جلسه سیزدهم

ارتباط مبتنی بر جویبار داده‌ای (Stream Oriented Communication)

آخرین مبحث مربوط به ارتباط بین فرآیندها در یک سیستم توزیع شده، ارتباط مبتنی بر جویبار داده‌ای خواهد بود. در این نوع ارتباط دو نوع رسانه با نام‌های رسانه های پیوسته media (Continues) و دیگری رسانه های گسسته (Discrete media) مدنظر قرار می گیرد .

در رسانه های پیوسته آنچه حائز اهمیت است این است که باید بین داده ای که در حال انتقال است ارتباط زمانی مؤثری وجود داشته باشد، یعنی فاصله زمانی داده های ارسال شده از اصول این نوع ارتباط است. از جمله این نوع داده ها می توان به ویدئو و صوت اشاره کرد. اما در Discrete media ها یا مدیاهای گسسته در واقع این ارتباط زمانی به آن معنی وجود ندارد و داده ها می توانند مستقل از هم ارسال شوند. از جمله این نوع داده ها می توان به متن ها و تصاویر ثابت اشاره کرد. در ارتباطات مبتنی بر جویبار داده ای، یک جویبار داده ای به عنوان یک ترتیب از واحدهای داده ای خواهد بود که در مدهای مختلف انتقال، می توانند منتقل شوند. اولین نوع انتقال یک Data stream ، مد انتقالی آسنکرون یا غیرهمگام است. در این نوع انتقال آیتم های داده ای در یک جویبار، یکی پس از دیگری منتقل می‌شوند. داده ها، داده‌های Discrete فرض می شوند و در این نوع انتقال هیچ محدودیت زمانی و هیچ قید زمانی وجود ندارد.

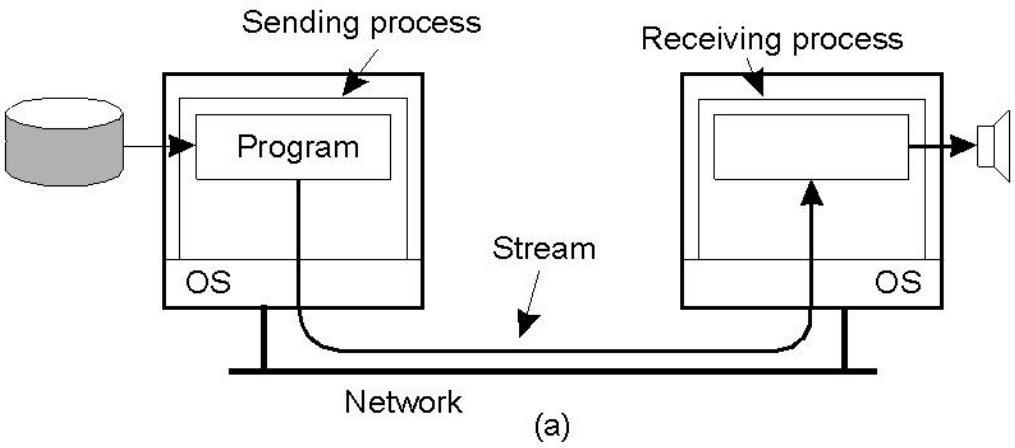
از جمله این نوع انتقال‌ها، می توان به انتقال یک فایل یا file transfer اشاره کرد. نوع دوم stream های داده‌ای و نوع انتقال آنها، انتقال همگام یا Synchronous است. در مد انتقال سنکرون یا همگام تنها نکته ای که بسیار حائز اهمیت است محدودیت زمانی است، که برای تأخیر هر واحد داده ای در جویبار داده ای در نظر گرفته می شود و یک end to end delay ماکزیمم یا یک تأخیر از انتها تا انتهای بسته ی داده ای در حد ماکزیمم این تأخیر در این سیستم تعریف می شود و هیچ گاه نباید نحوه انتقال و زمان بندی انتقال داده ها به گونه ای باشد که از این حد ماکزیمم تأخیر عبور نماید. نوع دیگری که به عنوان مدلهای انتقال در سیستم های Stream Oriented مورد توجه قرار می‌گیرد مد انتقال Isochronous است. در این نوع که آن را به عنوان نرخ انتقال متقارن یا متوازن می شناسیم لازم است که داده ها کاملاً on time و

بدون تأخیر ارسال شوند و یا برای تأخیر ارسال داده ها یک حد مینیمم و یک حد ماکسیمم در نظر گرفته شود. به عبارت دیگر باید بطور حتم یک jitter bounded یا یک jitter محدود شده رعایت شود.

از دیدگاه دیگر جویبارهای داده ای را به دو گروه می توان تقسیم نمود:

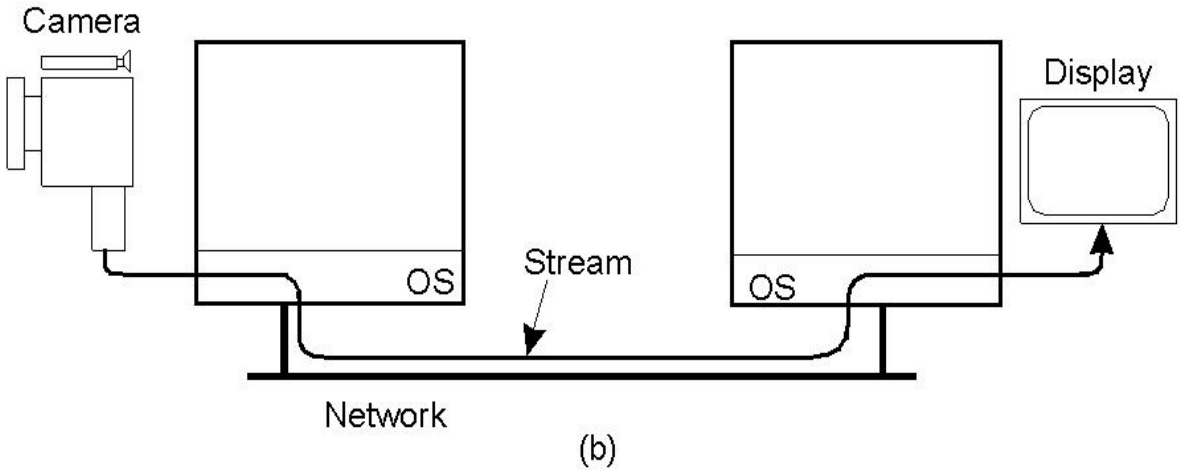
- Simple stream یا جویبارهای ساده که تنها یک sequence ساده از داده ها هستند یعنی یک ترتیب ساده از داده ها بدون اینکه سربار داده ای خاصی بر روی آن قرار گرفته باشد.
- Complex Stream تعدادی از Simple stream های مرتبط به هم هستند.

اگر به ازای هر Stream یک Substream در نظر بگیرید، یعنی به ازای هر جویبار یک زیر جویبار داده در نظر گرفته شود که هر یک از این زیر جویبارها خود یک جویبار داده ای هستند آنگاه ما یک Complex stream خواهیم داشت. به عنوان مثالی از این نوع stream ها می توان به انتقال Movieها یا فیلم ها اشاره کرد که با توجه به مطالب پیشین، در ساختار یک Movie معمولاً چندین نوع داده از جمله ویدئو، متن و صدا وجود داشت که می‌توان هر کدام را به عنوان یک Substream شناخت. اما نکته ای که در اینجا حائز اهمیت است نحوه برقراری stream بین دو فرآیند یا به عبارتی مفهوم stream در سطح یک شبکه توزیع شده است. همانطور که قبلاً عنوان شد stream یک ارتباط مجازی (virtual connection) بین یک source و sink است. Source در اینجا ماشینی است که داده‌ای را تولید می کند و sink ماشینی است که آن داده را دریافت می کند. در این حالت که ساده ترین حالت ایجاد stream است، یک stream بین دو فرآیند در سطح یک شبکه ایجاد می‌شود. یعنی یک فرآیند به عنوان فرآیند ارسال کننده داده و دیگری به عنوان فرآیند دریافت کننده داده شناخته شده است.



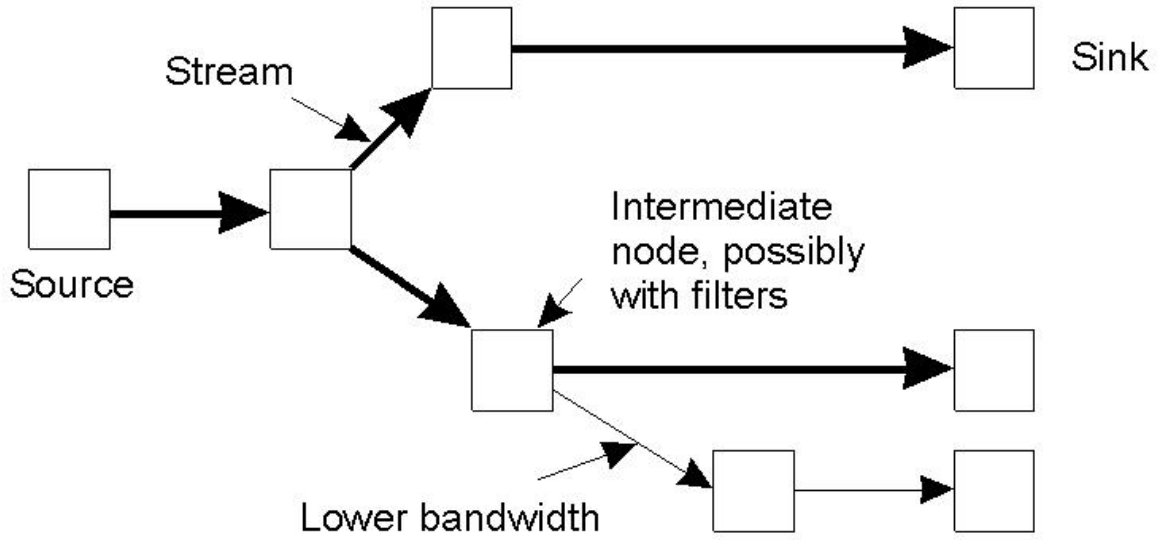
ارتباط مبتنی بر Stream

حالت دیگری که برای streamها می‌توان در نظر گرفت و مزایایی نیز به همراه دارد، برقراری یک stream مستقیم بین دو Device است. با توجه به شکل، یک دوربین وظیفه تصویربرداری را بر عهده دارد و مستقیماً داده تصویر برداری شده را از طریق یک stream بر بستر شبکه به یک Display که در قسمت دیگری از شبکه است ارسال می کند.



بدیهی است، چنانچه چنین ارتباط مستقیمی بین دو Device برقرار شود برخی سربارهای محاسباتی که در حالت قبل وجود داشت، حذف خواهد شد. اما بهینه سازی بر روی داده ها نیز قابل انجام نخواهد بود.

حالت دیگر، برقراری stream به صورت Multicast است. در حالت Multicast یک Source داده‌ها را به سمت چندین sink ارسال می‌کند. در شکل نمونه ای از این stream را می بینید. با توجه به شکل، stream ارسالی از سمت Source در ندهایی به چندین stream مجزا تشکیل شده و به سمت Sink ارسال می شود.

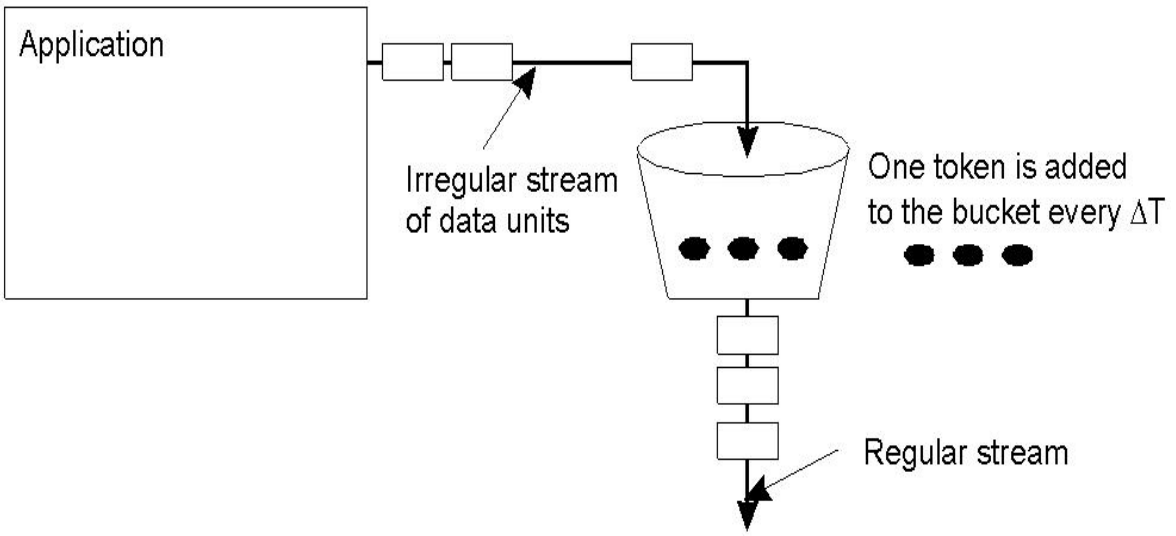


در واقع نکته ای که در اینجا وجود دارد و در شکل نیز نشان داده شده است این است که استفاده از این روش به کاربر یا به مدیر شبکه این امکان را می دهد که در ندهای میانی که برای شکستن stream به چندین stream تعبیه شده است، فیلترهایی نیز در نظر گرفته شود. از جمله کاربرد این فیلترها این است که چنانچه یک Sink تحت یک پهنای باند ضعیف قرار گرفته است و امکان دریافت داده های حجم بالا را ندارد، ندهای میانی می‌توانند با فیلتر کردن و احیاناً انجام اعمالی مانند data Compression حجم داده را کاهش داده و بنابراین در این مسیر، داده را با حجم بالا و در مسیر دیگر داده را با حجم پایین‌تر ارسال کند. قطعاً کیفیت داده ها در این دو مسیر یکسان نخواهد بود. اما نکته دیگری که در مبحث مربوط به ارتباط مبتنی بر stream مطرح می شود بحث تعیین Quality of servies ها است. مبحث مهم در of Quality servies این است که بتوان یک توصیف دقیق از جریان داده شامل پهنای باند مورد نیاز، نرخ انتقال، تأخیرها و مواردی مانند آن را تعیین نمود و آنگاه سیستمی که مدیریت stream را انجام می‌دهد، باید بتواند این تعریف و specification دقیق را به نوعی رعایت کند. یکی از ابزارهایی که برای رعایت

Qualityof servies مورد نظر قرار می گیرد استفاده از مدل Partridge است. در این مدل stream ها در قالب باکت هایی از tokenها یا عبارت های کوچک مدل می شوند. به عبارت دیگر از الگوریتمی که آن را token bucket می‌نامیم، استفاده می‌شود.

token معمولاً به عنوان یک واحد کوچک انتقال اطلاعات نامگذاری می‌شود، و bucket به عنوان یک مجموعه یا یک بسته از داده هاست. در روش token bucket در واقع آنچه که حائز اهمیت خواهد بود، این است که چطور stream مدیریت شود به شکلی که بتوان با توجه به داده های قرار گرفته در استریم ترافیک شبکه مدیریت شود؟ ایده اصلی بر این اساس است که معمولاً tokenها در سمت دریافت کننده بهتر است که در یک فاصله زمانی ثابت دریافت شوند. در این تکنیک اتفاقی که خواهد افتاد این خواهد بود که tokenهایی که در سمت فرستنده ارسال می شوند و ممکن است که فاصله زمانی ایجاد آنها فاصله زمانی مشخصی نباشد در داخل یک بافر ذخیره می شوند و سپس با پر شدن این بافر که یک بسته (bucket) است، در فاصله های زمانی منظم سعی خواهد شد که tokenها به سمت مقصد حرکت کنند.

با توجه به شکل زیرمی‌بینید، چگونه الگوریتم token bucket مدیریت بسته ها را انجام می دهد. یعنی Application واحدهای داده ای را به صورت نامنظم تولید می کند سپس این داده ها در بافری قرار می گیرند و پس از پر شدن بافر داده‌ها ارسال می شوند البته برای اینکه بتوانیم یک regular stream یعنی یک Stream منظم داشته باشیم.



لزوماً پر شدن bucket نیز مهم نخواهد بود. یعنی ممکن است که پیش از پر شدن bucket برای اینکه فاصله زمانی منظم رعایت شود. بسته ها به سمت مقصد ارسال شود. نکته ای که در اینجا مهم است این است که چنانچه bucket پیش از آنکه فاصله زمانی منظم برسد پر شود چه اتفاقی برای tokenهایی که بعد از آن می رسند خواهد افتاد؟

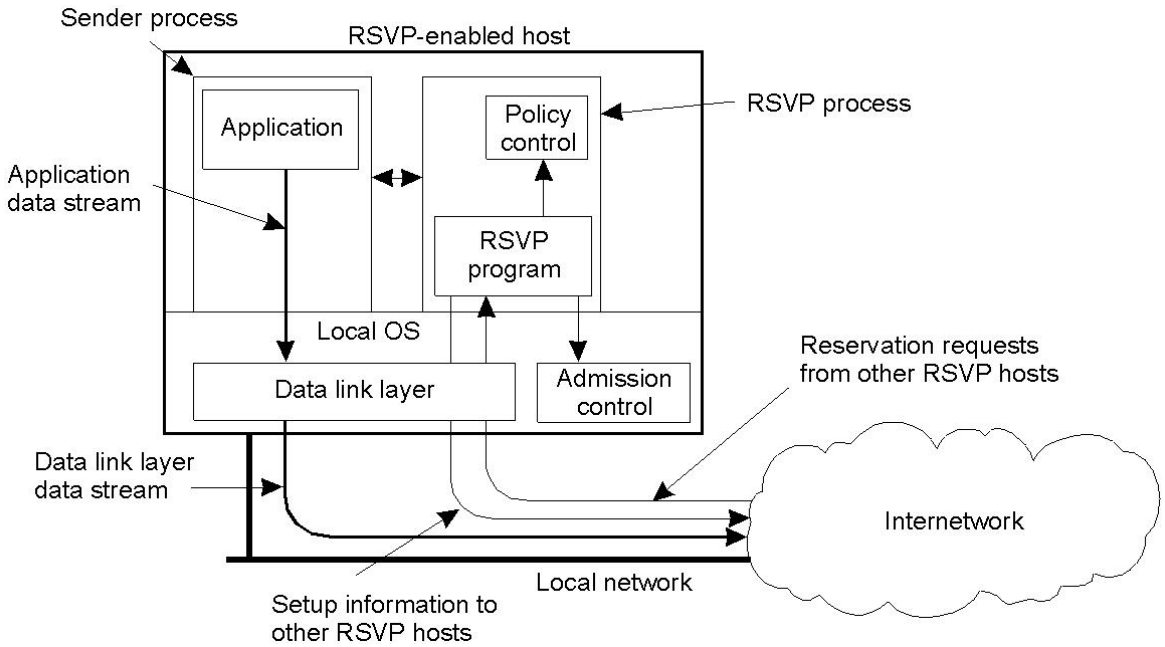
در اینجا ساده ترین تکنیکی که استفاده می شود این است که tokenها ، drop می‌شوند. بنابراین استفاده از این ساختار باعث می شود که بتوانیم داده ها را با یک زمان نسبتاً منظم به سمت مقصد حرکت دهیم. Tokenها در اینجا معمولاً یک تعداد ثابتی از بایت ها هستند که Application می تواند به سمت شبکه ارسال کند. همانطور که عنوان شد، هدف نهایی این الگوریتم این است که یک Stream منظم داشته باشیم. از دیدگاه دیگری نیز می توان بحث Quality of servies را مورد بررسی قرار داد. در واقع آنچه که در اینجا می توان بررسی کرد نیازمندی ها یا الزامات سرویس هاست. در واقع سرویس‌هایی که در سطح یک Stream باید ارائه شوند باید نکات زیر را رعایت کنند. اول اینکه باید Loss sensitivity و loss interval را رعایت کنند. در اینجا Loss به معنای از دست دادن داده ها است. یعنی باید سیستم به گونه ای باشد که در مقابل از دست دادن داده ها تا حد خاصی حساسیت داشته باشد و در واقع به نوعی اگر قرار است داده ای در طول ارسال داده ها و در حین برقراری Stream از بین برود آنگاه بازه گم شدن داده ها و از بین رفتن آنها باید یک بازه توزیع شده باشد. یعنی نباید در یک فاصله زمانی کوتاه تعداد زیادی داده از

بین برود و سپس در یک بازه زمانی بزرگ اتفاق خاصی نیفتد. معمولاً چنین سیستمی مورد قبول کاربر قرار نمی گیرد و معمولاً برای رعایت Quality of servies در تعریف specification یا توصیف مربوط به یک Stream Oriented Communication باید یک نرخ از دست دادن داده های قابل قبول تعریف شود و سیستم به گونه‌ای داده ها را مدیریت کند که هیچ گاه از این نرخ ماکزیمم میزان از دست رفتن داده ها بیشتر نباشد. مسأله دیگر مبحث مربوط به Burst loss یا گم شدن داده های پشت سر هم می‌باشد. در این مبحث نیز سیستم باید به نوعی گارانتی نماید که چند داده پشت سر هم مجاز هستند که از بین بروند. بحث دیگر، بحث حداکثر تأخیر است. حداکثر تأخیر یعنی بیشترین میزان جیتری که اجازه داده می‌شود در سیستم ایجاد شود. این مطالب از جمله مواردی بودند که باید به عنوان specification سیستم تعیین شوند و در واقع سرویس هایی که ارائه می شود با توجه به این مفاهیم مورد نظر قرارگیرد.

یکی از روشهایی که مبحث فوق را مدیریت می کند و می‌تواند مورد استفاده قرار گیرد برای اینکه بتوان سرویس های مناسبی را در یک ارتباط مبتنی بر stream ارسال کرد، روش RSVP(Resource reservation Protocol) است. این پروتکل، یک پروتکل لایه Transport است و اتفاقی که در آن می افتد این است که فرستنده باید specification یا توصیف جریان را آماده کند و در این پروتکل یک RSVP Process یا یک فرآیند Process وجود دارد که collocated است یعنی همراه با sender در یک مکان قرار گرفته است، و specification توصیف شده یا specification تهیه شده توسط Sender را به صورت محلی ذخیره می‌کند. پروتکل یک پروتکلی است که توسط دریافت کننده آغاز می شود و Receiver و دریافت کننده کاری که انجام می دهد این است که سعی می کند یک مسیری را در طول مسیر ارتباطی از دریافت کننده به فرستنده رزرو کند. فرآیند مربوط به RSVP کاری که انجام می دهد این است که از رزرو کردن یا ذخیره کردن منابعی بیش از آنچه که مورد نیاز است جلوگیری کند. در این راستا فرستنده یک مسیر را تنظیم می کند و سپس specification مورد نظر را در ند های میانی مسیر قرار می دهد. پس از آن درخواست رزرو یا درخواست نگهداری منابع به سمت Policy control یا در واقع ابزار Policy control فرستاده می شود. Policy control کاری که انجام می دهد این است که سطح دسترسی Receiver یا دریافت کننده را چک می کند تا اینکه بتواند با توجه به اینکه سطح دسترسی

Receiver در چه سطحی است، انجام عمل رزرو را ممکن سازد. بدیهی است که برخی از Receiver ها ممکن است سطح دسترسی داشته باشند که این امکان را برای آنها فراهم نمی آورد که بتوانند هر منبعی را رزرو کنند. قسمت دیگری که درخواست reservation به آن ارسال می‌شود جدا از Policy control بحث Admission control یا قسمت کنترل ورودی است. Admission control با توجه به درخواست‌های reservation و با توجه به Policy control اگر درخواست توسط Policy control کنترل شده باشد و تأیید شده باشد، سعی خواهد کرد که بررسی کند آیا منابع کافی وجود دارد که این درخواست پذیرفته شود؟ و پس از آن با توجه به میزان منابع، ورودی فرآیندها به سیستم را کنترل خواهد کرد.

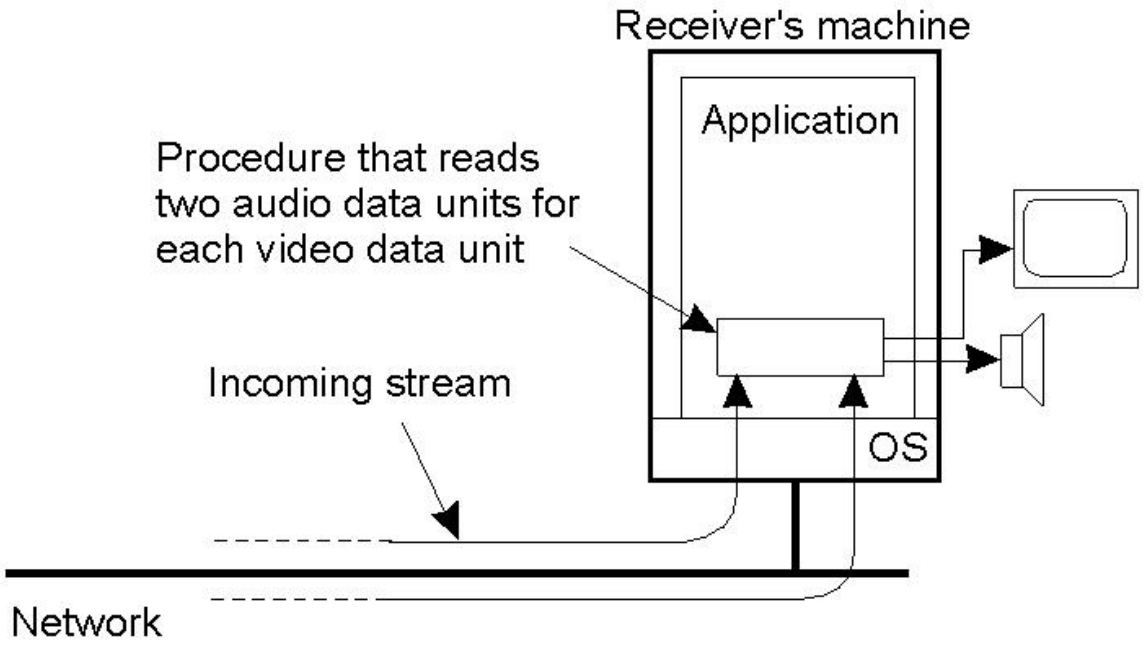
ارتباط مبتنی بر استریم



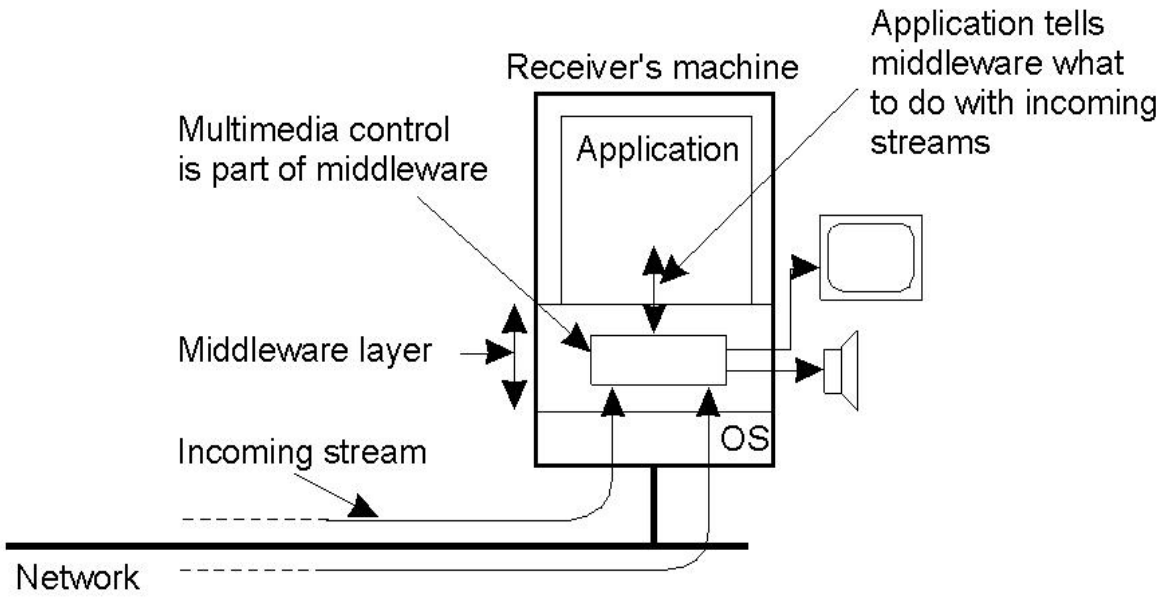
در این شکل ساختار کلی پروتکل RSVP را مشاهده می‌کنید. در این شکل، یک RSVP Process وجود دارد که Policy control و Admission control را به نوعی مدیریت می کند. همانطور که عنوان شد، این پروتکل Reserver Initiated است. یعنی دریافت کننده، اول پروتکل را آغاز می‌کند و سپس پس از

آنکه Sender مسیری را در اختیار گرفت با توجه به نیازمندی های مربوط به سرویس، Policy control و Admission control عملیات مربوطه را انجام می‌دهند.

آخرین مبحثی که در مورد مفاهیم مربوط به Stream Oriented Communication مورد نظر قرار می‌گیرد نحوه همگام سازی Stream هاست. مکانیزم همگام سازی در سطوح مختلفی ممکن است انجام شود. زمانی که عنوان می‌شود همگام سازی Stream ها، با این فرض این مبحث مطرح می‌شود که چندین Stream به طور همزمان از Sourse های مختلف به سمت یک Sink حرکت می‌کنند، اما مجموعه این Stream ها با هم یک داده را مدل خواهند کرد. بنابراین باید این داده ها به نوعی با هم همگام شوند. ساده‌ترین مدل همگام سازی این است که همگام سازی در پایین‌ترین لایه Abstraction انجام شود. این مطلب به این معنی است که همگام سازی بر سطح Data Unit های یک Simple stream انجام شود و به این صورت که در خواندن و نوشتن، اعمال خواندن و نوشتن مدیریت شود. اگر از این ایده استفاده شود، ایرادی که به آن وارد می شود این است که برنامه کاربردی (Application) وظیفه همگام سازی را بر عهده خواهد داشت.



با توجه به شکل بالا، در سمت ماشین Receiver چندین Stream به سمت Application وارد می شود و Application نحوه ترکیب Stream ها با یکدیگر را مدیریت می‌کند. به عنوان یک مثال از مفهوم کلی چندین Stream می توان فرض کرد که یکی از Streamها، Streamهای داده‌ای و دیگری Stream مربوط به ویدئو است.



روش دیگری که برای همگام سازی Stream ها مطرح می شود، این است که Interface ای تعبیه شود که این Interface عملیات خواندن و نوشتن و همگام سازی Stream ها را مدیریت کند. در واقع این Interface می‌تواند یک میان افزار (MiddleWare) باشد که در خارج از Application قرار می‌گیرد. Application تنها در یک سطح Abstract بالا و به صورت توصیفی برای این MiddleWare مشخص می‌کند که چطور با Stream های ورودی برخورد کند. در این شکل نحوه مدیریت Stream های ورودی به کمک یک MiddleWare نشان داده شده است. آنچه که در این MiddleWare مهم است این است که در واقع کنترل مالتی مدیا قسمتی از MiddleWare است.