6
Trabalhos Relacionados

O objetivo deste capítulo é apresentar alguns trabalhos relacionados ao Moratus e compará-los entre si, segundo alguns critérios selecionados que são considerados importantes em processos de adaptação de software.

6.1.Critérios Comparativos

Esta seção apresenta os critérios para estabelecer a comparação entre os trabalhos que são apresentados no decorrer deste capítulo, que são:

- *Transparência da Adaptação*. Refere-se à percepção do usuário do sistema no momento da substituição de um serviço distribuído. Foram consideradas duas formas de substituição de serviços: interrompendo a execução do sistema até que sua substituição dos serviços esteja finalizada, tanto no dispositivo local como nos dispositivos remotos; ou substituindo sutilmente os serviços de forma que se o usuário perceber a breve indisponibilidade do sistema, ela seja discreta.
- *Tratamento da Desconexão*. Refere-se à forma como os dispositivos tratam a desconexão de um dispositivo que se desconectou.
- *Coordenação da Adaptação Distribuída*. Refere-se à existência de um mecanismo para coordenar a adaptação de um serviço em dois ou mais dispositivos.
- *Composição de Pilhas de Protocolos*. Refere-se à capacidade de um sistema compor pilhas de protocolos.

6.2.Ensemble

O Ensemble é uma arquitetura para protocolos de rede reconfiguráveis e bastante otimizada, que tem o objetivo de auxiliar o desenvolvimento de aplicações adaptáveis (Hayden, 1998). A funcionalidade básica do Ensemble é

PUC-Rio - Certificação Digital Nº 0611910/CA

cuidar da composição de grupos e prover um mecanismo para permitir a comunicação entre os membros do grupo.

A arquitetura do Ensemble é baseada no conceito de pilhas de protocolos. Tal pilha é construída a partir de módulos de micro-protocolos, empilhados e re-empilhados de diversas formas para atender aos requisitos da aplicação. Os micro-protocolos do Ensemble podem implementar protocolos de fragmentação e junção, controle de fluxo, encriptação, controle de grupo e ordenação de mensagens, entre outras coisas.

O Ensemble possui um framework para permitir a composição de camadas, que não apenas permite a construção da pilha de micro-protocolos para uma aplicação em particular em um determinado ambiente, como também é um instrumento para prover um bom desempenho quanto a utilização das diversas camadas de um pilha de protocolos.

Para o Ensemble, as entidades denominadas detectores de contexto iniciam ações de reconfiguração. Portanto, cada detector é projetado para monitorar um determinado conjunto de variáveis de contexto, que aqui é chamado de ambiente, monitorando possíveis violações de premissas que foram assumidas, quando a configuração atual do sistema foi determinada. Tais detectores de contexto são implementados como micro-protocolos.

À medida do possível, a adaptação é feita de forma transparente para a aplicação. As primeiras camadas a serem adaptadas são as mais baixas. Caso elas não estejam aptas a responderem corretamente a uma mudança no ambiente, elas repassam a notificação para a camada imediatamente acima. Eventualmente a aplicação é notificada. Portanto nesse caso, a aplicação terá que decidir como reconfigurar a camada da qual ela é responsável. Esse processo pode ser visualizado através da Figura 16.

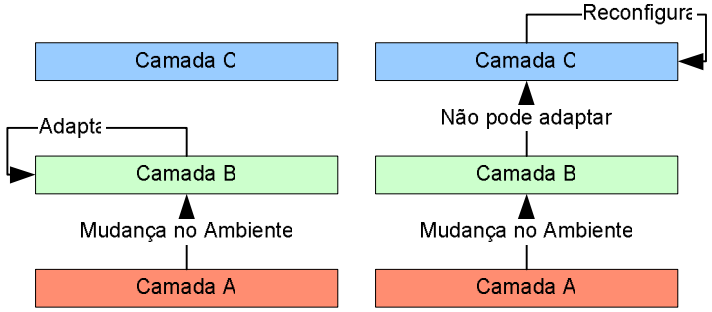


Figura 16 – Retransmissão da Mensagem de Adaptação entre Camadas.

PUC-Rio - Certificação Digital Nº 0611910/CA

A reconfiguração no Ensemble é feita através de um mecanismo denominado *Protocolo de Troca de Protocolo* (PTP). Ele é um protocolo tolerante à falhas que sincroniza os participantes, os assiste na finalização de suas pilhas de protocolos e cuida dos procedimentos necessários para a substituição das pilha antes da nova inicialização da comunicação. Além dessas funcionalidades, o PTP inclui mecanismos através dos quais os participantes podem baixar o código dos micro-protocolos e armazená-los para uso posterior. Dessa forma, através do PTP, o Ensemble pode tratar de adaptações que não foram previamente previstas.

6.2.1.Arquitetura do Ensemble

O Ensemble fornece um framework para a composição de módulos (micro-protocolos) cuja combinação define um protocolo. Cada módulo é associado a uma interface de um micro-protocolo do Ensemble, e possui uma interface superior e uma interface inferior. A interface superior do módulo se comunica com a interface inferior do módulo, abstratamente, acima, na hierarquia de pilha de protocolos.

A interface é direcionada a eventos, ou seja, módulos transmitem eventos para outros módulos adjacentes. Certos tipos de eventos trafegam para baixo, e outros para cima, na pilha de protocolos. Eles são chamados de *eventos para baixo* e *eventos para cima*, respectivamente. Para tentar contemplar uma vasta quantidade de micro-protocolos, o Ensemble identificou aproximadamente 40 tipos de eventos, dentre os quais os mais relevantes são mostrados na Figura 17. É importante ressaltar, no entanto, que a maioria dos micro-protocolos apenas repassam eventos para cima ou para baixo, sem de fato processá-los. Apenas os eventos que interessam a uma camada é que são tratados por ela.

"Eventos para baixo"	"Eventos para cima"
Enviar Mensagem Deixar Grupo Requisitar Timeout Reconfigurar	Entregar Mensagem Problemas de Comunicação Timeout Não pode adaptar

PUC-Rio - Certificação Digital Nº 0611910/CA

Figura 17 – Uma amostra dos eventos mais comuns de serem transitados para cima e para baixo na pilha de protocolos.

Diferentemente de outras propostas de modelos de pilhas de protocolos, a aplicação não se situa no topo da pilha de protocolos (Figura 18). Ao invés disso, ela possui uma interface com cada uma das camadas da pilha de protocolos. Por exemplo, a camada do protocolo da janela deslizante deve oferecer uma interface para mudar o temporizador de retransmissões. A camada *Application* oferece puramente interfaces com camadas da pilha de protocolos para enviar e receber mensagens. Ela gera um evento *enviar para baixo*, associado a sua interface de envio e intercepta eventos do tipo *enviar para cima*, associado a interface de recebimento. Uma aplicação pode inclusive instalar mais de uma camada de aplicação em uma mesma pilha de protocolos para se comunicar com mais camadas. Por exemplo, pode haver uma interface com a camada abaixo da camada de encriptação para enviar mensagens não-codificadas ou pode haver uma camada acima da camada de encriptação para enviar mensagens codificadas. Para reduzir a latência, a aplicação tentará usar a interface mais baixa da camada Application para enviar e receber mensagens. No topo da pilha de protocolos reside a camada chamada de *rolha* que será responsável por descartar eventos do tipo *para cima*.

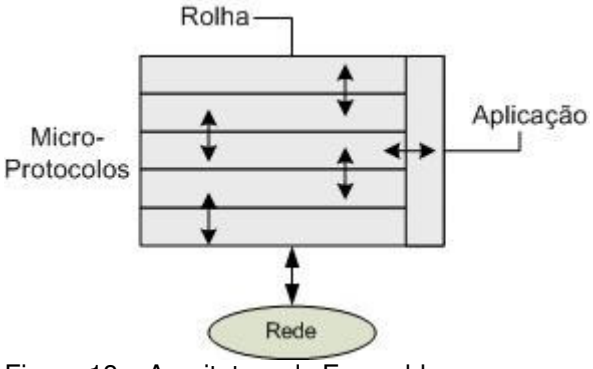


Figura 18 – Arquitetura do Ensemble.

6.2.2.Protocolo de Troca de Protocolo

Quando o sistema detecta que as premissas definidas inicialmente sobre o ambiente no qual ele está sendo executado, não são mais válidas, ou a aplicação passa a ter outros requisitos diferentes dos anteriores, o sistema irá compor uma nova pilha de protocolos para se adequar aos novos requisitos. Como a nova pilha

PUC-Rio - Certificação Digital Nº 0611910/CA

de protocolos usa um cabeçalho de mensagens diferente, é necessário haver um mecanismo de coordenação eficaz para finalizar o processamento na configuração atual e trocar a pilha existente por uma nova pilha. Entretanto um problema se ocorre quando os participantes de um grupo perdem a conexão ou falharem de alguma forma. Para solucionar esses problemas, o Ensemble define o PTP (Protocolo de Troca de Protocolo), usado para instalar uma nova pilha de protocolos em um conjunto de participantes.

O PTP é composto por um conjunto de micro-protocolos que devem ser empilhados para permitirem o processo de reconfiguração.

Cada instância da pilha de protocolos é unicamente identificada pelo *Identificador de Instâncias da Pilha de Protocolos* (ID-IPP). A lista de nós participantes que usa uma pilha de protocolos, é associada a cada IPP. Todas as mensagens enviadas por um IPP começam com o ID-IPP, para que estas possam ser encaminhadas pela interface de rede rede para as pilhas apropriadas. A tarefa do IPP é:

1. Finalizar os micro-protocolos da antiga pilha de protocolos;
2. Distribuir uma nova pilha e atribuir um ID-IPP a ela; e
3. Iniciar uma nova pilha de protocolos assim que possível.

Primeiramente, o IPP elege um coordenador através da escolha do participante que possui o menor endereço de rede (Figura 19). Então, o coordenador gera um novo ID-IPP. Depois dissemina uma mensagem do tipo FINALIZAR para o grupo, que inclui uma descrição da nova pilha de protocolos, a pertinência a um grupo e o ID-IPP da pilha do coordenador. A mensagem é entregue a todos os participantes alcançáveis incluindo o coordenador. Assim que essa mensagem é recebida, cada um dos participantes do grupo constrói uma nova pilha e registra o ID-IPP para que futuras mensagens possam ser entregues a essa nova pilha.

PUC-Rio - Certificação Digital Nº 0611910/CA

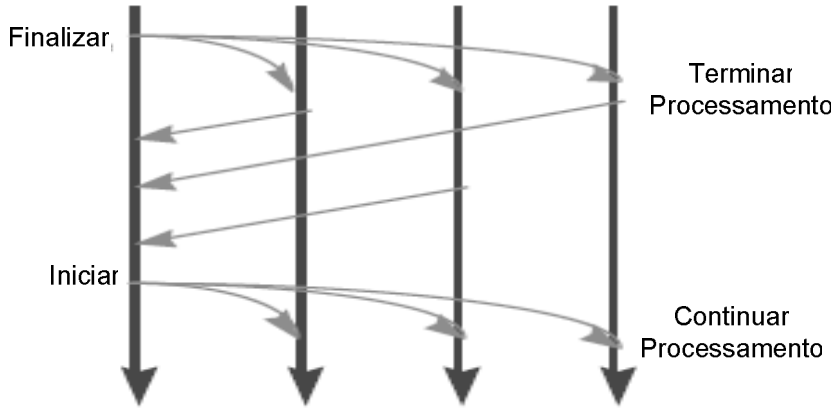


Figura 19 – O protocolo de troca de protocolos.

Da mesma forma, um evento do tipo FINALIZAR é entregue à camada mais alta da antiga pilha (*rolha*). Cada camada da pilha repassa o evento FINALIZAR para sua camada sub-adjacente para interromper o envio de mensagens. Porém, antes de repassar os eventos, a camada deve executar seu próprio protocolo para finalização interna. Quando o evento FINALIZAR chega ao fundo da pilha, indicando que todas as camadas executaram o procedimento de finalização local, uma mensagem do tipo CONFIRMAR-FINALIZACAO é enviada para o coordenador. Uma vez que o coordenador tenha recebido todas as mensagens CONFIRMAR-FINALIZACAO de todos os participantes do seu grupo, ele dissemina uma mensagem do tipo INICIAR referente à nova pilha a ser ativada, utilizando os novos ID-IPP. Quando tal mensagem é recebida, cada participante entrega o evento INICIAR para a camada mais baixa da nova pilha e a antiga camada é finalmente descartada. O evento INICIAR percorre a pilha para cima executando a função finalizar de cada camada e ao alcançar a camada mais superior (rolha), o evento é dissipado.

O PTP é considerado tolerante a falhas. Foi percebido que para recuperar mensagens perdidas, apenas um protocolo para efetuar retransmissões é o suficiente. No caso da falha ou desconexão de um participante, o coordenador dissemina novamente uma mensagem do tipo FINALIZAR para todos os participantes. O evento FINALIZAR que é entregue para a parte mais baixa da pilha contém a identificação da pertinência a um grupo, permitindo que os micro-protocolos se recuperem da espera por mensagens de participantes falhos. O

PUC-Rio - Certificação Digital Nº 0611910/CA

coordenador apenas aguarda a mensagem CONFIRMAR-FINALIZACAO dos participantes.

Caso o coordenador falhe ou se desconecte, o participante que tenha o segundo menor endereço de rede assume a posição do antigo coordenador e dissemina uma mensagem FINALIZAR. Em nenhuma circunstância o PTP aguarda indefinidamente. Caso seja necessário, o participante instala apenas a instância de uma pilha e retoma as operações.

6.3.Processo de Adaptação Graciosa

A arquitetura proposta nesse trabalho (Chen, 2001) consiste de múltiplos componentes organizados em forma de camadas hierarquicamente compostas. Tais camadas abrangem desde o nível do sistema operacional e do subsistema de rede até componentes do *middleware*. Cada uma dessas camadas deve ser adaptável e pode ser composta por diversos componentes adaptáveis. Por exemplo, a camada de *middleware* que pode implementar um serviço de comunicação de grupo poderia usar diversos componentes adaptáveis para implementar a transmissão confiável de mensagens, ordenação de mensagens, gerência de grupo e comunicação segura. Por sua vez, cada um dos componentes pode implementar adaptações do algoritmo que implementa sua funcionalidade.

6.3.1.Arquitetura

A arquitetura básica proposta nesse trabalho é a ilustrada na Figura 20. Tal arquitetura consiste de diversas camadas que podem ser adaptáveis ou não-adaptáveis. As camadas adaptáveis são compostas por entidades menores, denominadas *componentes adaptáveis* (CAs). À esquerda da Figura 20 pode-se perceber um elemento denominado *Controlador de Adaptação*, que deve coordenar a adaptação entre camadas e entre componentes em um determinado computador.

PUC-Rio - Certificação Digital Nº 0611910/CA

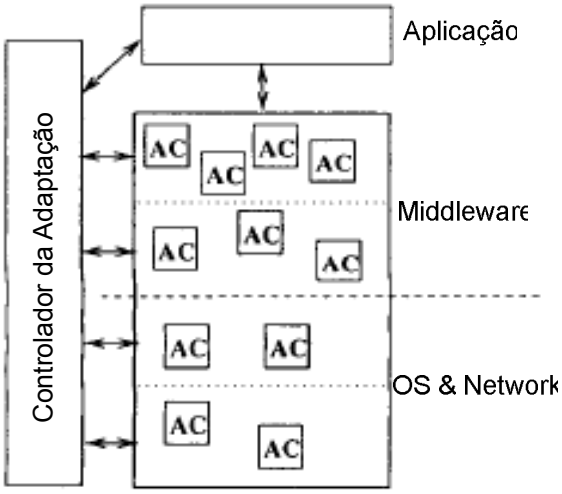


Figura 20 – Arquitetura do sistema no processo de adaptação graciosa.

A coleção de componentes adaptáveis em múltiplos computadores que cooperam para implementar um determinado serviço distribuído é denominada *componentes adaptávis distribuídos* (CAD). O tipo de funcionalidade oferecida por um CAD pode ser desde um mecanismo de ordenação de mensagens até um mecanismo de replicação ou consistência de dados.

Como pode ser visualizado na Figura 21, a estrutura interna de um CA consiste de dois tipos de módulos diferentes: um módulo adaptador de componentes (MAC) e módulos de algoritmos de adaptação alternativos (MAA). Cada MAA oferece um algoritmo diferente para a implementação de uma mesma funcionalidade de um componente. Já o MAC, controla o comportamento adaptativo de um componente. Ele monitora constantemente o componente e quando percebe que algum outro MAA é mais adequado para implementar a funcionalidade do componente, é iniciado um processo coordenado de troca entre dois algoritmos em um grupo, permitindo uma possível transferência de estados entre os MAAs, caso isto seja necessário.

PUC-Rio - Certificação Digital Nº 0611910/CA

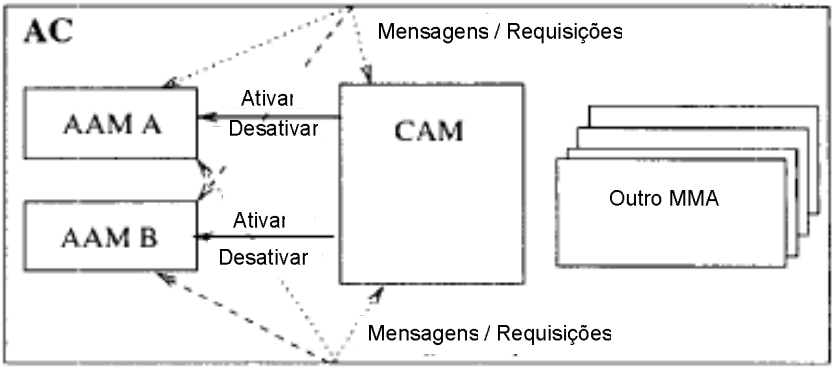


Figura 21 – Estrutura de um componente adaptável.

Um grande objetivo deste trabalho é minimizar a dependência entre diferentes módulos de um CA. A independência entre MAAs permite que diferentes módulos possam ser desenvolvidos isoladamente e que novos módulos possam ser substituídos sem que os atuais sejam modificados, enquanto a independência entre MACs e MAAs permite que o MAC controle um conjunto qualquer de MAAs, sem a necessidade de prever todos os módulos possíveis que serão utilizados. A independência é obtida através da especificação de um conjunto de operações que cada CAM e MAA deve exportar para permitir: a adaptação, a identificação das etapas de adaptação e a transferência de estados em cada passo. Porém, não é necessário que as estruturas de dados internas e o formato das mensagens sejam os mesmos entre MAAs alternativos.

6.3.2.Técnicas de Adaptação

Todo o processo de adaptação de um CAD pode ser dividido em três fases: detecção de mudanças, consenso e ação de adaptação (Hiltunen, 1996).

6.3.2.1.Detecção de Mudanças

A primeira fase é responsável por detectar alguma mudança no ambiente de execução e determinar se é necessário ou não efetuar alguma adaptação em resposta a essa mudança. A arquitetura descrita nesse trabalho não especifica que módulo seria responsável por detectar as mudanças no ambiente pois, dependendo do tipo de serviço e da abordagem da implementação usada, tanto o MAC como o MAA corrente poderiam ser usados.

PUC-Rio - Certificação Digital Nº 0611910/CA

O MAC usa as *funções de adaptação* associadas a cada MAA para determinar o melhor MAA para o atual conjunto de requisitos e ambiente de execução. Cada uma dessas funções mapeia um vetor de valores que refletem o atual estado global do sistema, *sysState[]*, em um vetor de valores que provê uma indicação quantitativa da relação do MAA com o estado atual. Por exemplo, para um serviço de comunicação, *sysState[]* deve conter a taxa de falha e a latência fim-a-fim da rede e a saída, após o processamento da função de adaptação, teria o número de mensagens esperadas para a entrega da mensagem de uma aplicação (associado a taxa de falha) e o tempo de resposta esperado (associado a latência fim-a-fim).

Após cada um dos MAAs informar os resultados de cada uma das funções de adaptação, o MAC seleciona o melhor MMA para assumir o processamente no novo ambiente. A forma como esse processamento é feito é específica para cada tipo de serviço, onde é possível identificar diversas políticas e heurísticas. Nesse trabalho, é assumido apenas que uma função *bestFit()* irá receber como argumento diversos vetores *sysState[]* e irá retornar o nome do MAA mais adequado, após invocar as funções de adaptação de cada módulo. Caso o resultado da função bestFit() seja diferente do atual MAA, o MAC irá iniciar um processo de consenso. É importante ressaltar que é desejável que a função bestFit() considere o custo da adaptação para otimizar o CA, além de tentar prever a oscilação entre módulos diferentes.

PUC-Rio - Certificação Digital Nº 0611910/CA

6.3.2.2.Consenso

A segunda fase do processo de adaptação é o processo de consenso, no qual deve ser decidido, entre todos os CAs, se todos os participantes consideram que de fato houve uma mudança no ambiente, e se um processo adaptativo deve ser executado. Há diversas abordagens para a obtenção do consenso.

Uma das abordagens possíveis é alcançar o consenso através do estado global do sistema e em seguida determinar deterministicamente se, e qual, processo adaptativo deve ser executado. Nessa abordagem, o processo de consenso dissemina o vetor sysState[] local para todos os MACs e cada MAC aplica a mesma função determinística para calcular um estado global do sistema, *globalSysState[]*. Uma vez que o estado global é calculado, ele é usado como

entrada da função bestFit() para que seja encontrado o MAA mais adequado. É assumido que as funções de adaptação e bestFit() também sejam determinísticas, o que representa que todo MAC tomara a mesma decisão.

Uma outra abordagem é permitir que cada MAC escolha localmente o MAA mais adequado baseado em seu estado local, depois o processo de consenso é usado para escolher um MAA entre os candidatos propostos. Se a função de seleção é determinística e cada MAC recebe a mesma entrada, todos os MACs serão capazes de tomar a mesma decisão.

Em ambos os casos, se, em um determinado instante, a decisão é que um MAA é mais adequado do que outro, o processo de adaptação é executado em cada um dos computadores para alterar de um algoritmo para o outro.

6.3.2.3.Ação da Adaptação

Por fim, a última fase define o chamado *processo de adaptação gracioso* (graceful adaptation process), o qual organiza a troca de um MAA para outro de forma que o CAD continue operando corretamente. Porém, é praticamente inviável trocar o algoritmo que está sendo usado sem levar em consideração alguns fatores como: o tratamento de mensagens que estão em trânsito durante a troca e os diferentes momentos em que o algoritmo é trocado em cada um dos computadores participantes. Portanto, esse trabalho utiliza o chamado *Protocolo de Adaptação Graciosa* para coordenar tal processo.

O protocolo é baseado na identificação de fluxos de mensagens que passam através de um CA. Tipicamente, um CA lida com dois fluxos de mensagens: um *fluxo de saída*, que é o fluxo de mensagens que chega de uma camada superior, onde as mensagens serão processadas e enviadas para uma camada inferior, e o *fluxo de entrada*, que recebe mensagens que chegam da rede, (onde as mensagens serão processadas e enviadas para uma camada superior). Dados esses dois fluxos de mensagens, há três passos no protocolo para substituir de um MAA_{antigo} para um MAA_{novo}:

1. *Preparação*. Cada CA se prepara para receber mensagens que futuramente serão enviadas pelo MAA_{novo} ou quaisquer mensagens relacionadas à adaptação enviadas pelo MAA_{antigo}.

PUC-Rio - Certificação Digital Nº 0611910/CA

2. *Troca da Saída*. Cada CA troca o módulo responsável por enviar mensagens, do MAA_{antigo} para o MAA_{novo}.
3. *Troca da Entrada*. Cada CA troca o módulo responsável por receber mensagens, do MAA_{antigo} para o MAA_{novo}.

Ambos os MAAs estão ativos durante todos os três passos, porém o MAA_{antigo} pára de processar mensagens que serão enviadas no segundo passo, e pára de receber mensagens no terceiro passo. Entre os passos troca da saída e troca da entrada, o fluxo de entrada das mensagens que estão chegando pode conter mensagens enviadas por ambos MAA_{antigo} e MAA_{novo}, onde cada mensagem deve ser processada por seu respectivo módulo em um determinado computador. Portanto, é necessário algum mecanismo no CA para delegar as mensagens para o módulo apropriado de cada MAA, baseado por exemplo no formato da mensagem ou em algum tipo de identificador da mensagem.

Na seqüência de passos apresentada anteriormente, o passo da preparação deve ser completado por todos os CAs antes que qualquer CA possa executar a troca para o passo troca de saída, já que todo o ambiente deverá estar preparado para receber novas mensagens antes que qualquer mensagem seja enviada. Esse requisito é implementado por um protocolo *barrier synchronization* executado no fim do primeiro passo. Contudo, esse protocolo é executado em *background* sem interferir na transmissão de mensagens por parte do MAA_{antigo}. Ainda, o passo da troca de saída deve ser executado atomicamente, para que nenhuma mensagem que esteja sendo recebida seja perdida ou processada por ambos os algoritmos.

Dependendo da semântica do serviço, a transferência de estado entre o MAA_{antigo} e o MAA_{novo} deve ocorrer nos passos 2 e 3. Portanto, durante o passo troca de saída, por exemplo, o MAA_{antigo} pode transferir as mensagens que serão transmitidas e que ele ainda não processou para o MAA_{novo}. Enquanto esse procedimento não é estritamente necessário, já que qualquer um dos módulos pode processar qualquer mensagem que está prestes a ser enviada, ele pode agilizar o processo de adaptação. Da mesma forma, em alguns casos, o MAA_{antigo} pode transferir as mensagens recém chegada que ainda não foram processadas para o o MAA_{novo} no início do passo transferência de chegada. Entretanto isso é apenas possível apenas se os formatos das mensagens forem equivalentes. Caso não sejam, então o MAA_{novo} deve armazenar todas as mensagens que chegaram

PUC-Rio - Certificação Digital Nº 0611910/CA

para ele até que o MAA_{antigo} processe todas a suas mensagens e que o passo 3 seja tomado. Dessa forma, pode-se garantir que a ordenação de mensagens não seja violada durante o processo de adaptação.

6.3.2.4.Interface dos Módulos de Algoritmos de Adaptação Alternativos

Cada MAA deve oferecer uma API apropriada para executar os três passos do protocolo de adaptação gracioso. Portanto cada MAA deve implementar as interfaces mostradas na Figura 22. As interações são ordenadas por tempo do topo para o fundo. Na figura, o passo da preparação é implementado por PreAtivacao() e PreDesativacao(). Desativacaote() é usado para iniciar o processo de troca, enquanto Ativacao_OS() e Ativacao_IS() implementam os passos de troca de saída e troca de entrada respectivamente.

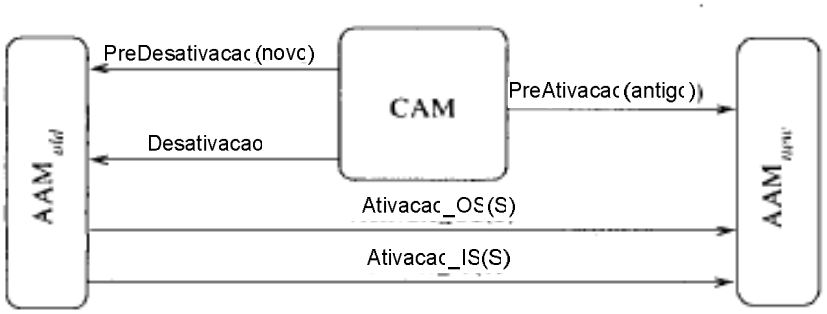


Figura 22 – Processo de Adaptação.

6.4.NeCoMan: Network Consistency Management

O objetivo desse trabalho é permitir a adaptação de serviços ponto-a-ponto em tempo de execução. Para os autores, tais serviços são aqui descritos como um par de entidades colaborativas que implementam um determinado serviço distribuído (Janssens, 2004). Essa descrição se encaixa bem no funcionamento de serviços como: compressão, fragmentação e encriptação de dados.

Para garantir que a adaptação dos serviços ponto-a-ponto não prejudique o correto funcionamento de uma rede, é essencial que o processo de adaptação dos serviços seja *seguro*, evitando que um serviço seja inadequadamente adaptado. Por exemplo, se durante o processo de adaptação um nó da rede receber um componente para codificar dados e não receber um componente para decodificá-

PUC-Rio - Certificação Digital Nº 0611910/CA

los, as mensagens codificadas não serão processadas corretamente, gerando o funcionando incorreto da rede. Assim, a adaptação de serviços em tempo de execução requer um protocolo para coordenar tal adaptação, de forma a preservar a integridade da rede nesses períodos (Janssens, 2004b).

Portanto, foi construído o middleware NeCoMan (*Network Consistency Management*), que é uma plataforma distribuída para coordenar a adaptação segura de serviços ponto-a-ponto. Este middleware implementa um protocolo distribuído que permite controlar a adição, substituição e remoção segura de serviços, de uma forma transparente para os usuários da aplicação, construída com base nessa plataforma. Além disso, o NeCoMan aprimora o processo de adaptação baseado em algumas propriedades dos serviços que serão adaptados e do ambiente de execução no qual se encontra imerso. Dessa forma, personalizando o processo de adaptação, é possível reduzir a penalidade sofrida pelo processo de adaptação transparente de serviços em tempo de execução.

PUC-Rio - Certificação Digital Nº 0611910/CA

6.4.1.Processo de Adaptação Segura de Serviços Distribuídos

O algoritmo que orquestra o processo seguro de adaptação de serviços do middleware é composto por três fases: instalação do novo serviço, ativação do novo serviço e remoção do antigo serviço. Esse algoritmo pode ser melhor visualizado através da Figura 23.

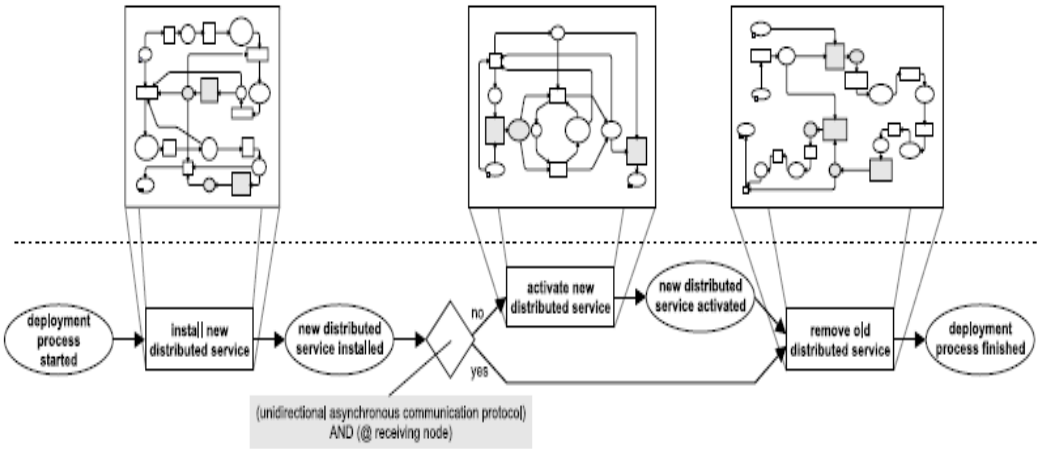


Figura 23 – Algoritmo do processo de adaptação segura de serviços.

6.4.1.1.Instalação de Novos Serviços

O processo de adaptação é iniciado com a instalação de um novo serviço ponto-a-ponto em ambos os nós. Após isso, dois serviços irão coexistir em um mesmo nó da rede: um serviço instalado e ativo, que é o serviço corrente, e um serviço instalado e ainda inativo que será o novo serviço a ser utilizado.

Para permitir que a adaptação do serviço seja feita da forma mais transparente para o usuário, é necessário que o período de substituição dos serviços seja o mais breve possível. Portanto, para o NeCoMan os serviços são compostos por dois tipos de módulos: o módulo transmissor e o módulo receptor. É através dessa estrutura que dois serviços poderão executar em paralelo durante um período de tempo no processo de adaptação, como será explicado mais adiante.

Como os dois serviços, antigo e novo, estarão executando em paralelo durante o processo de adaptação, é necessário também instalar um mecanismo de suporte para distinguir as mensagens de cada uma das versões do serviço. Portanto, isso inclui a instalação de (1) um mecanismo responsável por inserir nas mensagens um identificador da versão do módulo remetente que processou cada uma das mensagens; (2) um mecanismo para demultiplexar as mensagens quando chegarem a seu destino; e (3) delegar as mensagens recebidas para a sua respectiva versão do módulo destinatário.

PUC-Rio - Certificação Digital Nº 0611910/CA

6.4.1.2.Ativação do Novo Serviço

Nessa fase, o novo serviço instalado torna-se ativo. Essa operação apenas envolve o nó remetente, que deve finalizar o antigo módulo transmissor e redirecionar as suas mensagens para a nova versão desse módulo. Como o nó destinatário já foi preparado para tratar as mensagens marcadas pelos módulos transmissores de ambas versões, não é necessário efetuar mais nenhuma operação nessa fase.

É importante ressaltar que a reconfiguração segura de software requer que os módulos envolvidos estejam: consistentes (*consistent*) e congelados (*frozen*) (Kramer, 1990). Quando módulos de software são *consistentes*, eles não incluem resultados de serviços parcialmente completos. Quando módulos de software estão *congelados*, eles não processam mais nenhuma requisição e não possuem

mais requisições pendentes para serem aceitas ou processadas. Quando um componente de software está consistente e congelado, pode-se dizer que ele se encontra no estado de repouso (*quiescence*) (Ambriola, 1993).

6.4.1.3.Remoção do Antigo Serviço

Nessa fase, o antigo serviço deve ser completamente removido do nó. Como o antigo módulo transmissor foi finalizado na fase de ativação (anterior), ele pode ser seguramente removido. Devido a mensagens enviadas pelo antigo módulo transmissor, que ainda se encontram em trânsito, entre ambos os nós, o antigo módulo receptor deve ser monitorado até que chegue a um estado consistente (Ambriola, 1993). Portanto, apenas quando todas as mensagens que foram enviadas pelo antigo módulo transmissor alcançarem seu destino é que o antigo módulo receptor poderá ser seguramente removido. Por fim, como todas as novas mensagens trocadas entre ambos os nós utilizam o novo módulo transmissor, não há mais necessidade de haver um mecanismo de suporte para distinção de mensagens de versões de serviços diferentes, portanto o mecanismo de suporte que foi instalado na fase de instalação, pode ser seguramente removido nessa fase.

6.4.2.Reflexividade

Como mencionado anteriormente, o NeCoMan personaliza o processo de adaptação de serviços de acordo com as propriedades do serviço e do ambiente de execução. Portanto, pode-se considerar que o NeCoMan é um middleware reflexivo.

Para o NeCoMan há três caraterísticas principais que podem ser personalizáveis e dependendo das propriedades selecionadas dessas características, o protocolo será personalizado de uma forma particular para ser usado. As três características consideradas pelo *middleware* são: *o modelo de comunicação do serviço*, *tolerância ao embaralhamento de pacotes* e *transferência de estado*.

Cada fase do processo de adaptação – instalação do novo serviço, adaptação do novo serviço e remoção do antigo serviço - possui um modelo genérico que pode ser modificado de acordo com a seleção das propriedades das três

PUC-Rio - Certificação Digital Nº 0611910/CA

características mencionadas anteriormente. O modelo genérico da fase de instalação pode ser melhor visualizado através da Figura 24.

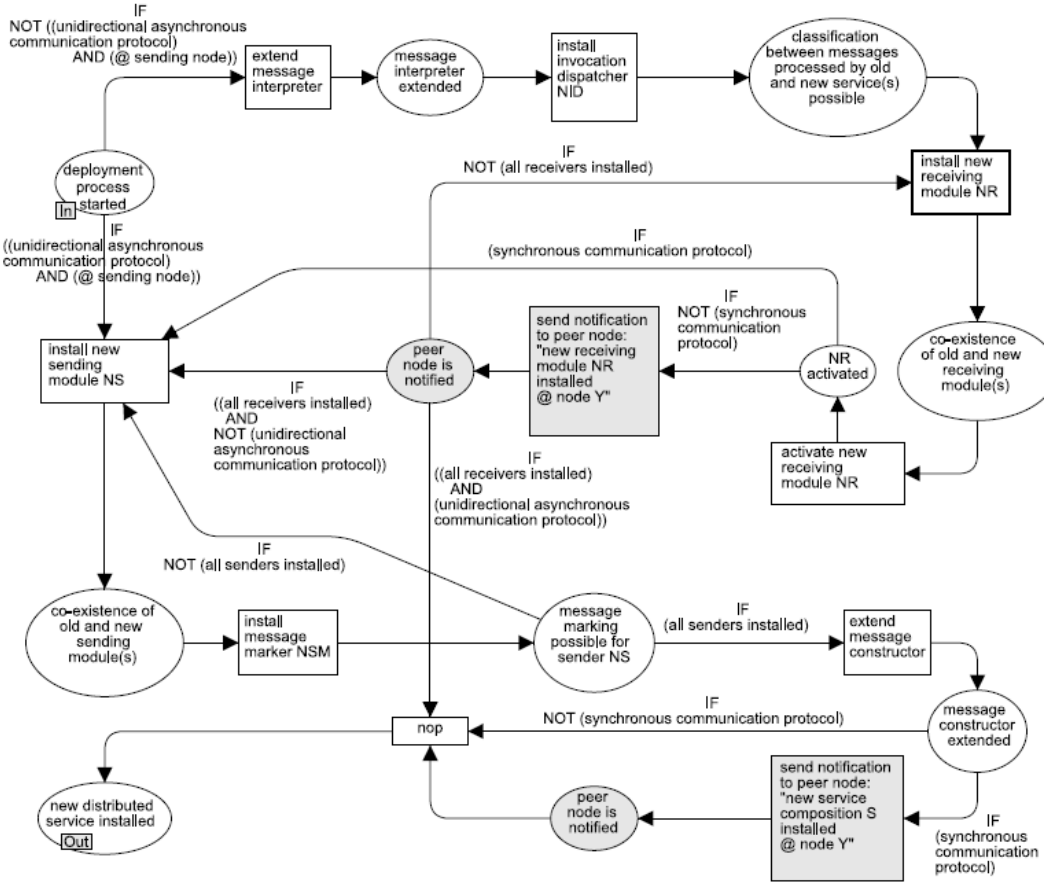
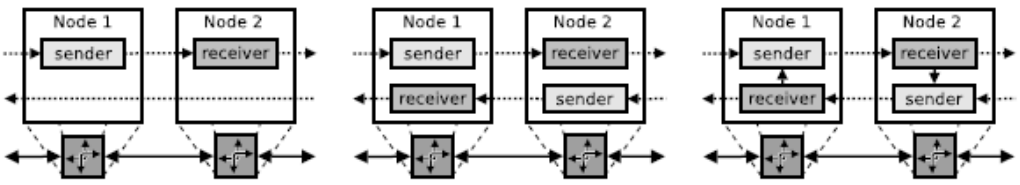


Figura 24 – Modelo genérico do processo de instalação de serviços.

6.4.2.1.Modelo de Comunicação do Serviço

O modelo de comunicação utilizado pelo par de entidades colaborativas ponto-a-ponto pode ser de três tipos: *assíncrono unidirecional*, assíncrono *bidirecional* e *síncrono*, como mostrado na Figura 25. Dependendo da escolha feita quanto a essa característica, a fase de instalação de um novo serviço e a fase de remoção de um antigo serviço podem ser personalizadas.



(a) Asynchronous unidirectional communication (b) Asynchronous bidirectional communication (c) Synchronous communication

Figura 25 – Modelos de comunicação entre os nós de um serviço.

PUC-Rio - Certificação Digital Nº 0611910/CA

O modelo de comunicação *assíncrona unidirecional* (Figura 25-a) apenas permite a comunicação no canal *upstream* ou *downstream* entre os nós envolvidos na comunicação. Tal modelo de comunicação é implementado por um módulo transmissor e um módulo receptor, cada um localizado em nós distintos. Como exemplo, pode-se imaginar que caso a qualidade do sinal de um canal *upstream* de uma rede sem fio tenha sido reduzida, poderia substituir o módulo de compressão atual por um módulo de compressão de dados mais eficiente.

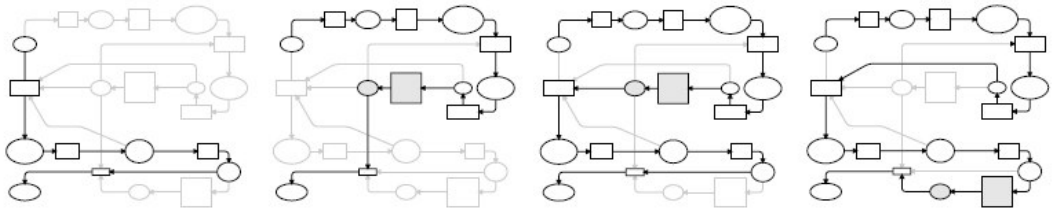
Durante a fase de instalação, a instância do NeCoMan que controla o nó remetente instala o novo módulo de compressão assim como o mecanismo de suporte para marcar mensagens que serão comprimidas pelo novo módulo (Figura 26-b). Simultaneamente, no nó destinatário o novo módulo de descompressão e o suporte para delegar mensagens marcadas para o módulo apropriado de descompressão serão adicionados (Figura 26-c). Após todos os módulos terem sido instalados, o módulo receptor envia uma mensagem de sincronização para o outro nó, iniciando a ativação de um novo serviço de compressão. É importante ressaltar que apenas o nó remetente executará o processo de ativação, como pode ser visto na parte de baixo da Figura 23.

No modelo de comunicação *assíncrona bidirecional* (Figura 25-b), o canal *upstream* e o canal *downstream* entre os dois nós envolvidos na comunicação utilizam o mesmo serviço, porém como tais canais são independentes entre sí, os serviços executados em cada um dos canais também são independentes entre sí. Portanto, a instância do NeCoMan em um nó irá controlar a adaptação da parte remetente do canal *downstream* e a parte destinatária do canal *upstream*. Assim, a fase de instalação do serviço é idêntica em ambos os nós (Figura 26-d). Da mesma forma, como o modelo de comunicação assíncrona unidirecional, após concluir a instalação no módulo receptor, uma mensagem de sincronização deverá ser enviada para o outro nó para a ativação do serviço.

Por fim, alguns serviços de rede seguem o modelo de comunicação *síncrona* (Figura 25-c). Um exemplo desse modelo de comunicação seria um serviço do tipo TCP. Nesse caso, tanto o canal *downstream*, representado por pacotes de dados, como o canal *upstream*, representado por confirmações (*acknowledgements*) fazem parte do mesmo serviço, o que implica na dependência entre o módulo transmissor e o módulo receptor localizados no mesmo nó.

PUC-Rio - Certificação Digital Nº 0611910/CA

Consequentemente, ambos, o novo módulo transmissor e receptor serão instalados antes da sincronização com o outro nó para ativar o novo serviço (Figura 26-e).



(b) Asynchronous unidirectional service at the sending node (c) Asynchronous unidirectional service at the receiving node (d) Asynchronous bidirectional service (e) Synchronous service

Figura 26 – Instâncias personalizadas do modelo genérico da instalação de um novo serviço.

6.4.2.2.Embaralhamento de Pacotes

A escolha feita quanto a essa característica – importar-se ou não com a ordenação dos pacotes – irá influenciar somente na personalização da fase de ativação de um novo serviço, que envolve a ordem da ativação de um novo módulo transmissor e a finalização de sua antiga versão.

Se a rede permite que os pacotes sejam entregues fora de ordem, os fluxos de pacotes transmitidos podem ser direcionados para o novo módulo transmissor enquanto, simultaneamente, o antigo módulo transmissor é finalizado. Porém, caso a rede não permita que os pacotes sejam entregues fora de ordem, o antigo módulo transmissor deve ser finalizado antes que a nova versão desse módulo seja ativada.

6.4.2.3.Transferência de Estado

Quando o embaralhamento de pacotes não é permitido, a *transferência de estado* entre o antigo (congelado) módulo de envio e o novo (ainda não ativado) pode acelerar a fase de ativação de um novo serviço. Por exemplo, a substituição de um serviço do tipo TCP por uma nova versão do mesmo serviço, após o antigo ter completado a configuração da sessão através do *3-way handshake*. Ao invés de aguardar cada um dos nós envolvidos alcançar o estado esperado *CLOSED* antes da ativação do novo serviço, o processo de adaptação pode ser agilizado com a interrupção (*freezing*) do serviço atual substituindo-o pelo novo serviço. Entretanto, para preservar a validade de um serviço, esse procedimento implica a

PUC-Rio - Certificação Digital Nº 0611910/CA

transferência do estado interno do antigo serviço de TCP para o novo serviço, antes que o novo seja ativado. Consequentemente, o processo de ativação do serviço será personalizado dependendo da: capacidade das camadas mais baixas da arquitetura do nó programável em capturar e restabelecer o estado do protocolo em tempo de execução; e da semântica do serviço de rede (com estado ou sem estado).

6.5.Estudo Comparativo

Esta seção descreve cada uma das soluções, incluindo o Moratus/SACS, com base nos critérios selecionados, e ao final apresenta uma tabela para facilitar a compreensão das características de cada trabalho de acordo com os critérios de comparação utilizados.

6.5.1.Transparência da Adaptação

O NeCoMan, o Protocolo de Adaptação Graciosa e o Moratus são os trabalhos que executam adaptações de forma transparente para a aplicação.

Nesses três trabalhos, a arquitetura de um serviço é composta por um módulo transmissor e um módulo receptor. Tal arquitetura é fundamental para que a substituição de um serviço no processo adaptativo seja imperceptível para a aplicação, que pode continuar trocando mensagens normalmente durante o processo, conforme é descrito nas seções 3.1.2, 6.4.1 e 6.3.2.3. Portanto, pode-se classificar esses trabalhos como soluções que executam adaptações transparentes.

Já o processo de adaptação no Ensemble interrompe a aplicação para que o protocolo de adaptação (PTP) substitua as antigas pilhas de protocolos pelas novas. Somente após completar a substituição das pilhas em todos os nós é que o sistema volta a operar para oferecer novamente seus serviços à aplicação. Portanto, pode-se classificar o Ensemble como uma solução que *não* executa adaptações transparentes.

6.5.2.Tratamento da Desconexão

O Moratus e o Ensemble são as únicas soluções preparadas para tratar períodos de desconexão ou falha de dispositivos. Tanto o NeCoMan como o

PUC-Rio - Certificação Digital Nº 0611910/CA

Protocolo de Adaptação Graciosa não apresentam mecanismos para tratamento de desconexão, apenas reconhecem tal limitação e mencionam que esses mecanismos ainda serão implementados.

O tratamento para desconexão no Moratus garante que os membros do grupo não esperem indefinidamente por mensagens de um membro desconectado. Na etapa 1, caso o membro do grupo se desconecte, todos os demais membros tratarão tal desconexão e determinarão de qualquer forma o contexto global. Na etapa 2, caso o participante se desconecte, o mesmo processo é aplicado, os membros tratam a desconexão e o processo adaptativo continua. Porém se o desconectado for o coordenador, escolhe-se um outro dispositivo para assumir esse papel (usando a informação sobre a visão do grupo) e a etapa é reiniciada.

O tratamento para desconexão no Ensemble também foi desenvolvido para que membros de um grupo não esperem indefinidamente por mensagens de um membro desconectado. Caso um membro do grupo se desconecte, o coordenador do grupo envia uma mensagem para o grupo notificando a desconexão de um membro, e o membro desconectado é simplesmente ignorado. Caso o coordenador se desconecte, um membro do grupo assume o papel de coordenador, envia uma mensagem para o grupo informando a desconexão do antigo coordenador e assume o controle da adaptação coordenada.

6.5.3.Composição de Pilhas de Protocolos

O Ensemble é a única solução de fato que permite a composição de pilhas de protocolos, o que lhe dá uma maior flexibilidade na escolha funcionalidade de processamento de mensagens a ser substituída.

O Moratus não considera a existência de pilhas de protocolos, apenas serviços isolados, da mesma forma como é feito no NeCoMan.

O Protocolo de Adaptação Graciosa não permite que CAs possam ser conectados diretamente formando um pilha de protocolos, porém sua arquitetura prevê a existência de três camadas que acomodam CAs que tenham funcionalidades diferentes.

No Ensemble, é possível que o desenvolvedor explicitamente conecte (e substitua) diversos serviços, chamados de micro-protocolos, organizados em uma pilha de protocolos em cada entidade.

PUC-Rio - Certificação Digital Nº 0611910/CA

6.5.4.Coordenação da Adaptação Distribuída

O NeCoMan é uma solução voltada apenas para a coordenação da adaptação do serviço entre dois dispositivos, tornando o processo de coordenação da adaptação de serviços o menos complexo dos três trabalhos.

Já o Protocolo de Adaptação Graciosa é uma solução voltada para coordenar adaptação de MMAs em grupos de dispositivos; o Ensemble está preparado para coordenar a adaptação das pilhas de protocolos em diversos dispositivos; e o Moratus está preparado para executar adaptação de serviços de transformação de mensagens em grupos de dispositivos.

A Tabela 7 resume as características dos trabalhos estudados e do Moratus, mostrando que o último e o Ensembre são os únicos trabalhos que trataram a questão da coordenação da adaptação nos membros de um grupo. Enquanto a ênfase do Ensembre está na flexibilidade da adaptação, que não é o foco do Moratus, este último provê transparência de adaptação para a aplicação, o que é particularmente importante para adaptações dinâmicas.

PUC-Rio - Certificação Digital Nº 0611910/CA

Solução	Transparência da Adaptação	Tratamento da Desconexão	Composição de Pilhas de Serviços	Coordenação da Adaptação Distribuída
NeCoMan	Adaptação Transparente	Não	Não	Sim, em pares
Protocolo de Adaptação Graciosa	Adaptação Transparente	Não	Não	Sim, em grupos
Ensemble	Adaptação Não-Transparente	Sim	Sim	Sim, em grupos
Moratus	Adaptação Transparente	Sim	Não	Sim, em grupos

Tabela 7 – Comparação entre os trabalhos relacionados ao Moratus.