فصل 6

برنامه نویسی شبکه Network Programing :

برای کسب توانایی در ساخت برنامه هایی که بتوانند تحت شبکه و استاندارد های موجود برای ساخت شبکه کارکنند نیاز است اطلاعات مقدماتی در مورد شبکه های کامپیوتری داشته باشید در ابتدا ما به معرفی کوتاه در این زمینه می پردازیم .

سرور (server) چیست ؟

سِرور که در برخی متون فارسی کارساز یا خادم هم نامیده می‌شود، به برنامه‌ای رایانه‌ای گفته می‌شود که خدمات خود را به دیگر برنامه‌های رایانه‌ای (و کاربران آن‌ها) در همان رایانه یا در رایانه‌های دیگر ارائه می‌کند. به رایانه‌ای که چنین برنامه‌ای روی آن اجرا شود نیز *کارساز* گفته می‌شود.

کلاینت (client) چیست ؟

کارخواه ، یک نرم افزار کاربردی یا سامانه است که از طریق یک شبکه به خدمات یک سامانهٔ رایانه‌ای دیگر به نام سرور یا کارساز دسترسی دارد.این عبارت نخستین بار برای افزاره‌هایی که قابلیت اجرای برنامه‌های مستقل خودشان را نداشتند اما می‌توانستند با رایانه‌های دور از طریق شبکه برهم‌کنش داشته باشند، به کار رفت. مدل کارخواه-کارساز امروزه نیز در اینترنت به کار می‌رود. مرورگرهای وب، کارخواه‌هایی هستند که به کارسازهای وب وصل می‌شوند و صفحات وب را برای نمایش بازیابی می‌کنند.

میزبان (Host) چیست ؟

در شبکه‌های رایانه‌ای، رایانه‌ای است که به اینترنت -یا به طور کلی‌تر- به هر شبکهٔ داده‌ای، متصل است. یک میزبان شبکه می‌تواند اطلاعات و نیز نرم‌افزار کارخواه (client) و/یا کارساز (server) را میزبانی کند.هر میزبان اینترنتی یک نشانی آی‌پی یکتا دارد که بخش نشانی میزبان را نیز شامل می‌شود. نشانی میزبان، یا به طور دستی توسط مدیر رایانه وارد می‌شود و یا به طور خودکار در آغاز بوسیلهٔ پروتکل پیکربندی پویای میزبان (DHCP)اختصاص می‌یابد.

هر میزبانی، یک گرهٔ شبکهٔ فیزیکی (دستگاه شبکه) است، ولی هر گرهٔ شبکهٔ فیزیکی، یک میزبان نیست. به گره‌های شبکه مانند مودم‌ها و سوئیچ‌های شبکه نشانی‌های میزبان اختصاص نمی‌یابد، و به عنوان میزبان در نظر گرفته نمی‌شوند. به دستگاه‌هایی چون چاپگرهای شبکه و مسیریاب‌های سخت‌افزاری، نشانی آی‌پی میزبان اختصاص می‌یابد، اما چون آن‌ها رایانه‌های همه‌منظوره نیستند، به طور عمومی گاهی به عنوان میزبان درنظر گرفته نمی‌شوند.

سوکت (socket) چیست ؟

اصلی ترین عامل در یک ارتباط شبکه ای Socket میباشد که اعمال شبکه را بصورت خواندن و نوشتن در یک فایل شبیه سازی نموده است . سوکت در اصل مانند یک کانال ارتباطی است که میان دو نقطه ایجاد شده و ارتباط را برقرار می سازد . برای داشتن یک ارتباط شبکه ای باید یک سوکت ایجاد کنیم که لازمه این کار داشتن آدرس IP ، نوع پروتکل ارتباط (TCP / UDP) و شماره Port مقصد می باشد.

پروتکل (protocol) چیست ؟

در شبکه های کامپیوتر به مجموعه قوانینی اطلاق می‌گردد که نحوه ارتباط را قانونمند می‌نماید. نقش پروتکل در کامپیوتر نظیر نقش زبان برای انسان است.برای مطالعه یک کتاب نوشته شده به فارسی می‌بایست خواننده

شناخت مناسبی از زبان فارسی را داشته باشد. به منظور ارتباط موفقیت آمیز دو دستگاه در شبکه نیز باید هر دو دستگاه از یک پروتکل مشابه استفاده کنند.

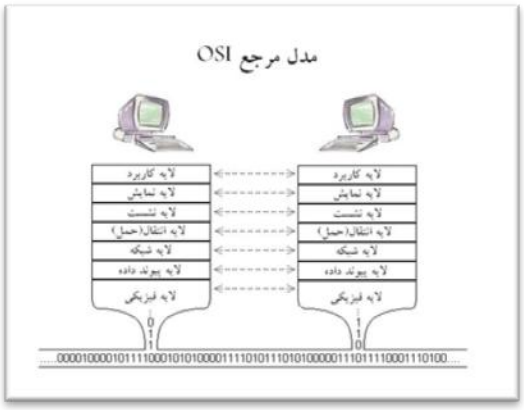
در علوم کامپیوتر و ارتباطات، پروتکل عبارت است از استاندارد یا قراردادی که برای ارتباط میان دو نقطه برقرار می‌شود. پروتکل اتصال بین دو ند، انتقال داده بین آن دو و تبادلات میان را ممکن کرده و آن را کنترل می‌کند. پروتکل در ساده‌ترین حالت می‌تواند به عنوان قوانین ادارهٔ منطق، ترکیب و همزمانی ارتباطات در نظر گرفته شود. پروتکل‌ها ممکن در سخت‌افزار یا نرم‌افزار یا ترکیبی از این دو پیاده سازی شوند. پروتکل در پایین‌ترین سطح رفتار اتصال سخت‌افزاری را تعریف می‌کند. معنی لغوی پروتکل مجموعه قوانین است.

مدل OSI :

مدل مرجع اتصال داخلی سیستم‌های باز که گاه «مدل هفت‌لایه ASI» نیز خوانده می‌شود، توصیفی مفهومی و مجرد از لایه‌هایی است که دو یا چند سیستم مخابراتی یا شبکه های کامپیوتری از طریق آن به یکدیگر متصل می‌شوند.این مدل خود یک معماری شبکه نیست چون هیچ سرویس یا پروتکلی در آن تعریف نمی‌شود.

اگر صفرها و یک‌ها همینطور پشت سر هم قرار بگیرند اطلاعات منتقل نمی شود. بلکه باید درباره نحوه ارسال و شکل اطلاعات توافق شود. برای این منظور در شبکه پروتکل هایی به وجود آمد OSI .یکی از مدل‌های استاندارد و پذیرفته شده است که برای استفاده پروتکل‌ها در شبکه به کار می رود.

در مدل OSIشبکه را به هفت لایه تقسیم می کنند که این لایه ها مستقل از هم هستند و هر لایه کاری را انجام می دهد و وظیفه دارد که به لایه های بالاتر سرویس بدهد .ارتباط بین دو لایه Interface نام دارد و لایه های متناظر پروتکل متناظر دارند .این هفت لایه به صورت زیر می باشند:



لایه ها در مدل OSI :

لایه یک (Physical) :

این لایه با مشخصات فیزیکی محیط های انتقال مانند کابل ها و کانکتورها در ارتباط است در این لایه مشخصات مکانیکی ، فیزیکی و عملیاتی محیط انتقال همچون ولتاژ الکتریکی ، سرعت و فاصله مجاز تعریف شده است.

لایه دو (Data) :

این لایه فراهم کننده ارتباط با لایه فیزیکی و محیط انتقال است .این لایه با آدرس های فیزیکی کار می کند و بر اساس آن سخت افزارها یا لایه Data Link دیگری را پیدا می کنند این لایه وظیفه کشف خطا و تشخیص صحت داده رسیده را نیز برعهده دارد .از پروتکل های این لایه می توان FDDI, Ethernet نام برد.

لایه سه (Network) :

وظیفه اصلی لایه شبکه مسیریابی و هدایت بسته بین مسیرهای مختلف تا رسیدن به مقصد است.این لایه بر خلاف لایه دوم که با آدرس های فیزیکی کار می کند از IP Address جهت شناخت مقصد استفاده می کند Routerکه یکی از ابزارهای مسیریاب است در این لایه عمل میکند.
پروتکل های IP,IPX در این لایه عمل می کنند.

لایه چهار (Transport) :

از وظایف لایه انتقال می توان ، اطمینان از رسیدن داده ها به مقصد فرستادن پیغام پس از دریافت بسته کنترل و ارسال بسته ها و ارسال مجدد بسته های خراب و تقسیم بسته های بزرگ به قطعات کوچک قابل انتقال نام برد.

لایه پنج (Session) :

لایه جلسه وظیفه برقراری تماس بین مبدا و مقصد و قطع آن پس از اتمام ارتباط را برعهده دارد .همچنین در صورتی که حین انتقال ارتباط قطع شود این لایه وظیفه دارد مجددا ارتباط را برقرار کند تا ارسال داده ها از سر گرفته شود.

لایه شش (Presentation) :

وظیفه این لایه ترجمه کدهای رسیده از لایه های پایینتر به Format قابل استفاده توسط برنامه های کاربردی مانند Mail می باشد. در این لایه عملیات رمز گذاری و فشرده سازی انجام می شود.

لایه هفت (Application) :

در این لایه برنامه های کاربردی قرار دارند که کاربران با آنها در تعامل هستند مانند : انواع Browser ها ، برنامه های ارسال Mail و غیره.

مدل TCP/IP :

مدل TCP/IP یا مدل مرجع اینترنتی که گاهی به مدل(DOD وزارت دفاع)، مدل مرجع ARPANET نامیده می‌شود، یک توصیف خلاصه لایه TCP/IP برای ارتباطات و طراحی پروتکل شبکه کامپیوتراست . TCP/IPدر سال ۱۹۷۰ بوسیله DARPAساخته شده که برای پروتکل‌های اینترنت در حال توسعه مورد استفاده قرار گرفته است، ساختار اینترنت دقیقاًبوسیله مدل TCP/IP منعکس شده‌است.

مدل اصلی TCP/IP از ۴ لایه تشکیل شده‌است. هرچند که سازمان IETF استانداردی که یک مدل ۵ لایه‌ای است را قبول نکرده‌است.به هر حال پروتکل‌های لایه فیزیکی ولایه پیوند داده‌ها بوسیله IETF استاندارد نشده‌اند. سازمان IETF تمام مدل های لایه فیزیکی را تایید نکرده‌است. با پذیرفتن مدل ۵ لایه‌ای در بحث اصلی بامسولیت فنی برای نمایش پروتکل می‌باشد عجیب نیست که نمایش ۵لایه‌ای را درآموزش بیاوریم واین امکان را می‌دهد که راجع به پروتکل‌های غیر IETF در لایه فیزیکی صحبت کنیم. این مدل قبل از مدل مرجع OSIگسترش یافته و واحد وظایف مهندسی اینترنت(IETF) ، برای مدل و پروتکل‌های گسترش یافته تحت آن پاسخگو است، هیچ گاه خود را ملزم ندانست که توسط OSI تسلیم شود. درحالیکه مدل بیسیک OSI کاملاً در آموزش استفاده شده‌است و OSI به یک مدل ۷ لایه‌ای معرفی شده‌است، معماری یک پروتکل واقعی (RFC ۱۱۲۲)مورد استفاده در محیط اصلی اینترنت خیلی منعکس نشده‌است. حتی یک مدرک معماری IETFکه اخیرا منتشر شده یک مطلب با این عنوان دارد: " لایه بندی مضر است ". تاکید روی لایه بندی به عنوان محرک کلیدی معماری یک ویژگی از مدل TCP/IP نیست، اما نسبت به OSI بیشتر است.

لایه ها در مدل TCP/IP :

لایه کاربرد (Application) :

لایه کاربردی بیشتر توسط برنامه‌ها برای ارتباطات شبکه استفاده می‌شود .داده‌ها از برنامه در یک قالب خاص برنامه عبور می‌کنند سپس در یک پروتکل لایه انتقال جاگیری می‌شوند. از آنجاییکه پشته IPبین لایه‌های

Application (کاربردی) و Transport (انتقال) (هیچ لایه دیگری ندارد، لایه کاربردی Application می‌بایست هر پروتکلی را مانند پروتکل لایه نشست (session) و نمایش (presentation) در OSI عمل می‌کنند در بگیرد. داده‌های ارسال شده روی شبکه درون لایه کاربردی هنگامیکه در پروتکل لایه کاربردی جاگیری شدند عبور می‌کنند. از آنجا داده‌ها به سمت لایه‌های پایین تر پروتکل لایه انتقال می‌روند. دو نوع از رایجترین پروتکل‌های لایه پایینی TCP و UDP هستند .سرورهای عمومی پورتهای مخصوصی به اینها دارند (HTTPپورت ۸۰و FTP پورت ۲۱ را دارند و...) در حالیکه کلاینتها از پورتهای روزانه بی دوام استفاده می‌کنند. روترها و سوئیچها این لایه را بکار نمی‌گیرند اما برنامه‌های کاربردی بین راه در در پهنای باند این کار را می‌کنند، همانطور که پروتکل (RSVPپروتکل ذخیره منابع) انجام می‌دهد.

لایه انتقال (transport) :

مسئولیتهای لایه انتقال، قابلیت انتقال پیام را END-TO-END و مستقل از شبکه، به اضافه کنترل خطا، قطعه قطعه کردن و کنترل جریان را شامل می‌شود .ارسال پیام END-TO-END یا کاربردهای ارتباطی در لایه انتقال می‌توانند جور دیگری نیز گروه بندی شوند :.۱ اتصال گرا مانند TCP ۲ .بدون اتصال مانند UDP لایه انتقال می‌تواند کلمه به کلمه به عنوان یک مکانیزم انتقال مانند یک وسیله نقلیه که مسئول امن کردن محتویات خود (مانند مسافران و اشیاء) است که آنها را صحیح و سالم به مقصد برساند، بدون اینکه یک لایه پایین تر یا بالاتر مسئول بازگشت درست باشند. لایه انتقال این سرویس ارتباط برنامه‌های کاربردی به یکدیگر را در حین استفاده از پورتها فراهم آورده‌است. از آنجاییکه IP فقط یک delivery فراهم می‌آورد، لایه انتقال اولین لایه پشته TCP/IPبرای ارائه امنیت و اطمینان است. توجه داشته باشید که IPمی‌تواند روی یک پروتکل ارتباط داده مطمئن امن مانند کنترل ارتباط داده سطح بالا (HDLC)اجرا شود. پروتکل‌های بالای انتقال مانند RPCنیز می‌توانند اطمینان را فراهم آورند. بطور مثال TCPیک پروتکل اتصالگر است که موضوع‌های مطمئن بیشماری را برای فراهم آوردن یک رشته بایت مطمئن و ایمن آدرس دهی می‌کند : داده in orderمی‌رسند. داده‌ها حداقل خطاها را دارند. داده‌های تکراری دور ریخته می‌شوند. بسته‌های گم شده و از بین رفته دوباره ارسال می‌شوند .دارای کنترل تراکم ترافیک است SCTP .جدیدتر نیز یک مکانیزم انتقالی مطمئن و امن و اتصالگراست -رشته پیام گراست نه رشته بایت گرا مانند - TCP و جریانهای چندگانه‌ای را روی یک ارتباط منفرد تسهیم می‌کند. و همچنین پشتیبانی چند فضا را (multi-homing) نیز در مواردی که یک پایانه ارتباطی می‌تواند توسط چندین آدرس IPبیان شود.(اینترفیس‌های فیزیکی چندگانه) را فراهم می‌آورد تا اینکه اگر یکی از آنها دچار مشکل شود ارتباط دچار وقفه نشود. در ابتدا برای کاربردهای تلفنی (برای انتقالSS۷ روی (IPاستفاده می‌شود اما می‌تواند برای دیگر کاربردها نیز مورد استفاده قرار بگیرد.

لایه شبکه (Network) :

لایه شبکه مشکل گرفتن بسته‌های سرتاسر شبکه منفرد را حل کرده‌است .نمونه‌هایی از چنین پروتکل‌هایی X.۲۵ و پروتکل HOST/IMP مربوط به ARPANET است. با ورود مفهوم درون شبکه‌ای کارهای اضافی به این لایه اضافه می‌شوند از جمله گرفتن از شبکه منبع به شبکه مقصد و عموماً routing کردن و تعیین مسیر بسته‌های میان یک شبکه از شبکه‌ها را که به‌عنوان شبکه داخلی یا اینترنت شناخته می‌شوند را شامل می‌شود. در همه پروتکل‌های شبکه IP وظیفه اساسی گرفتن بسته‌های داده‌ای را از منبع به مقصد انجام می‌دهد IP . می‌تواند داده‌ها را از تعدادی از پروتکل‌های مختلف لایه بالاتر حمل کند .این پروتکل‌ها هرکدام توسط یک شماره پروتکل واحد و منحصر به فرد شناسایی می‌شوند ICMP:و IGMP به ترتیب پروتکل‌های ۱و۲ هستند. برخی از پروتکل‌های حمل شده توسط IP مانند) ICMP مورد استفاده برای اطلاعات تشخیص انتقال راجع به انتقالات) IGNP , (IP مورد استفاده برای مدیریت داده‌های multicast در (IPدر بالای IP لایه بندی شده‌اند اما توابع لایه داخلی شبکه را انجام می‌دهند، که یک ناهمسازی بین اینترنت و پشته IP و مدل OSI را ایجاد کرده‌اند. تمام پروتکل‌های مسیریابی مانند OSPT و RPTنیز بخشی از لایه شبکه هستند. آنچه که آنها را بخشی از لایه شبکه کرده‌است این است که هزینه load آنها (play load) در مجموع با مدیریت لایه شبکه در ارتباط است. کپسول بندی و جاگیری خاص آن به اهداف لایه بندی بی ارتباط است.

لایه ارتباط داده ها (Data link) :

لایه ارتباط داده از متدی که برای حرکت بسته‌ها از لایه شبکه روی دو میزبان مختلف که در واقع واقعاً بخشی از پروتکلهای شبکه نیستند، استفاده می‌کند، چون IPمی‌تواند روی یک گستره ار لایه‌های ارتباطی مختلف اجرا شود .پردازشهای بسته‌های انتقال داده شده روی یک لایه ارتباطی داده شده می‌تواند در راه انداز وسایل نرم‌افزاری برای کارت شبکه به خوبی میان افزارها یا چیپ‌های ویژه کار صورت گیرد. این امر می‌تواند توابع ارتباط داده‌ها را مانند اضافه کردن یک headerبسته به منظور آماده کردن آن برای انتقال انجام دهد سپس واقعاً فرم را روی واسط فیزیکی منتقل کند. برای دسترسی اینترنت روی یک مودم dial-up معمولاً بسته‌های IPبا استفاده از PPP منتقل می‌شوند .برای دسترسی به اینترنت با پهنای باند بالا مانند ADSLیا مودم‌های کابلی PPPOEغالباً استفاده می‌شود. در یک شبکه کابلی محلی معمولاً اترنت استفاده می‌شود و دو شبکه‌های بی سیم محلیIEEE ۸۰۲٫۱۱ معمولاً استفاده می‌شود .برای شبکه‌های خیلی بزرگ هردو روش PPP یعنی خطوط T-Carrierیا E-Carrier تقویت کننده فرم، ATMیا بسته روی SONET/SDM (POS) اغلب استفاده می‌شوند .لایه ارتباطی همچنین می‌تواند جاییکه بسته‌ها برای ارسال روی یک شبکه خصوصی مجازی گرفته می‌شوند نیز باشند. هنگامیکه این کار انجام می‌شود داده‌های لایه ارتباطی داده‌های کاربردی را مطرح می‌کنند و نتایج به پشته IP برای انتقال واقعی باز می‌گردند. در پایانه دریافتی داده‌ها دوباره به پشته stack

می‌آیند (یکبار برای مسیر یابی و بار دوم برای .(VPN لایه ارتباط می‌تواند ابتدای لایه فیزیکی که متشکل از اجزای شبکه فیزیکی واقعی هستند نیز مرتبط شود. اجزایی مانند هاب‌ها، تکرار کننده‌ها، کابل فیبر نوری، کابل کواکیسال، کارتهای شبکه، کارتهای وفق دهنده host.و ارتباط دهنده‌های شبکه مرتبط : -۴۵ (R ,BNC,...) و مشخصات سطح پایینی برای سیگنالها (سطوح ولتاژ، فرکانسها و(...

لایه فیزیکی (Physical) :

لایه فیزیکی مسئول کد کردن و ارسال داده‌ها روی واسط ارتباطی شبکه‌است و با داده‌ها در فرم بیتهایی که از لایه فیزیکی وسیله ارسال کننده (منبع) هستند و در لایه فیزیکی و دستگاه مقصد دریافت می‌شوند کار می‌کند. اترنت، Token ring، SCSI، هاب‌ها، تکرار کننده‌ها، کابلها و ارتباط دهنده‌ها وسایل اینترنتی استانداردی هستند که روی لایه فیزیکی تابع بندی شده‌اند. لایه فیزیکی همچنین دامنه بسیاری از شبکه سخت‌افزاری مانند LAN، و توپولوژی WAN و تکنولوژی بی سیم (Wireless) را نیز دربرمی گیرد.

تفاوت های بین لایه های OSI و TCP/IP :

سه لایه بالایی در مدل - OSI لایه کاربردی، لایه نمایش و لایه اجلاس معمولاً درون یک لایه در مدل TCP/IPیک جا جمع شده‌اند. درحالیکه بعضی از برنامه‌های کاربردی پروتکل OSI مانند.X ۴۰۰ نیز با همدیگرجمع شده‌اند، نیاز نیست که یک پشته پروتکل TCP/IP برای هماهنگ کردن آنها بالای لایه انتقال باشد. برای مثال پروتکل کاربردی سیستم نایل شبکه (NFS) روی پروتکل نمایش داده خارجی (XDR) اجرا می‌شود و روی یک پروتکل با لایه اجلاس کار می‌کند و فراخوان رویه راه دور (RPC) را صدا می‌زند . RPCمخابرات را به طور مطمئن ذخیره می‌کند، پس می‌تواند با امنیت روی پروتکل UDP اجرا شود. لایه اجلاس تقریباً به پایانه مجازی Telnet که بخشی از متن براساس پروتکل‌هایی مانند پروتکل‌های کاربردی مدل HTTPو TCP/IP SMTP هستند مرتبط می‌شود.و نیز با شمارش پورت UDP و TCP که بخشی از لایه انتقال در مدل TCP/IP است مطرح می‌شود. لایه نمایش شبکه استاندارد MIME است که در HTTP و SMTPنیز استفاده می‌شود.

از آنجایی که سعی برای پیشرفت پروتکل IETF به لایه بندی محض ربطی ندارد، بعضی از پروتکل‌های آن ممکن است برای مدل OSI متناسب باشند. این ناسازگاری‌ها هنگامیکه فقط به مدل اصلیISO ۷۴۹۸، OSI نگاه کنیم بیشتر تکرار می‌شوند، بدون نگاه کردن به ضمایم این مدل (مانند چارچوب مدیریتی(ISOیا سازمان درونی ISO ۸۶۴۸ لایه شبکه (IONL) هنگامیکه IONL و اسناد چهارچوب مدیریتی مطرح می‌شوند،

ICMPوIGMP ، بطور مرتب به عنوان پروتکل‌های مدیریت لایه برای لایه شبکه تعریف می‌شوند. در روشی مشابه، IONLیک ساختمان برای ‘‘قابلیتهای همگرایی وابسته به زیر شبکه” مانند ARP و RARP را فراهم آورده‌است. پروتکلهای IETFمی‌توانند پشت سر هم کاربرد داشته باشند چون توسط تونل زدن پروتکل‌هایی مانند GRE توضیح داده می‌شوند در حالیکه اسنادبیسیک OSIبا تونل زدن ارتباطی ندارند بعضی مفاهیم تونل زدن هنوز هم در توسعه‌های معماری OSI وجود دارند. مخصوصاً دروازه‌های لایه انتقال بدون چهارچوب پروفایل بین‌المللی استاندارد شده‌است. تلاشهای پیشرفت دهنده مرتبط باOSI ، به خاطر استفاده پروتکل‌های TCP/IPدر جهان واقعی رها شده‌اند.. لایه‌ها در ادامه توضیح ازهرلایه در پشته رشته IP آمده‌است.

لایه کاربردی لایه کاربردی بیشتر توسط برنامه‌ها برای ارتباطات شبکه استفاده می‌شود. داده‌ها از برنامه در یک قالب خاص برنامه عبور می‌کنند سپس در یک پروتکل لایه انتقال جاگیری می‌کنند.

از آنجایی که پشته IPبین لایه‌های کاربردی و انتقال هچ لایه دیگری ندارد، لایه کاربردی باید هر پروتکلی را مانند پروتکل لایه اجلاس و نمایش در OSI عمل می‌کنند در بگیرد. داده‌های ارسال شده روی شبکه درون لایه کاربردی هنگامیکه در پروتکل لایه کاربردی جاگیری شدند عبور می‌کنند. از آنجا داده‌ها به سمت لایه‌های پایین تر پروتکل لایه انتقال می‌روند. دو نوع از رایجترین پروتکل‌های لایه پایینی TCP و UDP هستند. سرورهای عمومی پورتهای مخصوصی به اینها دارند (HTTP پورت ۸۰ و FTP پورت ۲۳ را دارند و (...در حالیکه کلاینت‌ها از پورتهای روزانه بی دوام استفاده می‌کنند. روترها و سوئیچها این لایه را بکار نمی‌گیرند اما برنامه‌های کاربردی بین راه در در پهنای باند این کار را می‌کنند، همانطور که پروتکل) RSVP پروتکل ذخیره منابع) انجام می‌دهد.

۳ لایه بالایی در مدل - OSI لایه کاربردی، لایه نمایش و لایه نشست معمولاً درون یک لایه در مدل TCP/IP مجتمع می‌شوند. درحالیکه برخی از برنامه‌های کاربردی پروتکل OSI مانندX ۴۰۰ نیز با یکدیگر جمع شده‌اند، نیاز نیست که یک پشته پروتکل TCP/IP برای یکپارچه کردن آنها بالای لایه انتقال باشد. برای نمونه پروتکل کاربردی سیستم نایل شبکه (NFS) روی پروتکل نمایش داده خارجی (XDR)اجرا می‌شود و روی یک پروتکل با لایه نشست کار می‌کند و فراخوان رویه راه دور (RPC) را صدا می‌زند (Remote Procedure Call).RPCمخابرات را به طور مطمئن ذخیره می‌کند، پس می‌تواند با امنیت روی پروتکل UDP اجرا شود . لایه نشست تقریباً به پایانه مجازی Telnet که بخشی از متن براساس پروتکلهایی مانند پروتکلهای کاربردی مدل HTTP و TCP/IP SMTP هستند مرتبط می‌شود.و نیز با شمارش پورت UDP و TCP که بخشی از لایه انتقال در مدل TCP/IPاست مطرح می‌شود. لایه نمایش شبیه استاندارد MIME که در HTTP و SMTPنیز استفاده می‌شود است. از آنجاییکه تلاش برای پیشرفت پروتکل IETF به لایه بندی محض ربطی

ندارد، برخی از پروتکلهای آن ممکن است برای مدل OSI متناسب باشند. این ناسازگاریها هنگامیکه فقط به مدل اصلیOSI ، ISO۷۴۹۸ نگاه کنیم بیشتر تکرار می‌شوند، بدون نگاه کردن به ضمایم این مدل (مانند چارچوب مدیریتی ISO۷۴۹۸\۴ (یا سازمان درونی ISO ۸۶۴۸ لایه شبکه (IONL) هنگامیکه IONL و مستندات چهارچوب مدیریتی مطرح می‌شوند، ICMPوIGMP ، بطور مرتب به عنوان پروتکلهای مدیریت لایه برای لایه شبکه تعریف می‌شوند. در روشی مشابه، IONLیک ساختمان برای «قابلیتهای همگرایی وابسته به زیر شبکه «مانند ARP و RARP را فراهم آورده‌است. پروتکلهای IETFمی‌توانند پشت سر هم کاربرد داشته باشند چون توسط تونل زدن پروتکلهایی مانند (Generic Routing Encapsulation) GRE شرح داده می‌شوند در حالیکه مستندات پایه‌ای OSI با تونل زدن ارتباطی ندارند برخی مفاهیم تونل زدن هنوز هم در توسعه‌های معماری OSI وجود دارند. مخصوصاً دروازه‌های لایه انتقال بدون چهارچوب پروفایل استاندارد شده بین‌المللی. تلاشهای پیشرفت دهنده مرتبط باOSI ، به خاطر استفاده پروتکلهای TCP/IP در دنیای واقعی رها شده‌اند.

سوکت در پایتون :

تا اینجا اطلاعات مقدماتی در رابطه با سوکت و انواع مدل های شبکه ای پیدا نمودیم حال به سراغ برنامه نویسی در شبکه می رویم . یکی از راه هایی که برنامه نویس می تواند یک ارتباط را در شبکه ایجاد نماید باز نمودن یک سوکت بر روی مقصد مورد نظر می باشد برای این کار در پایتون کلاس socket طراحی گردیده ، برای اینکه بتوانید به سراغ این توابع بروید ابتدا باید مقداری اطلاعات کلی در مورد سوکت در پایتون داشته باشید:
پایتون از دو حوزه ارتباطی در شبکه استفاده می کند که به ترتیب عبارتند از حوزه اینترنت AF_INET و حوزه یونیکس AF_UNIX که به عنوان خانواده آدرس حوزه ها مورد استفاده قرار می گیرند . آدرس های حوزه یونیکس به صورت یک رشته متنی می باشند که به آنها مسیر محلی می گویند ولی در حوزه اینترنت ادرس ها به صورت میزبان و پورت (host , port) می باشند . که میزبان می تواند به صورت یک رشته متنی باشد که نشان دهنده یک آدرس معتبر برای میزبان است همچنین می تواند شامل یک آدرس آی پی که با نقطه از هم جدا شده اند نیز باشد . پورت نیز یک مقدار عددی از 1 تا 65535 می باشد و نشان دهنده درگاهی است که سوکت برای اتصال از آن استفاده می نماید.
از میان انواع سوکتها در پایتون دو نمونه از آنها کاربرد بیشتری نسبت به سایر نمونه ها دارد تا جایی که گاهی از این دو نمونه به عنوان تنها نمونه های سوکت در زبان برنامه نویسی یاد می شود.

سوکت های رشته ای (Stram Socket) :
این سوکت ها از نوع اتصال گرا (Connection Oriented) می باشد که یک نوع ارتباط دوطرفه و قابل اطمینان را با رعایت ترتیب و نظارت بر خطاهای احتمالی ایجاد می نماید . شایان ذکر است که این ارتباط توسط پروتوکل tcp پشتیبانی می شود.

سوکت دیتاگرام (Datagram) :
این سوکت از نوع غیر اتصالی (Connectionless) می باشد که یک نوع ارتباط دوطرفه غیر قابل اطمینان می باشد ، در این ارتباط هیچ تضمینی برای ترتیب و ارسال داده ها به طور کامل وجود ندارد. شایان ذکر است این ارتباط توسط پروتوکل udp پشتیبانی می شود.

نمونه سوکتهای بالا پر کاربردترین انواع سوکت ها در برنامه نویسی شبکه می باشد ولی تنها نمونه های سوکت در پایتون نیستند. برای مثال نوع دیگری از سوکت وجود دارد به نام سوکت raw ، این نوع سوکت ها اطلاعات را به صورت مرتب و قابل اطمینان منتقل می کنند . همچنین این نوع ارتباط به شما امکان ایجاد یک packet خام را می دهد . با استفاده از این امکان شما این اجازه را دارید که packet دلخواه خود را بسازید و ارسال نمائید.

حال زمان آن رسیده که به معرفی کلاس ها و توابع مربوط به کار کردن در پایتون برویم نحوه استفاده از کلاس سوکت در حالت کلی به صورت زیر می باشد :

```
socket(IpVersion, socketType[, protocol])
```

ورودی اول نوع آدرس آی پی را در شبکه مشخص می نماید که برای مثال آی پی ورژن 4 یا 6 باشد ورودی دوم نوع ارتباط در یک ارتباط سوکت را (TCP یا UDP) بودن را مشخص می نماید و ورودی سوم شماره پروتکلی را که شما برای ارتباط خود مد نظر دارید را مشخص می نماید این عدد می تواند 0 یا شماره مورد نظر شما باشد توجه دارید که این مقدار یک مقدار اختیاری است و مقدار پیش فرض آن صفر می باشد .

به مثال زیر توجه نمائید :

```
sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
```

همانطور که مشاهده می نمائید یک سوکت ایجاد نمودیم حال می توانیم با استفاده از متدهای کلاس سوکت اقدام به ارتباط و تبادل اطلاعات نمائیم برای مثال تکه کد زیر یک ارتباط با آدرس hostname و پورت portNumber بر قرار می نماید :

```
sock.connect((hostname, portNumber))
```

در ادامه به توضیح توابع ماژول سوکت خواهیم پرداخت .

socket.create_connection(*address*[, *timeout*])

یک ارتباط به مقدار آدرس (که یک تاپل متشکل از (ip,port)) می باشد برقرار می نماید و یک شیئ از نوع سوکت را بر می گرداند ارسال مقدار متغیر timeout که یک مقدار اختیاری است باعث می شود تا در صورت بر قرار نشدن ارتباط در زمان مشخص شده ارتباط fail شود.

socket.getaddrinfo(*host*, *port*[, *family*[, *socktype*[, *proto*[, *flags*]]]])

این تابع یک لیست با ساختار (family, socktype, proto, canonname, sockaddr) را بر می گرداند که سه آیتم اول به صورت عددی می باشند و مقادیری می باشند که از قبل به ماژول ما ارسال گردیده اند canonname یک رشته می باشد که نشان دهنده نام رسمی آن host می باشد همچنین این مقدار می تواند به صورت عددی از نوع ورژنهای مختلف از آدرس IP نیز باشد. Sockaddr نیز یک تاپل می باشد که اطلاعاتی را در مورد آدرس سوکت دارا می باشد.

socket.getfqdn([*name*])
تمامی اطلاعات لازم برای نام دامنه وارد شده را بر می گرداند ، همچنین اگر متغیر ورودی را به این تابع ارسال ننمائیم مقداری که این تابع بر می گرداند نام دامنه برای localhost می باشد.

socket.gethostbyname(*hostname*)
مقدار hostname را به معادل Ipv4 اش تبدیل می نماید که یک رشته به صورت 100.100.100.100 می باشد همچنین اگر مقدار ورودی این تابع آدرس Ip باشد مقداری که بر می گرداند معادل همان آدر آی پی است. این تابع از Ipv6 پشتیبانی نمی نماید.

socket.gethostbyname_ex(*hostname*)
مقدار *hostname را به متناظر Ipv4 تبدیل می نماید و یک لیست به صورت* (hostname, aliaslist, ipaddrlist) را بر می گرداند.

socket.gethostname()
این تابع نام ماشینی را که host شما بر روی آن قرار دارد را به شما بر می گرداند.

socket.gethostbyaddr(*ip_address*)
این تابع یک لیست با ساختار (hostname, aliaslist, ipaddrlist) را بر می گرداند .

socket.getnameinfo(*sockaddr*, *flags*)
این تابع یک آدرس سوکت را به یک تاپل دو عضوی (host,port) تبدیل می نماید که وابسته به تنظیمات پرچم ها (flags) می باشد.

socket.getprotobyname(*protocolname*)

وظیفه این تابع تبدیل نام یک پروتکل اینترنتی به یک مقدار ثابت مناسب برای ارسال به آرگومانهای تابع socket می باشد . این کار زمانی کاربرد خواهد داشت که سوکت در حالت raw (SOCK_RAW) باز شده باشد . برای حالتهای معمولی اگر پروتوکل مورد نظر را از قلم انداخته باشید این کار به صورت خودکار انجام می پذیرد.

socket.getservbyname(*servicename*[, *protocolname*])
وظیفه این تابع تبدیل نام یک پروتکل یا سرویس اینترنتی را به پورت متناظرش برای آن سرویس خاص می باشد .

socket.getservbyport(*port*[, *protocolname*])
وظیفه این تابع تبدیل شماره یک پورت یا یک پروتکل اینترنتی به نام سرویس متناظرش برای آن سرویس خاص می باشد .

socket.socket([*family*[, *type*[, *proto*]]])
وظیفه این تابع ایجاد یک سوکت جدید می باشد که در ورودی خانواده آدرس سوکت ، نوع و پروتکل مورد استفاده در سوکت را دریافت می نماید. خانواده آدرس ها در سوکت می تواند AF_INET (به صورت پیش فرض) ، AF_INET6 یا AF_UNIX باشد. نوع سوکت نیز می تواند SOCKET_STREAM (به صورت پیش فرض) SOCK_DGRAM یا دیگر ثابتهایی که برای سوکت وجود دارد تعریف شود .

socket.socketpair([*family*[, *type*[, *proto*]]])
وظیفه این تابع ایجاد یک جفت ارتباط با دریافت آدرس و نوع ارتباط و شماره پورت می باشد .

socket.fromfd(*fd*, *family*, *type*[, *proto*])
تکثیر فایل توصیفگر و ساخت یک ارتباط سوکت از روی نتایج حاصل . فامیلی آدرس و نوع سوکت و پروتکل نیز از سوکت گرفته می شود.

socket.ntohl(*x*)
یک نوع integer 32 بیتی را به یک میزبان بایتی سفارشی تبدیل می نماید . در ماشینها جایی که میزبان بایتی سفارشی با شبکه بایتی سفارشی معادل هم باشند یک (no-op) رخ می دهد.

socket.ntohs(*x*)

مانند تابع قبلی است با این تفاوت که یک مقدار 16 integer بیتی را به یک میزبان بایتی سفارشی تبدیل می نماید.

socket.htonl(*x*)
یک نوع 32 integer بیتی را از میزبان به شبکه بایتی سفارشی تبدیل می نماید. در ماشینها جایی که میزبان بایتی سفارشی با شبکه بایتی سفارشی معادل هم باشند یک (no-op) رخ می دهد.

socket.htons(*x*)
یک نوع 16 integer بیتی را از میزبان به شبکه بایتی سفارشی تبدیل می نماید. در ماشینها جایی که میزبان بایتی سفارشی با شبکه بایتی سفارشی معادل هم باشند یک (no-op) رخ می دهد.

socket.inet_aton(*ip_string*)
یک مقدار رشته ای از آی پی ورژن چهار را از فرمت نقطه ای (برای مثال 192.168.0.1) به یک بسته 32 بیتی در فرمت باینری تبدیل می کند. این توانایی زمانی مفید است که برنامه شما بخواهد با یک نرم افزار که از کتابخانه استاندارد C استفاده می نماید و احتیاج دارد که نوع آدرسش struct باشد گفتگو نماید. اگر ساختار آی پی که به این تابع به عنوان آرگومان ارسال می شود نادرست باشد خطای socket.error ایجاد می شود.

socket.inet_ntoa(*packed_ip*)
یک بسته 32 بیتی از آی پی ورژن چهار را در فرمت باینری به یک مقدار رشته ای در فرمت نقطه ای (برای مثال 192.168.0.1) تبدیل می نماید. این توانایی زمانی مفید است که برنامه شما بخواهد با یک نرم افزار که از کتابخانه استاندارد C استفاده می نماید و احتیاج دارد که نوع آدرسش struct باشد گفتگو نماید.

socket.inet_pton(*address_family, ip_string*)
یک آدرس آی پی را از فرمت آدرس فامیلی خاص خود به یک بسته در فرمت باینری تبدیل می نماید. این توانایی زمانی مفید است که یک کتابخانه یا پروتکل شبکه برای یک شیئ از نوع struct صدا زده شود. آدرس های فامیلی که در حال حاضر برای این کار پشتیبانی می شوند AF_INET و AF_INET6 می باشند. اگر ساختار آی پی که به این تابع به عنوان آرگومان ارسال می شود نادرست باشد خطای socket.error ایجاد می شود .

socket.inet_ntop(*address_family, packed_ip*)

یک بسته از آی پی آدرس را در فرمت باینری به استاندارد یک فامیلی آدرس تبدیل می نماید . این توانایی زمانی مفید است که یک کتابخانه یا پروتکل شبکه برای یک شیئ از نوع struct صدا زده شود. آدرس های فامیلی که در حال حاضر برای این کار پشتیبانی می شوند AF_INET و AF_INET6 می باشند. اگر ساختار آی پی که به این تابع به عنوان آرگومان ارسال می شود نادرست باشد خطای socket.error ایجاد می شود .

socket.getdefaulttimeout()
مقدار پیش فرض timeout را برای یک سوکت جدید بر می گرداند. اگر مقدار برگردانده شده None باشد این بدان معناست که برای این سوکت هیچ timeout مقدار دهی نشده است. زمانی که ماژول سوکت برای اولین بار ایجاد می شود این مقدار برابر None می باشد.

socket.setdefaulttimeout(*timeout*)
مقدار timeout را برای یک سوکت جدید مقدار دهی می نماید .

socket.SocketType
این یک نوع در پایتون می باشد که نوع شیئ سوکت را مشخص می نماید

شیئ سوکت (Socket) :
شیئ سوکت دارای متدهای زیر می باشند که در ادامه به معرفی آنها خواهیم پرداخت .

socket.accept()
وظیفه این متد پذیرفتن یک ارتباط در شبکه می باشد . سوکت در مرز یک آدرس گوش به زنگ می ماند تا یک ارتباط در شبکه بر قرار شود .

socket.bind(*address*)
وظیفه این متد مرتبط کردن یک سوکت با یک آدرس می باشد . توجه داشته باشید که سوکت از قبل نباید مرز بندی شده باشد . به مفهوم ساده تر سوکت برای اینکه قابل شناسایی باشد احتیاج دارد تا یک نام مخصوص خود داشته باشد که این تابع این کار را برای سوکت انجام می دهد. (ساختار آدرس بستگی به خانواده آدرسی دارد که شما برای سوکت خود معرفی نموده اید).

socket.close()
وظیفه این متد قطع کامل تمامی ارتباطاتی است که یک سوکت ایجاد نموده است . همچنین سوکت ها در زباله روبی (garbage collected) به صورت اتوماتیک بسته می شوند.

socket.connect(*address*)
یک ارتباط از طریق سوکت را به آدرس مورد نظر ایجاد می نماید.(ساختار آدرس بستگی به خانواده آدرسی دارد که شما برای سوکت خود معرفی نموده اید).

socket.connect_ex(*address*)
مانند تابع قبل یک ارتباط از طریق سوکت ایجاد می نماید با این تفاوت که در هنگام بروز خطاهایی که تابع connect() در سطح-سی (C-Level) بر می گرداند این متد به جای اینکه یک خطا را ایجاد نماید یک شاخص خطا را بر می گرداند .

socket.fileno()
این متد فایل مفسر سوکت را بر میگرداند (که به صورت یک مقدار عددی کوچک می باشد).

socket.getpeername()
این متد آدرس های سوکتی را که با آن در ارتباط هستند را بر می گرداند.کاربرد این متد برای پیدا کردن شماره پورت هایی است که سوکت به آنها در ارتباط می باشد . توجه داشته باشید که بعضی از سیستم ها از این تابع پشتیبانی نمی کنند.

socket.getsockname()
این متد آدرس یک سوکت را بر می گرداند . این توانایی مفید است که شما به شماره پورت یک سوکت نیاز دارید با استفاده از این سوکت می توانید به این مقدار دسترسی پیدا نمائید.

socket.listen(*backlog*)
وظیفه این متد منتظر ماندن برای گوش سپردن به ایجاد یک ارتباط یا ارسال اطلاعات در یک ارتباط می باشد . آرگومان backlog طول صفی را که این سوکت می تواند با آنها ارتباط بر قرار کند را مشخص می نماید که حداقل 1 و حداکثر آن بستگی به نیاز سیستم دارد (معمولا 5 می باشد).

socket.makefile([*mode*[, *bufsize*]])
این متد یک شیئ از فایلی که به سوکت وابسته شده را بر می گرداند.

socket.recv(*bufsize*[, *flags*])
وظیفه این متد دریافت اطلاعات از یک سوکت می باشد . مقداری که این تابع بر می گرداند یک مقدار از نوع string می باشد که شامل اطلاعات دریافت شده می باشد . آرگومان bufersize حداکثر سایز اطلاعاتی که در یک ارتباط قابل دریافت می باشد را مشخص می نماید.

socket.recvfrom(*bufsize*[, *flags*])
وظیفه این متد مانند تابع قبل می باشد با این تفاوت که مقداری که توسط این متد برگردانده می شود، یک جفت اطلاعات می باشد که به صورت (string , address) می باشد که مقدار آرگومان اول مقداری است که توسط سوکت دریافت شده و مقدار آرگومان دوم آدرس سوکتی است که اطلاعات را ارسال نموده است.

socket.recvfrom_into(*buffer*[, *nbytes*[, *flags*]])
وظیفه این متد دریافت اطلاعات از یک سوکت می باشد با این تفاوت نسبت به متدهای قبل که به جای اینکه یک مقدار رشته ای جدید تولید کند و اطلاعات را در آن بریزد اطلاعات را مستقیما در بافر ذخیره می نماید. خروجی این متد یک جفت به صورت (nbytes , address) می باشد که آرگومان اول تعداد بایت های اطلاعات دریافت شده را نمایش می دهد و آرگومان دوم آدرس سوکتی است که اطلاعات را ارسال نموده است.

socket.recv_into(*buffer*[, *nbytes*[, *flags*]])
وظیفه این متد دریافت اطلاعات در یک سایز مشخص از یک سوکت می باشد.همچنین اطلاعات دریافت شده را در بافر ذخیره می نماید که سریعتر از ذخیره آن ها در یک رشته می باشد . اگر مقدار nbytes مشخص نشده باشد یا برابر صفر باشد دریافت ازلاعات به اندازه سایز بافری که Available می باشد انجام می پذیرد.

socket.send(*string*[, *flags*])
وظیفه این متد ارسال اطلاعات به یک متد دیگر می باشد .سوکت مورد نظر باید با سوکتی که می خواهد اطلاعات را برایش ارسال نماید در ارتباط باشد . خروجی این متد تعداد بایتهایی می باشد که آنها را ارسال نموده است. این تابع هیچ تعهدی در قبال ارسال تمامی اطلاعات ندارد و برنامه خودش باید این وظیفه را به عهده بگیرد.

socket.sendall(*string*[, *flags*])

وظیفه این متد ارسال اطلاعات به یک متد دیگر می باشد . سوکت مورد نظر باید با سوکتی که می خواهد اطلاعات را برایش ارسال نماید در ارتباط باشد . برخلاف متد قبلی این متد وظیفه ارسال تمامی اطلاعات را به عهده می گیرد تا زمانی که خطایی رخ دهد . در صورتی که همه چیز با موفقیت انجام شود این متد هیچ مقداری را بر نمی گرداند و در زمانی که خطایی رخ دهد خروجی این متد معادل خطایی است که به وقوع پیوسته است. شایان ذکر است زمانی که خطایی رخ می دهد روند ارسال اطلاعات متوقف شده و اینکه چه مقدار از اطلاعات با موفقیت ارسال شده قابل تشخیص نمی باشد.

socket.sendto(*string*[, *flags*], *address*)

وظیفه این متد ارسال اطلاعات به یک متد دیگر است با این فرق که لزوما نباید با سوکتی که می خواهد اطلاعات را برایش ارسال نماید در ارتباط باشد . تنها کافیست آدرس سوکت مقصد را برایش مشخص نمائید تا اطلاعات را برای سوکت مورد نظر ارسال نماید.

socket.setblocking(*flag*)

وظیفه این متد مسدود کردن یا رفع مسدودیت از یک سوکت می باشد . اگر مقدار پرچم برابر صفر باشد سوکت رفع مسدودست می شود در غیر اینصورت اگر پرچم برابر یک شود سوکت مسدود می شود . تمام سوکت ها در ابتدا در وضعیت مسدود بودن به سر می برند. اگر برنامه نویس تابع recv() را زمانی صدا نماید که سوکت در وضعیت رفع مسدودیت به سر می بد این متد هیچ مقداری را دریافت نخواهد نمود ، همچنین اگر در این وضعیت تابع send() نیز صدا زده شود این تابع نمی تواند اطلاعات را به صورت منظم دسته بندی نماید و خطا رخ می دهد. در وضعیت مسدود بودن فراخوانی سوکت مسدود مس شود تا متد در خال اجرا کارش را به پایان برساند.

socket.settimeout(*value*)

وظیفه این متد تنظیم یک timeout برای مسدود بودن یک سوکت در حین انجام یک عملیات می باشد. مقدار آرگومان value یک عدد اعشاری غیر منفی می باشد همچنین برنامه نویس می تواند مقدار آن را با none نیز مقدار دهی نماید. اگر قبل از اینکه عملیات سوکت به پایان برسد این زمان تمام شود سوکت اقدام به ایجاد خطای timeout می نماید . همچنین اگر مقدار آرگومان value را با مقدار null مقدار دهی نمائید بدان معناست که هیچ محدودیت زمانی وجود ندارد همچنین اگر مقدار این آرگومان را با صفر مقدار دهی نمایند بدان معناست که سوکت دروضعیت رفع مسدودست قرار دارد.

socket.gettimeout()

مقداری که این متد بر می گرداند مدت زمانی است که به عنوان timeout برای یک سوکت خاص ثبت گردیده است . که یک مقدار اعشازی می باشد.همچنین اگر مقداری ثبت نشده باشد مقدار None برگردانده می شود.

socket.shutdown(*how*)

وظیفه این متد خاتمه دادن به یک یا دو طف یک ارتباط می باشد . اگر مقدار آرگومان how برابر با SHUT_RD باشد دریافت اطلاعات غیرفعال می شود ، اگر مقدار آرگومان how برابر SHUT_WR باشد ارسال اطلاعات غیر فعال می شود و اگر مقدار آرگومان how برابر با SHUT_RDWR باشد هم دریافت و هم ارسال اطلاعا هر دو غیر فعال می شوند.

حال که متدهای سوکت را به صورت کامل توضیح دادیم در ادامه چند مثال عملی را بررسی می نمائیم :

ابتدا قسمت Server برنامه را می نویسیم

```
# Echo server program
import socket

HOST = 'Server'                 # Symbolic name meaning all
available interfaces
PORT = 50007                    # Arbitrary non-privileged
port
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected by', addr
while 1:
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
conn.close()
```

حال به قسمت Client می پردازیم

```
# Echo client program
import socket

HOST = 'ServerAddress'    # The remote host
PORT = 50007                 # The same port as used by
the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', repr(data)
```

همانطور که مشاهده می نمائید ابتدا یک برنامه به عنوان سوریس دهنده ساختیم که که بر روی پورت شماره 50007 به صورت گوش به زنگ برای ایجاد یک ارتباط باقی می ماند زمانی که ارتباطی بر قرار می شود نام ارتباط گیرنده را نمایش می دهد همچنین اطلاعاتی را که از طریق سر.یس گیرنده برایش ارسال می گردد را در یک متغیر ذخیره و بعد از پایان ارسال آن را دوباره برای سرویس گیرنده ارسال می نماید
قسمت دوم کد مربوط به سرویس گیرنده می باشد در این قسمت یک اتصال با سرور ایجاد می شود و اقدام به ارسال اطلاعات برای آن می نماید (توجه داشته باشید که مقدار متغیر Host باید آدرس برنامه سرویس دهنده شما باشد) همچنین مقداری را که از طرف سرویس دهنده به عنوان پاسخ در یافت می کند را نیز چاپ می نماید.

همانطور که مشاهده نمودید برای یک ارتباط ساده تنها از تعداد محدودی از توابع و متدهای نام برده شده در بالا استفاده می شود حال یک مثال پیچیده تر را بررسی می نمائیم :

```
import socket

# the public network interface
HOST = socket.gethostbyname(socket.gethostname())
```

```
# create a raw socket and bind it to the public
interface
s = socket.socket(socket.AF_INET, socket.SOCK_RAW,
socket.IPPROTO_IP)
s.bind((HOST, 0))

# Include IP headers
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

# receive all packages
s.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

# receive a package
print s.recvfrom(65565)

# disabled promiscuous mode
s.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
```

در مثال بالا یک sniffer بسیار ساده با استفاده از سوکت raw را در سیستم عامل ویندوز طراحی نموده ایم .
تنها توجه داشته باشید برای اینکه اسکریپت بالا کار کند احتیاج دارید تا در وضعیت مدیر اصلی باشید.

برنامه نویسی در اینترنت با پایتون :

یکی دیگر از توانائیهای پایتون برنامه نویسی در محیط اینترنت و کارکردن با پروتکلهای معروف اینترنت از قبیل FTP ، HTTP ، POP3 و ... می باشد درادامه با توانائیهای پایتون در این زمینه بیشتر آشنا می شویم .

پروتکل FTP :

FTP که مخفف (File Transfer Protocol) می باشد به معنای قرارداد انتقال پرونده می باشد که در شبکه های رایانه ای برای انتقال پرونده ها درون شبکه مورد استفاده قرار می گیرد .

درمیان رایانه‌های میزبان، FTP به طور ویژه یک قراردادِ متداول برای دادوستد فرمان‌ها و پرونده‌ها در هر شبکه پشتیبان از قرارداد اینترنت و قرارداد هدایت انتقال (TCP/IP) (مانند اینترنت و اینترانت) است .درگاه (Port) پیش‌فرض برای خدمات FTP، درگاه 21 و برای انتقال داده از درگاه 20 استفاده می‌شود.

در یک انتقال FTP دو رایانه دخیل است، یک کارساز (Server) و یک کاربر(Client) . برنامه‌های کارساز FTP را اجرا می‌کند، و درخواست پذیرش در شبکه را با رایانهٔ دیگر (یعنی کاربر) مطرح می‌کند. رایانهٔ کاربر برنامه‌های کاربری FTP را اجرا و یک ارتباط با کارساز بر قرار می نماید .

هنگامی که یک ارتباط برقرار می‌شود کاربر می‌تواند تعدادی از برنامه‌ها را تغییر دهد (دستکاری محدود)، مانند بارگذاری پرونده در کارساز و بارگیری پرونده از آن، یا بازنامیدن یا حذف پرونده‌ها در کارساز و ...

در واقع همهٔ بسترهای رایانه‌ای از FTP پشتیبانی می‌کنند و به هر ارتباط رایانه‌ای که بر اساس قرارداد هدایت انتقال/قرارداد اینترنت باشد صرف‌نظر از این که از چه سامانهٔ عاملی استفاده می‌شود، اگر رایانه‌ها اجازهٔ دسترسی به FTP را داشته باشند، این اجازه را می‌دهد که در پرونده‌های رایانهٔ دیگر در این شبکه تغییراتی ایجاد نماید.

برای اینکه یک کاربر بتواند از این پروتکل استفاده نماید ابتدا باید به یک کارساز ارتباط برقرار نماید و سپس با استفاده از نام کاربری و کلمه عبور به آن سامانه وارد شود ، البته بدون با توجه به تنظیمات کار ساز این امکان نیز وجود دارد که بدون وارد شدن به سامانه کار ساز نیز بتوان از آن پرونده دریافت نمود. در زبان پایتون تمام امکاناتی که یک کاربر برای استفاده از یک سامانه احتیاج دارد طراحی و تعبیه گردیده است که در ادامه با آنها بیشتر آشنا خواهید شد.

برای استفاده از این پروتکل در پایتون ماژول ftplib طارحی گردیده که تمامی نیازهای برنامه نویس را برای استفاده از این پروتکل فراهم نموده در حالت کلی یک ارتباط با پروتکل FTP در زبان پایتون به صورت زیر می باشد :

```
from ftplib import FTP
f=FTP('ftp.mozilla.org')
print f.login('anonymous', 'guess@who.org')
f.retrlines('LIST')

-230
-230ftp.mozilla.org / archive.mozilla.org - files are in
/pub/mozilla.org
-230
-230Notice: This server is the only place to obtain
nightly builds and needs to
-230remain available to developers and testers. High
bandwidth servers that
-230contain the public release files are available at
ftp://releases.mozilla.org/
-230If you need to link to a public release, please link
to the release server,
```

```
-230not here. Thanks!
-230
-230Attempts to download high traffic release files from
this server will get a
 550" -230Permission denied." response.
 230Login successful.
-rw-r--r--    1 ftp      ftp             528 Nov 01  2007
README
-rw-r--r--    1 ftp      ftp             560 Sep 28  2007
index.html
drwxr-xr-x   35 ftp      ftp            4096 Nov 10 17:07
pub
```

همانطور که در مثال قبل مشاهده نمودید کار کردن با این کتابخانه بسیار راحت است به راحتی به یک کارساز وصل شدیم و لیست دایرکتوری های موجود در کارساز را دریافت نمودیم حال که یک دید کلی پیدا نمودید به معرفی توابع و ماژول های پایتون خواهیم پرداخت

class ftplib.FTP([host[, user[, passwd[, acct[, timeout]]]]])

یک نمونه از کلاس Ftp را بر می گرداند زمانی که آرگومان host را ارسال نموده باشید متد connect(host) به ساخته می شود. زمانی که آرگومان user ارسال شود متد login(user,passwd,acct) ساخته می شود.

class ftplib.FTP_TLS([host[, user[, passwd[, acct[, keyfile[, certfile[, timeout]]]]]]])

یک زیر کلاس برای کلاس Ftp که از Tls پشتیبانی می نماید در RFC 4217 به صورت کامل شرح داده شده است. به صورت عادی با درگاه شماره 21 ارتباط برقرار می نماید . برای اینکه مد امنیتی برای این ارتباط برقرار شود باید متد prot_p فراخوانی شود.