جلسه چهاردهم

فرآیند (Process)

در این بخش مفهوم فرآیند (process) و نکاتی که در مورد فرآیندها در یک سیستم توزیع شده وجود دارد، بررسی خواهد شد. اگر بخواهیم در ابتدا یک مرور کلی بر تعریف فرآیند و نخ (Thread) داشته باشیم می‌توان به عنوان یک تعریف کلی عنوان کرد که فرآیند یک برنامه در حال اجرا است و نخ تعریف مشابهی مانند فرآیند دارد. به عبارت دیگر نخ ها به عنوان اجزایی از فرآیندها شناخته می‌شوند که می توانند به طور همزمان فعالیت کرده و کارپردازشی انجام دهند. قطعاً نخ زیر مجموعه ای از یک فرآیند است و عنوان می‌شود که ریزدانگی کوچکتری نسبت به فرآیندها دارد و قطعاً در یک فرآیند مجموعه نخ ها با یکدیگر همکاری خواهند کرد تا هدف اصلی آن که همان انجام وظیفه مربوط به یک فرآیند است برآورده شود. آنچه که مفهوم نخ‌ها و فرآیندها را در یک سیستم توزیع شده بااهمیت می‌سازد این است که سیستم‌های توزیع شده اجازه می‌دهند که مشتری ها و سرورها به گونه ای ایجاد شوند که ارتباطات و پردازش محلی آنها بتواند به صورت همزمان(Overlap) انجام شود. برای ایجاد این همزمانی یکی از نکات مهم و یکی از ابزارهای مهم استفاده از نخ ها و یا فرآیندها است. اما آنچه بدیهی است و می دانید این است که Context یا بستری که یک نخ بر روی آن اجرا می شود معمولاً بسیار ساده تر از Context مربوط به یک فرآیند است. در مورد نخ ها معمولاً تنها CPU context switch انجام می شود و حفاظت از نخ ها در مقابل یکدیگر در سطح یک Process در اختیار Application developer یا نویسنده برنامه است. در صورتی که حفاظت از فرآیندها در مقابل دسترسی غیرمجاز به فضای آدرس و داده‌های آنها در اختیار سیستم عامل است و این قضیه، Context یا فضای اطلاعاتی مربوط به فرآیندها را به مراتب پیچیده تر از نخ‌ها می‌سازد.

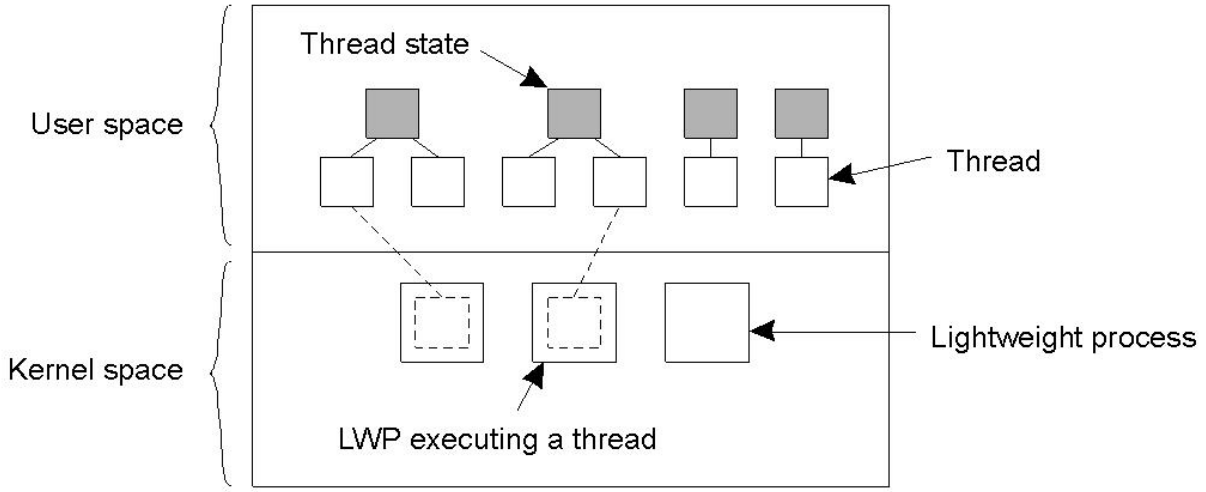
چند نخی (Multithreading)

چندنخی(Multithreading) مفهوم دیگری است که در حوزه نخ‌ها قابل بررسی است. چندنخی به نوعی یک (parallelism) اجرای موازی است که بر روی سیستم‌های چند پردازنده ای ایجاد می‌کنندو کارایی آن

در برنامه های بزرگ است. از دیدگاه مهندسی نرم افزار نیز می‌توان مفهوم چندنخی را توجیه کرد. از این دیدگاه چندنخی از آن جهت حائز اهمیت است که برنامه های بسیاری می توانند ساده‌تر ایجاد شوند اگر آنها را به صورت مجموعه ای از نخ‌ها یا اجزای کوچکتری که با یکدیگر همکاری می کنند مدل کرد. اما از مباحث پیشین به یاد دارید که پیاده سازی نخ ها در دو حالت سطح کاربر یا User level و در سطح هسته یا kernel level امکان پذیر بود. وقتی که نخ ها را در سطح کاربر پیاده سازی می کنیم آنگاه ایجاد و انجام Context switch آسان خواهد شد، چرا که اینها مفاهیمی خواهند بود که بر عهده کاربر است اما در این حالت Blocking یک thread معادل blocking process خواهد بود، یعنی با متوقف شدن یک نخ، کل process متوقف می شود. چنانچه نخ در سطح kernel یا هسته پیاده سازی شود آنگاه switching یا انتقال کنترل اجرا از یک نخ به نخ دیگر مشابه با مفهوم سوئیچینگ فرآیندهاست، به این معنی که از دیدگاه هسته نخ ها اجزای مجزایی هستند و زمان بندی آنها در هسته سیستم عامل انجام می شود.

فرآیندهای سبک وزن (Light weight process)

مفهوم دیگری که بسیار در سطح نخ‌ها حائز اهمیت است، مفهوم فرآیندهای سبک وزن یا Light weight process (LWP) است. فرآیندهای سبک وزن را به عنوان یک حالت ترکیبی از نخ‌های سطح کاربر و نخ‌های سطح هسته می شناسیم. در شکل نمونه‌ای از این نخ‌ها نشان داده شده است. این نوع فرایندها یا نخ‌ها که LWP نامیده شدند، در بستر یک فرآیند منفرد اجرا می شوند و قطعاً امکان وجود چندین LWP در داخل هر فرآیند وجود دارد.

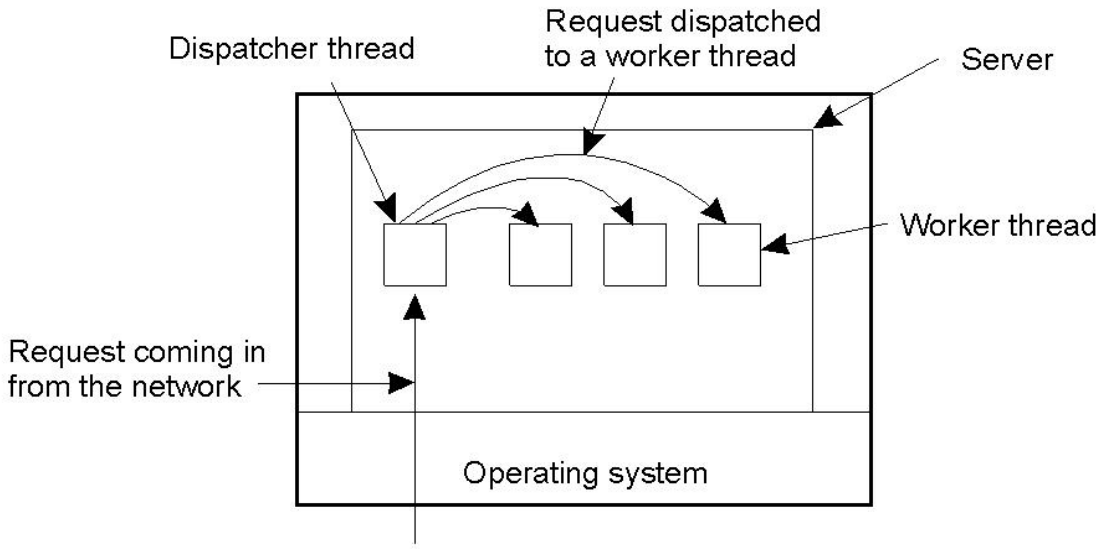


بنابراین همانطور که قبلاً اشاره شد، LWP یک هیبرید یا یک ترکیب از مجموعه ای از فرآیندهای سطح هسته و مجموعه ای از نخ‌های موجود در سطح کاربرمی‌باشد.

مفهوم threadها در یک سیستم توزیع شده

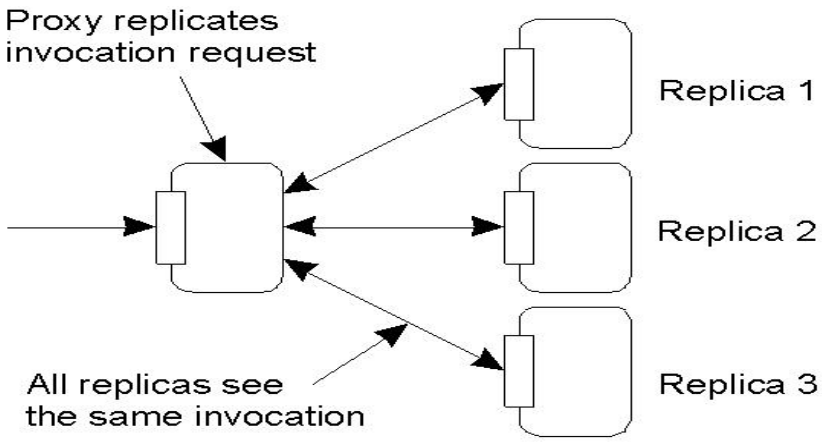
آنچه که به نخ‌ها در سیستم های توزیع شده اهمیت می بخشد، این است که می توان با استفاده از نخ‌ها چندین ارتباط همزمان منطقی را برقرار نمود. در این دیدگاه می توان کلاینتها و سرورها را نیز به شکل کلاینت‌های Multithread و سرور Multithread بررسی کرد. در یک کلاینت Multithread یا چند نخی آنچه که اتفاق خواهد افتاد این است که ممکن است ارتباطات با threadهای مختلفی برگزار شود و حتی در مفاهیمی مانند Object Sharing به ازاء هر Object، کپی‌های مختلفی یا تکرارهای مختلفی وجود داشته باشد، و هر ارتباط با یکی از این تکرارها برقرار شود و این امر کمک خواهد کرد که امکان ارسال همزمان داده وجود داشته باشد. در حالتی که یک سرور Multithread یا چند نخی وجود دارد، یک نخ توزیع کننده یا یک dispatcher thread در داخل سرور تعبیه می شود. وظیفه dispatcher این است که درخواست‌ها را خوانده و نخ‌های دیگر را با توجه به هر درخواست فعال می کند. فایل سرورها نمونه ای از این نوع سرورها

هستند. در شکل یک سرور چند نخی که با تکنیک Dispatcher worker ، سازماندهی شده است را می بینید.



با توجه به شکل، درخواست هایی که به سمت سرور می آید همگی در یک صف وارد Dispatcher می‌شوند،dispatcher thread این درخواست‌ها را به یک Thread کارگر (Worker Thread) ارسال می‌کند و این Thread های کارگر به طور همزمان می توانند درخواست هایی را که به آنها نسبت داده شده است، مدیریت کرده و پردازش کنند.

بحث Replication در سطح کلاینت که پیش از این عنوان شد، نیز حائز اهمیت است. به عنوان مثال اگر بخواهیم مفهوم Remote object را با استفاده از یک راه حل مبتنی بر مشتری مدیریت کنیم این امکان وجود دارد که تکرارهای متعددی از object ها در سطح مشتری ایجاد شود.

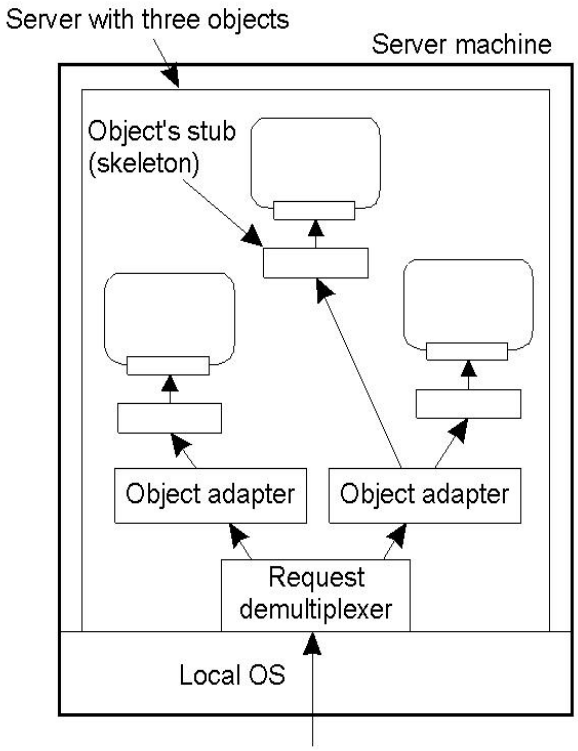


بنابراین همانطور که در شکل می بینید، قسمت مربوط به پروکسی ممکن است که هر درخواست را به سمت یک کپی ارسال کند و بدین گونه این امکان وجود خواهد داشت که بتوان همزمان object های مختلفی را فراخوانی کرد.

نخ‌ها در سیستم توزیع شده

در مفهوم سرورهای مربوط به سیستم های توزیع شده، نکات دیگری را می توان مورد بررسی قرار داد. از جمله این موارد، می توان این سرورها را به گروه‌های مختلفی از دیدگاه‌های مختلفی تقسیم بندی کرد. از جمله اینکه سرورها می توانند بدون وضعیت یا Stateless و یا Stateful باشند. سروری Stateless است که هیچ اطلاعی از مشتری خود ندارد و اگر وضعیت تمامی مشتری‌ها در اختیار سرورها باشد آن سرورها را Stateful می‌نامند. از دیدگاه دیگر یک Concurrent server یا یک سرور هم‌روند سروری است که درخواست‌ها را به نخ ها ارسال می کند. کلاینت ها، درخواست ها را به یک نقطه مشخص از سرور ارسال کرده و سپس سرور هر کدام از این درخواست ها را به یک نخ نسبت می‌دهد. در همین راستا می‌توان به عنوان نمونه، مفهومی از سرورها به نام Object Server ها را ارائه کرد. Object Server ها می‌توانند به عنوان اشیاء توزیع شده (Distributed Object) شناخته شوند، که هدف Object Server ها پشتیبانی این

Objectهای توزیع شده است. در این نوع سرورها هر Object به یک thread مختلف نسبت داده می‌شود و از threadهای متعددی برای هر فراخوانی استفاده می شود. در این سیستم Object adapter نیز وجود دارد و با تعریفی که از Adapterها در مباحث پیشین ارائه شد، در شکل نمونه ای از یک سرور که دارای سه Object است نشان داده شده است.

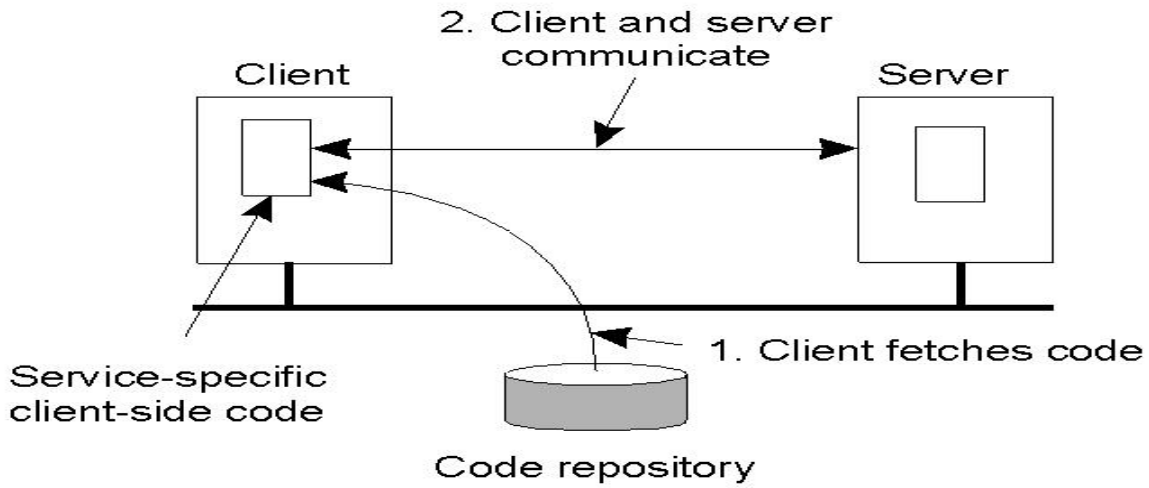


با توجه به شکل، یک Request demultiplexer ، درخواست ها را به نوعی تفسیر می کند و سپس با توجه به نوع درخواست و از طریق Object adapter متوجه خواهد شد که این درخواست‌ها باید به کدام Object ارسال شده و Object adapter سعی می کند که درخواست ها را به فرمتی تبدیل کند که توسط Object مربوطه قابل پذیرش باشد. بدیهی است که در این گونه معماری هر Object ی باید به سمت فضای مربوط به سرور حرکت کند. قرار گرفتن Object ها در فضای سرور بر عهده خود Object ها می‌باشد و این کار باید قبل از اینکه فراخوانی واقعی توسط Object adapter اتفاق افتد، رخ دهد.

مهاجرت کد (Code migration)

مفهومی که در اینجا بسیار حائز اهمیت است و باید در سیستم های توزیع شده به دقت مورد بررسی قرار گیرد، مبحث مهاجرت کدها و یا مهاجرت فرآیندها خواهد بود. در سیستم‌های توزیع شده ارتباط بین فرآیند ها و ماشین‌های مختلف محدود به انتقال داده نمی‌باشد. و ممکن است کدها نیز بین ماشین‌ها جابجا شوند. آنچه که باید در مهاجرت کدها مورد توجه قرار گیرد آن است که ابتدا تکنیک های مختلفی برای مهاجرت وجود دارد و دیگر اینکه مباحث مربوط به منابع محلی که توسط کدها مورد دسترسی قرار می‌گیرند مباحث با اهمیتی هستند که باید در مهاجرت کدها مورد توجه قرار گیرند. به طور کلی مهاجرت کدها در یک سیستم توزیع شده متشکل از مهاجرت فرآیندها، وضعیت اجرایی، سیگنال های معلق آن و برخی دیگر از قسمت‌های یک محیط است. مهمترین هدف مهاجرت کد این است که کارایی کلی سیستم بهبود یابد و با این بهبود سطح اجرای موازی(parallelism) افزایش یابد. هدف دیگر افزایش انعطاف (Flexibility) سیستم است چرا که اگر چنین ابزاری تعریف و طراحی شود می توان به صورت داینامیک و پویا سیستم های توزیع شده را پیکر بندی کرد. اما نکته بسیار مهم این است که بهبود کارایی در قبال انجام مهاجرت کدها معمولاً بستگی به برخی دلایل کیفی دارند و مدل های ریاضی کمتر اثر دارند. به عنوان مثال در پایگاه های داده ای بزرگ معمولاً بهتر است که قسمتی از کدهای مربوط به کلاینت به سمت سرور منتقل شود و تنها نتایج مربوط به جستجوها در طول شبکه انتقال پیدا کنند. بنابراین دیدگاه کیفی آن بدین صورت خواهد بود که پردازش داده نزدیک به مکانی که داده ها قرار دارند بهتر است. اما شاید در یک سیستم دیگر چنین پدیده ای بهتر نباشد. نکته دیگر در بحث مهاجرت کدها این است که انتقال پویای کدها نیاز به این دارد که پروتکلی برای downloading و initialization کدها استاندارد شود. یکی از تکنیک‌هایی که در مهاجرت کدها و در مبحث انتقال پویا مطرح می شود این است که تمام سیستم هایی که بر بستر یک شبکه قرار گرفته اند، مطابق شکل از یک پایگاه داده کد Code (Reposirtory) یا از انباره کد استفاده کنند و در زمانی که فراخوانی کدی(binding) انجام می‌شود، در آن حالت Client سعی می کند که کد را از سرور مربوطه دانلود کند و این عمل با استفاده از Code repository اتفاق می افتد. آنچه که در استفاده از این تکنیک مورد دقت باید قرار گیرد مفهوم امنیت و

مشکلات امنیتی است که ممکن است در این سیستم وجود داشته باشد. چرا که اگرCode repository مورد حمله قرار گیرد، ممکن است که اثرات بسیار بدی به بار آورد.



مکانیزم‌های مهاجرت کد (Code migration alternative)

جدا از اینکه عمل Migration چگونه انجام می شود، یعنی کدها چگونه از یک ماشین به ماشین دیگر منتقل می شوند و آیا این انتقال یک انتقال مستقیم است و یا به کمک یک repository انجام خواهد شد، انواع مختلفی را برای Mobility یا امکان جابجایی کدها می توان تعریف نمود. به طور کلی نحوه جابجایی کدها به دو گروه اصلی تقسیم می‌شوند:

۱- Weak Mobility (جابجایی‌پذیری از نوع ضعیف)

۲- Strong Mobility (جابجایی‌پذیری از نوع قوی)

هر کدام از این دو نوع Mobility به دو گروه تقسیم می شوند:

۱- Sender initiated یعنی آغازدهی شده توسط فرستنده

۲- Receiver initiated یعنی آغازدهی شده توسط گیرنده

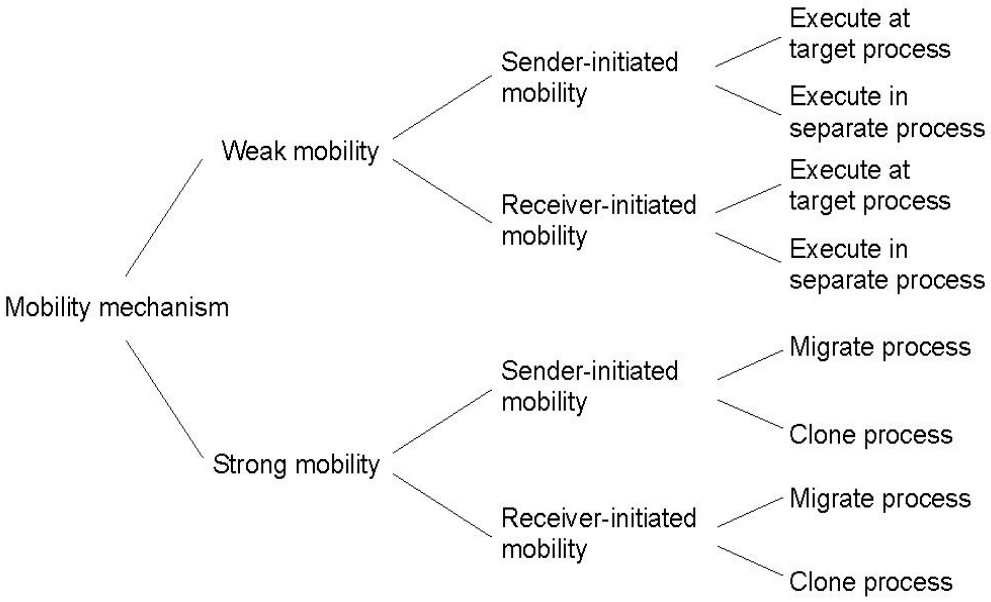
باز هر کدام از اینها به دو گروه تقسیم می شوند یکی در گروه اول، اجرا در داخل فرآیند هدف انجام خواهد شد و در حالت دوم اجرا در داخل یک فرآیند جداگانه صورت می‌گیرد.

حال به بررسی هر یک از این اجزاء و تعریف آنها می پردازیم. پیش از تعریف به این نکته دقت کنید که یک Frame work یعنی یک مجموعه کاری برای اجرای فرآیندها شامل اجزایی مانند Code segment شامل مجموعه‌ای از دستوراتی است که باید اجرا شود. Resource segment که ارجاع به منابع خارجی دارد. Execution segmentکه شامل وضعیت جاری اجرای یک کد مانند پشته ، وضعیت داده های خصوصی و امثال آن خواهد بود. این سه جزء با هم یک ماژول مربوط به یک فرآیند را می سازند که حال اگر قرار باشد این فرآیند مهاجرت کند باید حتماً این سه ماژول یا این سه segment را مدیریت کند.

انواع Mobility

برای اینکه بتوان مباحث مربوط به مدیریت مهاجرت را به خوبی مورد بررسی قرار داد، باید انواع Mobility را بررسی نمود. Weak Mobilityساده ترین نوع Mobility است که در این روش تنها Code segment و داده های اولیه جابجا می شوند و بخش های دیگر جابجا نمی شوند. بنابراین آنچه که در اینجا مهم است این است که برنامه ای که منتقل شده است همیشه از نقطه آغازین اجرا می‌شود. چنانچه فرآیندی قسمتی از کار خود را انجام داده باشد و با Weak Mobility جابجا شود باید مجدداً کارها را تکرار کند. بدیهی است که مزیت اصلی این کار سادگی است. Java applet ها نمونه هایی از این نوع Mobility هستند. نوع دیگر جابجایی همانطور که قبلا عنوان شد، Strong است. در حالت Strong mobility یا جابجایی از نوع قوی بخش اجرایی نیز می تواند به مانند بخش کد جابجا شود. آنچه که در اینجا مهم است این است که فرآیند در حال اجرا می تواند اجرای خود را از جایی که روی ماشین قبلی متوقف کرده بود روی ماشین جدید از سر بگیرد. قطعاً این روش برای پیاده سازی از Weak Mobility مشکل تر خواهد بود. همانطور که عنوان شد، حالت دیگری که در گروه بندی مربوط به مهاجرت فرآیندها مورد توجه قرار می گرفت، بحث Sender-

Initiation یا آغازدهی توسط ارسال کننده و Receiver-Initiation یا آغازدهی توسط دریافت کننده است. در Sender-Initiation مهاجرت از سمت ماشینی که کد در حال حاضر بر روی آن قرار گرفته است آغاز می شود. به عنوان یک مثال در یک برنامه جستجو، مهاجرت کد مربوط به جستجو از سمت ماشینی که دستورجستجو در آن صادر شده به سمت یک Data base server روی وب آغاز می شود. در حالت Receiver-Initiation، آغازدهی اولیه به وسیله ماشین مقصد انجام می شود.



Java applet ها می‌توانند نمونه‌ای از این حالت‌ها باشند. معمولاً وقتی که بحثSender initiation پیش می آید مفهوم مربوط به Uploading کدها نیز مطرح اند. در Uploading کدها باید دقت کرد که امنیت به شدت مورد توجه قرار گیرد، چرا که سرور لزوماً تمام کلاینت ها را به دقت نمی شناسد و بنابراین باید سعی کند که این شناخت ایجاد شود و منابع باید محافظت شوند. در downloading کدها امکان اینکه Downloading به صورت مخفیانه (anonymously) انجام شود، وجود دارد. اما همانطور که عنوان شد، چنانچه هر کدام از این گروه‌ها مطرح باشند، در آخر دسته بندی اجرا به دو نحو، اجرا در داخل فرآیند مقصد یا به عنوان یک فرآیند مجزا می تواند انجام شود. به عنوان نمونه ای از اینکه اجرا در داخل یک فرآیند مقصد

انجام می شود، می توان به مفهوم java applet ها که در داخل Browser ها اجرا می‌شوند اشاره کرد. ایرادی که بر آن وارد است این است که فرآیند مقصد باید در این روش از رفتارهای بد و غیرمتداول اجرایی کدها به نوعی محافظت شود که اجرای کد مهاجر باعث ایجاد مشکل برای این فرآیند نشود. در مفهوم مربوط به Strong Mobility نکته ای مورد توجه باید قرار گیرد و آن مفهوم Cloning است. در تکنیک Cloning هر کپی از یک پروسس بر روی یک ماشین مختلف اجرا می شود، Clone process در نوع Strong Mobility فرآیندی است که به طور موازی با فرآیند مبدأ یا اولیه آن (Original)اجرا خواهد شد.