشان مبتنی بر یک تقاضا و یک پاسخ است. (سیستم DNS)

نکات بکارگیری و ظرائف پروتکلهای TCP و UDP در فصل برنامه نویسی تحت شبکه مجدداً بررسی می‌شوند.

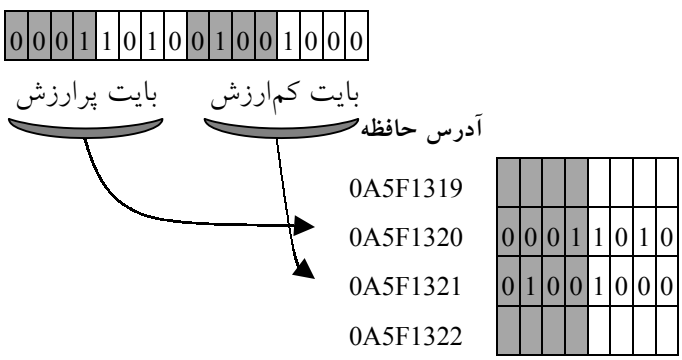
۸) ماشینهای Big Endian و Little Endian

ساختار بسته‌های IP و TCP در قالب کلمات ۳۲ بیتی سازماندهی می‌شوند. ماشینهای متفاوتی که در دنیا وجود دارد کلمات دوبایتی و چهاربایتی (۳۲ بیتی) را به یک نحو درون حافظه ذخیره نمی‌کنند . در برخی از ماشینها برای ذخیره یک کلمه در حافظه ابتدا بایت کم ارزش در حافظه ذخیره شده و پشت سر آن بایت پرارزش ذخیره می‌شود ، در حالی که در برخی دیگر دقیقاً برعکس است. بعنوان مثال فرض کنید کلمهٔ ۲ بایتی (0x1A48) با دستور Mov به حافظهٔ اصلی منتقل شود. ماشینها به دو روش آنرا در حافظهٔ خود ذخیره می‌کنند :

- ماشینهایی که ابتدا بایت کم ارزش و سپس بایت پرارزش را ذخیره می‌کنند ، ماشینهای Little Endian نامیده می‌شوند. کامپیوترهای شخصی با پردازنده سری 80X86 و پنتیوم از این دسته هستند. در مثال زیر طریقهٔ ذخیره‌سازی یک کلمهٔ ۱۶ بیتی در فضای حافظهٔ یک ماشین .L.E به تصویر کشیده شده است.



- ماشینهایی که ابتدا بایت پرارزش و سپس بایت کم‌ارزش را ذخیره می‌کنند ، ماشینهای Big Endian نامیده می‌شود. کامپیوترهای سری SUN از این دسته هستند. در مثال زیر طریقهٔ ذخیره‌سازی یک کلمهٔ ۱۶ بیتی در فضای حافظهٔ یک ماشین .B.E به تصویر کشیده شده است.



از آنجایی که بسته‌های IP ابتدا در حافظه تشکیل و سپس از طریق سخت‌افزار کارتِ شبکه ارسال می‌شوند لذا اگر بسته IP که بصورت سریال روی خط ارسال می‌شود از یک ماشین Little Endian تولید و توسط یک ماشین Big Endian دریافت شود ، ممکن است جای بایتها عوض شده و محتوی بسته‌ها اشتباه و فاقد ارزش دریافت شوند.

پروتکل TCP/IP ، استاندارد ماشینهای Big Endian را مبنا قرار داده است ، لذا در تمام ماشینهای Little Endian ، قبل از ارسال بسته‌های IP باید فیلدهای دوبایتی و چهاربایتی درون حافظه ، بگونه‌ای تنظیم و مقداردهی شوند تا در هنگام ارسال بسته روی خط ، ابتدا بایت پرارزش ارسال شده و استاندارد ماشینهای .B.E رعایت شود.

۹) شمارهٔ پورتهای استاندارد

در جدول (۷-۵) شمارهٔ پورتهای استاندارد که توسط IETF به عنوان استاندارد پذیرفته شده‌اند ، ارائه شده است.

پورت	نام اختصاری	نام پروسه
1	TCPMUX	TCP Port Service Multiplexer
5	RJE	Remote Job Entry
7	ECHO	Echo
9	DISCARD	Discard
11	USERS	Active Users
13	DAYTIME	Daytime
17	Quote	Quote of the Day
19	CHARGEN	Character Generator
20	FTP-DATA	File Transfer (Data Channel)
21	FTP	File Transfer (Control Channel)
23	TELNET	TELNET
25	SMTP	Simple Mail Transfer
27	NSW-FE	NSW User System FE
29	MSG-ICP	MSG-ICP
31	MSG-AUTH	MSG Authentication
33	DSP	Display Support Protocol
35	PPS	Private Printer Server
37	TIME	Time
39	RLP	Resource Location Protocol
41	GRAPHICS	Graphics
42	NAMESERVER	Host Name Server
43	NICNAME	Who Is
49	LOGIN	Login Host Protocol
53	DOMAIN	Domain Name Server

67	BOOTPS	Bootstrap Protocol Server
68	BOOTPC	Bootstrap Protocol Client
69	TFTP	Trivial File Transfer Protocol
79	FINGER	Finger
101	HOSTNAMENIC	Host Name Server
102	ISO-TSAP	ISO TSAP
103	X400	X.400
104	X400SND	X.400 SND
105	CSNET-NSCSNET	Mailbox Name Server
109	POP2	Post Office Protocol v2
110	POP3	Post Office Protocol v3
111	SUNRPC	SUN RPC Portmap
137	NETBIOS-NS	NETBIOS Name Service
138	NETBIOS-DGMNET	BIOS Datagram Service
139	NETBIOS-SSNNET	BIOS Session Service
146	ISO-TP0	ISO TP0
147	ISO-IP	ISO IP
150	SQL-NET	SQL-NET
153	SGMP	SGMP
156	SQLSRV	SQL Service
160	SGMP-TRAP5	SGMP TRAPS
161	SNMP	SNMP
162	SNMPTRAP	SNMPTRAP
163	CMIP-MANAGE	CMIP/TCP Manager
164	CMIP-AGENT	CMIP/TCP Agent
165	XNS-COURIER	Xerox Network
179	BGP	Border Gateway Protocol

(۷-۵) شمارهٔ پورتهای استاندارد

۱۰) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.

RFC 1072	"TCP Extensions for Long-Delay Paths," Jacobson, V.; Braden, R.T.; 1988
RFC 896	"Congestion Control in IP/TCP Internetworks," Nagle, J.; 1984
RFC 879	"TCP Maximum Segment Size and Related Topics," Postel, J.B.; 1983
RFC 813	"Window and Acknowledgment Strategy in TCP," Clark, D.D.; 1982
RFC 793	"Transmission Control Protocol," Postel, J.B.; 1981
RFC 768	"User Datagram Protocol," Postel, J.B.; 1980