



**República Federativa do Brasil**  
Ministério da Indústria, Comércio Exterior  
e Serviços  
Instituto Nacional da Propriedade Industrial

**(11) PI 0505512-1 B1**

**(22) Data do Depósito:** 14/12/2005

**(45) Data de Concessão:** 10/10/2017



\* B R P I 0 5 0 5 5 1 2 B 1 \*

---

**(54) Título:** MÉTODO E SISTEMA PARA TRANSFERIR ATUALIZAÇÕES

**(51) Int.Cl.:** G06F 9/44

**(52) CPC:** G06F 8/68

**(30) Prioridade Unionista:** 14/12/2004 US 11/012.367

**(73) Titular(es):** MICROSOFT TECHNOLOGY LICENSING, LLC

**(72) Inventor(es):** JASON F. MACKAY

## "MÉTODO E SISTEMA PARA TRANSFERIR ATUALIZAÇÕES"

### CAMPO DA INVENÇÃO

Em geral, a presente invenção relaciona-se com atualizações de computador e, em particular, com um sistema e método para obter e instalar atualizações utilizando um ambiente de rede não-hierárquica.

### FUNDAMENTO DA INVENÇÃO

Computadores, e em particular, software de computador, são freqüentemente atualizados por um usuário transferindo uma correção a partir de um computador central, tal como o provedor de software, via uma rede, tal como a Internet. Atualmente, as correções são transferidas em sua totalidade a partir de uma única fonte e instaladas na máquina que está transferindo para atualizar o software para um estado atual. Enquanto esta técnica é operacional, ela possui várias desvantagens.

Em primeiro lugar, o provedor de software que torna a correção disponível para transferência deve proporcionar uma quantidade suficiente de largura de banda de saída para permitir que vários clientes transfiram atualizações ao mesmo tempo. Para provedores de software com um grande número de clientes, tal largura de banda de saída pode se tornar muito cara.

Em segundo lugar, um cliente deve transferir toda a correção a partir de uma fonte antes de instalar a correção e atualizar o software. Enquanto que isto pode ser aceitável para correções de pequeno porte, para uma correção de porte maior, o tempo necessário para transferir toda a

atualização a partir de uma fonte pode ser inaceitável. Em terceiro lugar, se o software proporcionado não tiver feito largura de banda ampla disponível para as transferências (particularmente difícil e cara quando uma correção de alta  
5 prioridade é liberada, tal como uma atualização de segurança), um cliente pode ser temporariamente proibido de transferir a correção.

Avanços recentes na rede não-hierárquica proporcionaram a capacidade de transferir partes diferentes de um  
10 arquivo grande a partir de uma pluralidade de pares. Por exemplo, a FIGURA 1 ilustra um diagrama de blocos de um ambiente não-hierárquico típico para transferir um arquivo específico. Para transferir um arquivo específico, um cliente  
101 requisita o arquivo específico a partir de um computador  
15 central 103. Em resposta, o computador central transmite uma lista de pares identificando uma pluralidade de pares que tornam disponíveis para transferência diferentes partes destes arquivos requisitados. Por exemplo, um arquivo de vídeo pode ser subdividido em uma pluralidade de partes, es-  
20 tas partes divididas em sub-partes e estas sub-partes divididas em sub-sub-partes. Cada par que transferiu anteriormente o arquivo de vídeo, a partir de uma fonte ou a partir de vários outros pares, pode tornar disponível para transferência por outros pares, tal como o cliente 101, uma ou mais  
25 destas sub-sub-partes de todo o arquivo. A lista de pares transmitida pelo hospedeiro 103 para o cliente 101 identifica cada uma das sub-sub-partes do arquivo específico requisitado e um par associado que torna disponível para transfe-

rência estas sub-sub-partes.

O cliente 101, quando da recepção da lista de pares a partir do computador central 103, escolhe um par para cada uma das sub-sub-partes a partir do qual ele irá transferir esta sub-sub-parte. Quando da identificação dos pares, o cliente 101 transfere cada uma das sub-sub-partes a partir dos pares selecionados até todas as sub-sub-partes do arquivo específico terem sido transferidas. Por transferir partes a partir de diferentes pares, várias transferências podem ser conduzidas simultaneamente. A medida que cada sub-sub-parte é transferida, o cliente 101 pode decidir se ele deseja ou não se tornar um par que torna disponível para transferência esta sub-sub-parte (isto é, compartilhar esta sub-sub-parte). Se o cliente decidir que ele deseja compartilhar esta sub-sub-parte, o cliente 101 identifica para o computador central 103 a sub-sub-parte que ele está disposto a compartilhar.

Enquanto que os ambientes não-hierárquicos, tal como este descrito com respeito à FIGURA 1, proporcionam para um cliente a capacidade de transferir um arquivo específico que ele requisita, tal como um arquivo de vídeo, atualmente não existe qualquer técnica para a determinação de quais arquivos um cliente pode precisar. Por exemplo, referindo-se de volta à transferência de correções de software, antes de requisitar uma correção, um cliente deve saber não somente quais atualizações estão disponíveis, mas também quais atualizações são necessárias para o cliente. Ainda adicionalmente, atualmente não existe qualquer aspecto de

segurança para proteger um computador que está transferindo de ser comprometido por identificar o que ele está compartilhando antes deste material ser totalmente integrado no computador que está transferindo. Adicionalmente, não existe  
5 um método pelo qual várias correções na memória cache local do cliente possam ser priorizadas pelo provedor de software quando o cliente precisa recuperar espaço de memória cache local.

Por consequência, existe a necessidade de um sistema e método que permitam a um cliente identificar atualizações de software necessárias e transferir de forma segura estas atualizações utilizando um ambiente não-hierárquico.  
10

#### SUMÁRIO DA INVENÇÃO

As modalidades da presente invenção proporcionam para um provedor de software a capacidade de distribuir atualizações de software para vários clientes diferentes utilizando um ambiente não-hierárquico. A invenção descrita aqui pode ser utilizada para atualizar qualquer tipo de software, incluindo, mas não limitado ao software operacional, software  
15 re de programação, software anti-vírus, software de base de dados, etc. O uso de um ambiente não-hierárquico com segurança adicionada proporciona a capacidade de minimizar o tempo de transferência para cada par e também reduzir a quantidade de largura de banda de saída que deve ser proporcionada pelo provedor de software para permitir aos seus  
20 clientes (pares) obterem a atualização.

De acordo com um primeiro aspecto da presente invenção, é proporcionado um método para atualizar um primeiro

dispositivo de computação. O método recebe uma lista de atualizações disponíveis e determina se são necessárias algumas das atualizações na lista. Se forem necessárias algumas das atualizações, é recebida uma lista de dispositivos de computação que identifica partes da atualização e dispositivos de computação a partir dos quais estas partes podem ser obtidas. Estas partes são então transferidas a partir dos dispositivos de computação, onde pelo menos duas da pluralidade de partes são transferidas a partir de dispositivos de computação diferentes. Uma vez que as partes tenham sido transferidas, o software do primeiro dispositivo de computação é atualizado utilizando as partes transferidas.

De acordo com outro aspecto da presente invenção, é proporcionado um método para fornecer uma lista de pares para um cliente para transferir atualizações de software. O método transmite uma lista identificando uma atualização disponível e recebe uma requisição de lista de pares para a atualização disponível. Em resposta, é gerada uma lista de pares identificando uma pluralidade de pares, onde cada par identificado torna disponível para transferência pelo menos uma parte da atualização disponível. Esta lista de pares é então transmitida.

De acordo com outro aspecto da presente invenção, é proporcionado um sistema de computador possuindo um meio legível por computador incluindo um programa executável por computador no mesmo para executar o método de obter uma atualização. O sistema de computador recebe uma lista de atualizações disponíveis e identifica uma atualização necessária

a partir da lista de atualizações disponíveis. Uma lista de pares é então requisitada e recebida para a atualização necessária. Uma lista de pares recebida identifica uma pluralidade de partes da atualização necessária e identifica para  
5 cada parte, um ou mais dispositivos de computação que tornam esta parte disponível para transferência. O sistema de computador então transfere cada uma das partes a partir dos dispositivos de computação identificados.

#### BREVE DESCRIÇÃO DOS DESENHOS

10 Os aspectos precedentes e várias das vantagens acompanhantes desta invenção se tornarão mais prontamente apreciados a medida que os mesmos se tornam melhor entendidos por referência à descrição detalhada seguinte, quando considerada em conjunto com os desenhos acompanhantes, nos quais:

15 A FIGURA 1 ilustra um diagrama de blocos de um ambiente não-hierárquico típico para transferir um arquivo específico;

As FIGURAS 2A a 2C ilustram um diagrama de estado para transferir uma atualização de software utilizando um  
20 ambiente não-hierárquico, de acordo com uma modalidade da presente invenção.

A FIGURA 3 é um diagrama de blocos de uma lista de opções de atualização identificando opções diferentes para atualizações necessárias que podem ser geradas por um compu-  
25 tador central, de acordo com uma modalidade da presente invenção;

A FIGURA 4 ilustra um diagrama de blocos de uma lista de pares identificando pares para uma atualização ne-

cessária em resposta a uma requisição por uma lista de pares a partir de um cliente, de acordo com uma modalidade da presente invenção;

A FIGURA 5 é um fluxograma que ilustra uma rotina de atualização de software do cliente para atualizar o software de um cliente, de acordo com uma modalidade da presente invenção;

A FIGURA 6 ilustra um fluxograma de uma sub-rotina de transferência que pode ser executada como parte de uma rotina de atualização de software do cliente para transferir partes de uma atualização necessária a partir de diferentes pares, de acordo com uma modalidade da presente invenção;

A FIGURA 7 ilustra um fluxograma de uma sub-rotina de prioridade para expandir o tamanho da memória cache em um sistema cliente para permitir uma transferência de uma correção, de acordo com uma modalidade da presente invenção; e

A FIGURA 8 ilustra uma rotina do computador central para tornar disponíveis atualizações utilizando um ambiente não-hierárquico, de acordo com uma modalidade da presente invenção.

#### DESCRIÇÃO DETALHADA DA MODALIDADE PREFERIDA

As modalidades da presente invenção proporcionam para um provedor de software, referido aqui como um computador central, a capacidade de distribuir partes de atualizações de software para vários recipientes diferentes (referidos aqui como pares). Um par, como referido aqui, inclui qualquer tipo de dispositivo de computação que torna disponível para transferência uma parte (fragmento) de uma atua-

lização de software. Por exemplo, um par pode ser, mas não está limitado à um dispositivo de computação individual, um servidor, um dispositivo de computação central, um dispositivo de computação cliente, etc. Pares com partes de uma  
5 atualização de software tornam esta parte disponível para transferência para outros pares. A medida que pares adicionais obtêm as mesmas partes do software, eles também tornam estas partes disponíveis para transferência por outros pares. Assim, a medida que o número de pares contendo uma  
10 parte aumenta, o número de locais de transferência disponíveis para esta parte aumenta.

As FIGURA 2A a 2C ilustram um diagrama de estado para transferir uma atualização de software utilizando um ambiente não-hierárquico de acordo com uma modalidade da  
15 presente invenção. O ambiente não-hierárquico 200 proporciona para um cliente 201 a capacidade de identificar atualizações necessárias e obter uma lista de pares a partir de um computador central 203 a partir do qual ele pode obter partes destas atualizações.

20 Referindo-se primeiro à FIGURA 2A, em um estado inicial, o cliente 201 requisita a partir de um computador central uma lista de atualizações que estão disponíveis a partir do computador central 203. Em uma modalidade alternativa, o computador central 203 pode, de tempo em tempo,  
25 publicar uma lista de atualizações disponíveis que é obtida por um cliente 201. O computador central 203, quando da recepção de uma requisição por uma lista de atualizações disponíveis, envia a lista de atualizações disponíveis para o

cliente 201. O cliente 201 varre-se para determinar seu estado atual e compara este estado com a lista de atualizações disponíveis para determinar quais atualizações são necessárias. Quando de uma determinação das atualizações necessárias, o cliente 201 requisita uma lista de opções de atualização para as atualizações necessárias a partir do computador central 203. O computador central 203, quando da recepção de uma requisição por opções de atualização para atualizações necessárias, constrói uma lista de opções de atualização.

Por razões de privacidade de um cliente, em uma modalidade da presente invenção, uma quantidade mínima de informação do cliente é passada a partir do cliente para o computador central. Por exemplo, como recém discutido, o computador central proporciona uma lista de atualizações disponíveis e o cliente decide quais são necessárias. Em uma modalidade alternativa, um cliente poderia proporcionar para um computador central uma identificação de seu estado atual, ou a informação correspondendo aos vários estados de operação, e o computador central poderia informar ao cliente quanto às atualizações necessárias.

Referindo-se à FIGURA 3, é descrita uma lista de opções de atualização identificando diferentes opções para atualizações necessárias que pode ser gerada por um computador central, de acordo com uma modalidade da presente invenção. A lista de opções de atualização 300 identifica, para cada atualização requisitada pelo cliente 201, tal como a ATUALIZAÇÃO 1 301, as diferentes opções disponíveis para ob-

ter esta atualização. Adicionalmente, a lista de opções de atualização 300 pode incluir opções de atualização para várias atualizações requisitadas. Por exemplo, assumindo que o cliente 201 requisitou opções de atualização para ATUALI-  
5 ZAÇÃO 1 301 e ATUALIZAÇÃO 7 303, a lista de opções de atualização 300 incluiria uma identificação de cada uma das atualizações requisitadas 301, 303 e uma identificação das diferentes opções disponíveis para obter esta atualização. Por exemplo, a ATUALIZAÇÃO 1 301 pode incluir uma identifi-  
10 cação de uma Correção Completa 1 305, de uma correção Delta 1.1 307, de uma correção Delta 1.2 309 e de uma correção Delta 1.3 311. De modo semelhante, a ATUALIZAÇÃO 7 303 pode incluir uma identificação de uma Correção Completa 7 313, de uma correção Delta 7.1 315, de uma correção Delta 7.2 317 e  
15 de uma correção Delta 7.3 319.

A Correção Completa 1 305 é uma atualização completa que pode ser instalada para levar o software a partir de um estado existente para um estado atual. Cada correção delta pode ser utilizada para levar o software a partir de  
20 um estado existente conhecido para um estado atual sem ter que utilizar a correção completa. O software que deve ser atualizado freqüentemente está em um dentre vários estados diferentes conhecidos que podem ser identificados. Assim, para cada um destes estados conhecidos, uma correção delta  
25 pode ser utilizada para modificar este software para levá-lo para um estado atual. Por exemplo, a correção Delta 1.1 307 pode ser utilizada para levar uma versão do software em um estado existente conhecido para um estado atual.

De modo a permitir ao cliente 201 determinar se ele pode utilizar uma das correções delta para atualizar o software, cada correção delta inclui uma lista de arquivos e um valor de hash esperado para estes arquivos. O valor hash esperado pode ser utilizado para determinar se a correção delta é apropriada para o software existente. Por exemplo, a correção Delta 1.1 307 inclui um Arquivo 1.1 320 com um valor hash esperado 321, um Arquivo 1.2 322 com um valor hash esperado 323 e um Arquivo 1.3 324 com um valor hash esperado 325. Um cliente, quando recebendo a lista de opções de atualização 300, pode produzir um código hash para o estado atual dos arquivos que devem ser atualizados no sistema do cliente 201 e determinar se os valores hash se associam com os valores hash esperados contidos na lista de atualizações. Se os valores hash se associarem, então a correção delta correspondente pode ser utilizada para atualizar o sistema do cliente. Utilizar uma correção delta, tal como Delta 1.1 307, ao invés de utilizar uma correção completa, tal como a Correção Completa 1 305, resulta em uma transferência menor porque ela é uma atualização parcial ao invés de uma atualização completa.

Referindo-se agora à FIGURA 2B, uma vez que o computador central 203 tenha gerado a lista de opções de atualização requisitada, esta lista de opções de atualização é transmitida para o cliente 201. O cliente 201 seleciona a partir da lista de opções de atualização a correção para cada atualização que trará o estado do cliente 201 para um estado atual. Como discutido acima, o cliente pode selecionar

uma correção completa ou uma correção delta apropriada para as atualizações que são necessárias. Como será apreciado pelos com conhecimento na técnica relevante, uma ou mais atualizações necessárias podem ser selecionadas a partir da  
5 lista de opções de atualização. Quando da seleção de cada uma das correções apropriadas, o cliente 201 requisita uma lista de pares para as atualizações necessárias. O computador central 203, em resposta a receber uma requisição por uma lista de pares para atualizações necessárias, constrói a  
10 lista de pares requisitada e transmite esta lista de pares para o cliente 201.

O computador central mantém uma lista de pares mestre que identifica cada par que torna partes de uma atualização disponíveis para transferência. Se uma atualização  
15 nunca foi transferida a partir do computador central, uma lista de pares inicial somente pode identificar o computador central como sendo um par disponível. Em tal situação, o cliente seria o primeiro a transferir a atualização e selecionaria o computador central como o par e transferiria todo  
20 o arquivo a partir do computador central. Como discutido abaixo, uma vez que o cliente tenha transferido e instalado a atualização, ele identifica para o computador servidor que ele está tornando disponível para transferência (compartilhando) uma ou mais das partes da atualização. Assim, o  
25 computador central adicionaria este cliente à lista de pares mestre para as partes da atualização que o cliente irá compartilhar. A medida que clientes adicionais transferem a atualização a partir do computador central, do par, ou de

ambos, e se identificam como agora tornando partes da atualização disponíveis para transferência, o número disponível de pares cresce.

A FIGURA 4 ilustra um diagrama de blocos de uma  
5 lista de pares identificando pares para uma atualização necessária em resposta à uma requisição por uma lista de pares a partir de um cliente, de acordo com uma modalidade da presente invenção. Continuando com o exemplo acima, assumindo que um cliente 201 requisitou a correção Delta 1.1 401 e a  
10 Correção Completa 7 403, a lista de pares 400 incluiria detalhes de cada uma destas correções e a identificação dos pares para estas correções. Os valores de hash também são incluídos para verificar a autenticidade das correções transferidas.

15 Por exemplo, a correção Delta 1.1 401 inclui um valor de hash delta e de prioridade para esta correção. Como será discutido em maiores detalhes abaixo, o valor hash para a correção pode ser utilizado para confirmar a validade do hash antes da instalação. Em uma modalidade alternativa,  
20 o valor de prioridade não pode ser incluído na correção. Em tal modalidade, as partes são salvas na memória cache e a medida que o espaço de armazenamento adicional se torna necessário, o computador cliente pode requisitar uma lista de prioridades para as partes que estão localizadas atualmente  
25 na memória cache. Isto proporciona para o computador central a capacidade de controlar dinamicamente as prioridades das partes com o tempo. Além disso, como discutido abaixo, o valor de prioridade da correção pode ser utilizado para

determinar se partes de uma correção devem ser mantidas na memória cache do computador cliente ou excluídas. Para cada arquivo de uma correção, tal como o Arquivo 1.1 405 e o Arquivo 1.2 407, um hash delta também é incluído. Assim como  
5 o hash delta da correção, o hash delta para o arquivo pode ser utilizado para verificar a autenticidade do arquivo.

Adicionalmente, cada arquivo, tal como o Arquivo 1.1 405, pode ser dividido em partes, tais como Parte 1 409 e Parte 2 411. Cada parte de um arquivo pode adicionalmente  
10 ser dividida em sub-partes, e estas sub-partes divididas em sub-sub-partes. Por exemplo, a Parte 1 409 pode ser dividida em sub-partes, tais como a Sub-Parte 1.1 413 e a Sub-Parte 1.2 415. De modo semelhante, estas sub-partes podem ser adicionalmente divididas em sub-sub-partes, tais como a  
15 Sub-Sub-Parte 1.1.1 417 e a Sub-Sub-Parte 1.1.2 419. Como será apreciado pelos com conhecimento na técnica relevante, a divisão em partes, sub-partes e sub-sub-partes pode ser utilizada até qualquer nível de divisão, e a granularidade para sub-sub-parte, como descrita aqui dentro, é somente pa-  
20 ra propósitos de explicação.

O menor nível de divisão, tal como uma sub-sub-parte, inclui uma identificação de um ou mais pares que tornam esta parte disponível para transferência. Por exemplo, a Sub-Sub-Parte 417 inclui uma identificação dos dois pares  
25 que tornam esta parte disponível para transferência. Em particular, a Sub-Sub-Parte 1.1.1 417 pode ser transferida a partir do Par 1 421 ou do Par 4 425. Cada par que torna uma sub-sub-parte disponível para transferência é identificado

por um endereço, tal como um endereço do protocolo Internet. A lista de pares 400 pode incluir uma identificação de todos os pares que tornam uma sub-sub-parte disponível para transferência, de uma parte dos pares que torna uma sub-sub-parte disponível para transferência ou de um par para cada sub-sub-parte. 5 Adicionalmente, para cada par listado na lista de pares, podem ser proporcionadas informações adicionais. Por exemplo, informações adicionais a cerca de um par podem incluir, mas não estão limitadas à largura de banda do par, 10 à classificação do par (o quão confiável é o par), às afiliações do par com companhias ou organizações, ao histórico de uso dos pares do sistema de transferência não-hierárquico, etc. A lista de pares também pode incluir um conjunto de comandos de atualização 427 que proporciona instruções a 15 cerca de como instalar a própria correção delta.

Referindo-se de volta à FIGURA 2B, o cliente 201, quando da recepção de uma lista de pares, seleciona pares para as sub-sub-partes identificadas na lista de pares. A seleção de pares de partes será descrita em maiores detalhes 20 abaixo com respeito à FIGURA 6. Quando da seleção das sub-sub-partes e dos pares a partir dos quais estas partes devem ser transferidas, o cliente 201 abre vários canais de comunicações com os diferentes pares e começa a transferir as diferentes partes das atualizações simultaneamente. A técnica 25 de transferir diferentes partes de uma atualização a partir de diferentes pares aumenta a velocidade com a qual uma atualização pode ser obtida e deste modo diminui o tempo de transferência geral. Adicionalmente, proporcionar atua-

lizações via um ambiente não-hierárquico reduz a largura de banda de saída necessária pelo computador central 203. Utilizando as modalidades da presente invenção, a medida que os clientes entram em contato com o computador central, transferem uma atualização e começam a compartilhar partes destas atualizações com outros clientes (isto é, outros pares), a largura de banda de saída para fora do computador central 203 reduz a medida que os clientes estão obtendo partes da atualização a partir de outros pares ao invés do computador central. A técnica lida com aumentos extremos na demanda por conteúdo específico de forma bem mais eficaz do que o fornecimento de arquivos tradicional, o qual pode na melhor das hipóteses se tornar muito caro e na pior das hipóteses falhar totalmente sob a carga.

Referindo-se agora à FIGURA 2C, a medida que a transferência de cada sub-sub-parte é completada, esta sub-sub-parte sofre uma transformação numérica e comparada com o valor hash incluído na lista de pares para confirmar que a sub-sub-parte não foi alterada. Uma vez que todas as sub-sub-partes para uma correção particular tiverem sido transferidas e autenticadas, estas partes são montadas para recriar a correção. Adicionalmente, uma cópia de uma ou mais das sub-sub-partes transferidas é mantida em uma memória cache. Estas cópias podem se tornar disponíveis para transferência por outros pares.

A correção montada é instalada para atualizar o cliente para um estado atual. Quando da conclusão da instalação, o sistema do computador cliente pode ser reiniciali-

zado, se necessário. Uma vez que a instalação das correções e a reinicialização do computador cliente 201 (se necessária) estiverem completas, o cliente 201 transmite para o computador central 203 uma identificação das sub-sub-partes que são armazenadas na memória cache. Tal identificação pode ser um valor hash de cada sub-sub-parte armazenada. O computador central 203, quando da recepção de uma identificação de sub-sub-partes armazenadas em memória cache, atualiza a lista de pares mestre adicionando o cliente como um par que torna estas partes disponíveis para transferência. A lista de pares mestre inclui uma lista de cada par que mantém uma parte de uma correção que é feita disponível para transferência para outros clientes.

A FIGURA 5 é um fluxograma que ilustra uma rotina de atualização de software do cliente para atualizar o software de um cliente, de acordo com uma modalidade da presente invenção. A rotina de atualização de software do cliente 500 começa no bloco 501 e no bloco 503 o cliente requisita a obtém uma lista de atualizações. Como discutido acima, a lista de atualizações inclui uma identificação das atualizações que estão disponíveis para transferência. Utilizando esta lista, no bloco 505, um cliente compara a condição de seu sistema existente com as atualizações disponíveis. Baseado nesta comparação, no bloco de decisão 507, é feita uma determinação quando a se são necessárias atualizações para levar o sistema existente do cliente para um estado atual.

Se for determinado no bloco de decisão 507 que são necessárias atualizações, no bloco de decisão 509, é feita

uma determinação quanto a se a correção para a atualização necessária está disponível. Como discutido acima, uma determinação quanto a se uma correção delta está disponível para uma atualização necessária pode ser averiguada por requisitar opções de atualização para uma atualização necessária e receber uma lista de diferentes opções de atualização. A lista de diferentes opções de atualização inclui uma identificação das correções delta e dos valores hash esperados para os arquivos que serão atualizados pela correção delta.

10 Por comparar os valores hash esperados da correção delta com os valores hash dos arquivos no cliente, pode ser feita uma determinação pelo cliente quanto a se uma correção delta está disponível. Em uma modalidade alternativa, a lista de atualizações disponíveis também pode incluir opções de atualização para cada atualização disponível, deste modo removendo a necessidade de subsequente obter opções de atualização.

Permitir a um cliente determinar se uma correção delta está disponível, em contraste com o cliente proporcionando valores hash de seu sistema existente para um computador central, permite ao cliente manter privacidade com respeito a si mesmo. Se for determinado no bloco de decisão 509 que uma correção delta está disponível, esta correção delta é adicionada à uma lista de transferências, como ilustrado pelo bloco 511. Entretanto, se for determinado no bloco de decisão 509 que uma correção delta não está disponível, no bloco 513 uma identificação de uma correção completa para a atualização é adicionada à lista de transferên-

cias.

No bloco de decisão 515, é feita uma determinação quando a se são necessárias atualizações adicionais para levar o cliente para um estado atual. Se for determinado no  
5 bloco de decisão 515 que são necessárias atualizações adicionais, a rotina de atualização de software do cliente 500 retorna para o bloco de decisão 509 e a rotina continua. Entretanto, se for determinado no bloco de decisão 515 que não são necessárias atualizações adicionais, no bloco 517 o  
10 cliente requisita uma lista de pares para as correções identificadas na lista de transferências. Em resposta à requisitar uma lista de pares no bloco 517, no bloco 519, o cliente recebe uma lista de pares que inclui uma identificação dos pares que tornam disponível para transferência cada sub-  
15 sub-parte de cada correção identificada na lista de transferências. Quando da recepção de uma lista de pares, a sub-rotina de transferência, ilustrada pelo bloco 521, é executada. A sub-rotina de transferência será descrita em maiores detalhes com respeito à FIGURA 6. Quando da conclusão  
20 da sub-rotina de transferência, no bloco 523, as sub-sub-partes transferidas são montadas para recriar as correções requisitadas e estas correções são instaladas no sistema do cliente para levar o sistema do cliente para um estado atual.

25 Uma vez que a instalação das correções transferidas tiver sido concluída e o sistema do cliente tiver sido reinicializado, se necessário, a rotina de atualização de software do cliente 500 transmite um hash das sub-sub-partes

armazenadas em memória cache, como ilustrado pelo bloco 255. As sub-sub-partes armazenadas em memória cache são estas partes que são armazenadas em uma memória cache de um cliente que o cliente tornará disponível para transferência por outros pares. Em uma modalidade da presente invenção, é exigido que um cliente compartilhe as partes que ele transferiu a partir de outros pares para garantir que o ambiente não-hierárquico seja mantido.

Por exemplo, se um cliente transferir 50 MB de atualizações via um ambiente não-hierárquico, pode ser exigido que o cliente torne disponível para transferência a mesma quantidade de dados que ele próprio transferiu. Em uma modalidade real da presente invenção, o hash das sub-sub-partes armazenadas em memória cache pode não ser transmitido, deste modo identificando que o compartilhamento está disponível, até cada uma das partes ter sido verificada e instalada. Verificar e instalar as correções antes de compartilhar estas correções pode ser necessário porque quando uma lista de pares é proporcionada para outros pares a partir de um computador central, o endereço de um par poderia ser utilizado com o conhecimento da falta de uma correção particular do par, tal como esta sendo transferida, para comprometer a máquina par. Uma vez que o hash das sub-sub-partes armazenadas em memória cache tenha sido transmitida no bloco 525, ou se for determinado no bloco de decisão 507 que nenhuma atualização é necessária, a rotina de atualização de software do cliente 500 termina no bloco 527.

A rotina de atualização de software do cliente 500

pode ser iniciada manualmente, ou na alternativa, programada para executar em uma base periódica para garantir que o cliente seja mantido em um estado atual. Por exemplo, a rotina de atualização de software do cliente 500 pode ser programada para executar ao anoitecer ou quando o dispositivo de computação cliente não estiver sendo utilizado. Adicionalmente, como um aspecto de segurança adicionado, cada uma da lista de atualizações disponíveis, da lista de opções de atualização e da lista de pares pode ser assinada digitalmente pelo computador central de modo que elas não possam ser alteradas. A assinatura digital de documentos é conhecida na técnica e assim não será descrita em detalhes aqui.

A FIGURA 6 ilustra um fluxograma de uma sub-rotina de transferência que pode ser executada como parte de uma rotina de atualização de software do cliente para transferir partes de uma correção a partir de diferentes pares, de acordo com uma modalidade da presente invenção. Enquanto que a sub-rotina de transferência 600 é descrita com respeito à transferência de uma correção, será apreciado que qualquer número de correções podem ser transferidas simultaneamente ou em momentos diferentes utilizando a sub-rotina e transferência 600.

A sub-rotina de transferência 600 começa no bloco 601 e no bloco 603, é recebida uma lista de pares para cada uma das sub-sub-partes de uma ou mais correções. Como discutido acima, a lista de pares identifica os endereços dos pares que tornam disponível para transferência diferentes sub-sub-partes de uma correção. Durante a transferência das

sub-sub-partes, a lista de pares pode ser atualizada periodicamente (por exemplo, a cada dois minutos) para garantir que o cliente está requisitando transferências para pares disponíveis.

5           Utilizando esta lista de pares, no bloco de decisão 605, é feita uma determinação quanto a se será necessário memória cache adicional para armazenar as sub-sub-partes da correção antes da instalação. Se for determinado no bloco de decisão 605 que será necessário memória cache adicional, uma sub-rotina de prioridade é executada, como ilustra-  
10           do pelo bloco 607. A sub-rotina de prioridade para obter memória cache adicional será descrita em maiores detalhes com respeito a FIGURA 7. Entretanto, se for determinado no bloco de decisão 605 que não é necessário memória cache adicional, no bloco 609 um arquivo da correção que está sendo  
15           transferida é selecionado. No bloco 611, é escolhida uma parte do arquivo selecionado para a qual a transferência está para começar. Utilizando esta parte selecionada, no bloco de decisão 613, é feita uma determinação quanto a se uma  
20           das sub-partes desta parte já terminou a transferência. Se for determinado que uma das sub-partes desta parte ainda não terminou a transferência, no bloco de decisão 615 é feita uma determinação quanto a se uma das sub-sub-partes desta sub-parte foi transferida. Se, no bloco de decisão 615, for  
25           determinado que uma das sub-sub-partes de uma sub-parte não foi transferida, no bloco 617, uma sub-sub-parte de uma sub-parte é selecionada aleatoriamente. No bloco 619, um par que torna disponível para transferência a sub-sub-parte se-

leccionada aleatoriamente também é selecionado aleatoriamente. Quando da seleção de uma sub-sub-parte e de um par, no bloco 621, a transferência da sub-sub-parte a partir do par selecionado começa. Uma vez que a transferência da sub-sub-  
5 parte tiver começado, a sub-rotina de transferência 600 retorna para o bloco 613 e continua.

Referindo-se de volta para o bloco de decisão 615, se for determinado que uma das sub-sub-partes terminou a transferência, no bloco 623, é selecionada uma sub-sub-parte  
10 de uma sub-parte que possui pelo menos uma quantidade de pares para os quais esta sub-parte está disponível para transferência. Adicionalmente, no bloco 625, um dos pares para a sub-sub-parte selecionada é escolhido aleatoriamente. No bloco 627, a transferência da sub-sub-parte selecionada a  
15 partir do par escolhido aleatoriamente é iniciada. Uma vez que a transferência da sub-sub-parte selecionada tiver começado, no bloco 627, a sub-rotina de transferência 600 retorna para o bloco 613 e continua.

Devido ao fato de que várias sub-sub-partes podem  
20 ser transferidas ao mesmo tempo, deste modo diminuindo o tempo total necessário para transferir uma correção, ou várias correções, uma vez que a transferência tenha começado no bloco 621 ou no bloco 627, a sub-rotina de transferência 600 continua por escolher outra parte e outro par e iniciar  
25 esta transferência. Assim, uma saturação total da largura de banda para um cliente que está transferindo uma atualização pode ser realizada e a transferência de uma atualização pode ser completada rapidamente. Adicionalmente, a medida

que a transferência de cada sub-sub-parte é concluída, a autenticidade destas partes pode ser verificada por produzir um código hash para a sub-sub-parte e comparar este valor hash com o valor hash incluído na lista de pares.

5               Retornando agora para o bloco de decisão 613, se for determinado que uma sub-sub-parte de uma parte terminou a transferência, no bloco de decisão 629, é feita uma determinação quanto a se todas as partes de um arquivo terminaram a transferência. Se for determinado no bloco de decisão 629  
10   que todas as partes de um arquivo não terminaram a transferência, a sub-rotina de transferência 600 retorna para o bloco 611 e escolhe outra parte e o processo continua. Entretanto, se for determinado no bloco de decisão 629 que todas as partes de um arquivo terminaram a transferência, no  
15   bloco de decisão 631, é feita uma determinação quanto a se todos os arquivos para a correção terminaram a transferência. Se for determinado no bloco de decisão 631 que todos os arquivos para a correção não terminaram a transferência, a sub-rotina de transferência 600 retorna para o bloco 609 e  
20   outro arquivo que está para ser transferido é selecionado. Entretanto, se for determinado no bloco de decisão 631 que todos os arquivos da correção foram transferidos e assim, todas as partes, sub-partes e sub-sub-partes da correção também foram transferidas, a rotina termina no bloco 633 e re-  
25   torna o controle para a rotina de atualização de software do cliente 500 (FIGURA 5).

A FIGURA 7 ilustra um fluxograma de uma sub-rotina de prioridade para expandir o tamanho da memória cache em um

sistema cliente para permitir uma transferência de uma correção, de acordo com uma modalidade da presente invenção. Uma sub-rotina de prioridade 700 começa no bloco 701 e no bloco 703 uma lista de todas as sub-sub-partes armazenadas em memória cache atualmente que estão compartilhadas é transmitida para um computador central. Em resposta à transmissão de uma lista de sub-sub-partes armazenadas em memória cache que estão compartilhadas, no bloco 705, são recebidas as prioridades para cada uma destas sub-sub-partes. Podem ser designadas prioridades para as sub-sub-partes baseado na frequência com a qual elas estão sendo requisitadas para transferência. Por exemplo, se uma nova atualização tiver sido publicada para proporcionar uma defesa contra um novo vírus de computador, as correções, e assim, as sub-sub-partes, para esta atualização, podem ser transferidas em uma frequência muito alta. Assim, as sub-sub-partes desta atualização podem receber uma alta prioridade.

Quando da recepção das prioridades para sub-sub-partes armazenadas em memória cache, no bloco 707, uma sub-sub-parte armazenada em memória cache com uma prioridade mais inferior é excluída da memória cache. No bloco de decisão 709, é feita uma determinação quanto a se está disponível espaço suficiente na memória cache para a atualização, uma vez que uma sub-sub-parte compartilhada tenha sido excluída. Se for determinado no bloco de decisão 709 que está disponível espaço suficiente, a rotina retorna o controle para a sub-rotina de transferência 600 (FIGURA 6), como ilustrado pelo bloco 723. Entretanto, se for determinado no

bloco de decisão 709 que ainda não está disponível espaço suficiente, no bloco de decisão 711, é feita uma determinação quanto a se existem outras sub-sub-partes armazenadas em memória cache compartilhadas que ainda não foram excluídas.

5           Se for determinado no bloco de decisão 711 que ainda restam sub-sub-partes armazenadas em memória cache que estão compartilhadas, a sub-rotina de prioridade 700 retorna para o bloco 707 e exclui a próxima prioridade mais inferior e o processo continua. Em uma modalidade real da presente  
10       invenção, algumas sub-sub-partes armazenadas em memória cache associadas com correções para atualizações particulares podem ser identificadas como não apagáveis. Por exemplo, se a sub-sub-parte estiver associada com uma atualização que é transferida extremamente freqüentemente, ela pode ser iden-  
15       tificada como não apagável para garantir que existam pares suficientes a partir dos quais a sub-sub-parte pode ser transferida.

          Se for determinado no bloco de decisão 711 que não existem sub-sub-partes armazenadas em memória cache restan-  
20       tes que possam ser excluídas, no bloco de decisão 713 é feita uma determinação quanto a se qualquer correção já foi completamente transferida. Como discutido acima, várias correções podem ser transferidas simultaneamente, de acordo com uma modalidade da presente invenção. Se uma destas cor-  
25       reções completou a transferência, no bloco de decisão 713, ela é identificada e no bloco 717, esta correção é instalada. Tal instalação pode resultar no resto das transferências sendo interrompidas enquanto esta atualização está sen-

do instalada. Entretanto, a tecnologia existente permite que as transferências interrompidas sejam retomadas sem ter que reiniciá-las. Assim, não existe perda substancial de dados transferidos resultando a partir de uma pausa / reini-

5 cialização de uma transferência. Uma vez que a correção transferida tenha sido instalada, no bloco 717, as sub-subpartes desta correção são excluídas da memória cache, como ilustrado pelo bloco 719, a menos que elas sejam identificadas como não apagáveis. Quando da exclusão das transferên-

10 cias, no bloco de decisão 721, é feita uma determinação quanto a se existe espaço suficiente na memória cache para transferir a atualização. Se for determinado no bloco de decisão 721 que não existe espaço suficiente na memória cache, a sub-rotina de prioridade 700 retorna para o bloco de

15 decisão 713 e determina se existem outras correções que terminaram a transferência, e o processo continua.

Voltando para o bloco de decisão 713, se for determinado que não existem correções que terminaram a transferência, no bloco 715 o tamanho da memória cache é expandi-

20 do. O tamanho de uma memória cache pode ser expandido por requisitar que um usuário do sistema do cliente expanda o tamanho da memória cache. De forma alternativa, o tamanho da memória cache pode ser expandido automaticamente. Uma vez que tenha sido preparado espaço suficiente na memória

25 cache para transferir atualizações, por excluir partes existentes a partir da memória cache ou por expandir o tamanho da memória cache, a sub-rotina de prioridade 700 termina no bloco 723, retornando o controle para a sub-rotina de trans-

ferência 600.

A FIGURA 8 ilustra uma rotina do computador central para tornar disponíveis atualizações utilizando um ambiente não-hierárquico, de acordo com uma modalidade da presente invenção. A rotina do computador central 800 começa no bloco 801 e no bloco 803 o computador central recebe uma requisição por uma lista de atualizações. Como discutido acima, a lista de atualizações é uma lista de atualizações disponíveis que são proporcionadas por este computador central para transferência. No bloco 805, o computador central transmite a lista de atualizações identificando transferências disponíveis.

Uma vez que a lista de atualizações tenha sido transmitida, em algum momento posterior, uma requisição por opções de atualização para uma ou mais das atualizações disponíveis identificadas na lista de atualizações é recebida a partir de um cliente, como ilustrado pelo bloco 807. No bloco 809, uma lista de opções de atualização é gerada para as atualizações identificadas na requisição recebida no bloco 807. Como discutido acima, a lista de opções de atualização inclui uma identificação das correções que podem ser utilizadas para obter a atualização. Esta lista é então transmitida para o cliente que requisitou as opções de atualização. No bloco 811, uma requisição de lista de pares é recebida requisitando uma lista de pares para uma ou mais das correções identificadas na lista de opções transmitida no bloco 809. O computador central, no bloco 813, gera e transmite uma lista de pares para as correções requisitadas.

Como discutido acima, a lista de pares identifica, para cada correção, as partes, sub-partes e sub-sub-partes. Adicionalmente, para cada uma destas sub-sub-partes, um ou mais pares e o endereço destes pares são incluídos para identificar a partir de onde estas sub-sub-partes podem ser obtidas. A lista de pares é então transmitida para o cliente, e no bloco 815, é recebida uma identificação das sub-sub-partes que foram transferidas e armazenadas por este cliente que agora estão disponíveis para compartilhamento com outros pares. A lista recebida é utilizada para atualizar uma lista de pares mestre 817 que inclui uma identificação de todas as sub-sub-partes para atualizações disponíveis e dos pares que possuem estas sub-sub-partes disponíveis para transferência. A rotina do computador central 800 termina no bloco 819.

15                Enquanto que as modalidades da invenção foram ilustradas e descritas, será apreciado que várias alterações podem ser feitas nas mesmas sem se afastar do espírito e escopo da invenção.

## REIVINDICAÇÕES

1. Método para atualizar um primeiro dispositivo de computação em uma rede que compreende o primeiro dispositivo de computação e um hospedeiro, o primeiro  
5 dispositivo de computação tendo um cache para armazenar pelo menos uma atualização, que compreende as etapas de:

(a) receber (503) no primeiro dispositivo de computação, a partir do hospedeiro, uma lista de opções de atualização (300) para obter uma atualização de  
10 atualizações disponível, a lista compreendendo informações que identificam uma atualização parcial, uma atualização completa e um valor de hash esperado de um componente atualizável por meio da atualização parcial;

(b) determinar (507) no primeiro dispositivo de  
15 computação, em resposta a etapa (a), uma opção de atualização a partir da lista de atualizações atualizadas, opções de atualização;

(c) receber no primeiro dispositivo de computação, a partir do hospedeiro, uma lista de  
20 dispositivos de computação na rede, em que a lista de dispositivos de computação identifica, para cada parte da pluralidade de partes (409, 411), pelo menos um dispositivo de computação que disponibiliza essa parte para download;

25 (d) descarregar para o primeiro dispositivo de computação, cada parte da pluralidade de partes (409, 411) do pelo menos um dispositivo de computação na lista de dispositivos de computação, em que pelo menos duas

partes da pluralidade de partes (409, 411) são descarregadas a partir de diferentes dispositivos de computação; e

(e) atualizar o software do primeiro dispositivo  
5 de computação usando a pluralidade descarregada de partes, em que a etapa (e) inclui: montar as partes descarregadas em um patch e instalar o patch, em que cada parte descarregada é confirmada com base em um hash da parte antes da montagem das partes descarregadas no patch  
10 **caracterizado pelo fato de que** a etapa (b) ainda compreende:

determinar se o primeiro dispositivo de computação compreende um componente para o qual um valor de hash corresponde ao valor de hash esperado;

15 quando o primeiro dispositivo de computação compreende o componente para o qual um valor de hash corresponde ao valor de hash esperado, selecionar a atualização parcial como a opção de atualização; e

quando o primeiro dispositivo de computação não  
20 compreende o componente para o qual um valor de hash corresponde ao valor de hash esperado, selecionar a atualização completa a partir da opção de atualização, em que a opção de atualização selecionada necessária é segmentada em uma pluralidade de partes (409, 411).

25 2. Método, de acordo com a reivindicação 1, **caracterizado pelo fato de que** a lista de atualização completa compreende um patch completo (305), a atualização parcial compreendendo pelo menos um patch

delta (307, 309, 311).

3. Método, de acordo com a reivindicação 2, **caracterizado pelo fato de que** o patch delta (307, 309, 311) inclui um nome de arquivo de um arquivo a ser  
5 atualizado utilizando o patch delta (307, 309, 311) e o valor de hash esperado associado à atualização parcial compreende um valor de hash esperado para o arquivo.

4. Método, de acordo com a reivindicação 1, **caracterizado pelo fato de que** compreende ainda  
10 proporcionar uma identificação da pluralidade de partes (409, 411) descarregadas; e compartilhar a pluralidade de partes (409, 411) descarregada identificadas com outros dispositivos de computação.

5. Método, de acordo com a reivindicação 4, **caracterizado pelo fato de que,** em um ponto  
15 predeterminado, a pluralidade de partes (409, 411) descarregadas e identificadas não são compartilhadas.

6. Método para proporcionar uma lista de pares (400) a um cliente para descarregar atualizações de  
20 software em uma rede que compreende o cliente e um hospedeiro compreendendo:

receber no hospedeiro, a partir do cliente, uma solicitação de lista de pares para uma lista de pares, correspondendo a uma opção de atualização na lista, a  
25 lista de pares identificando os pares a partir do qual solicitar a opção de atualização da lista;

**caracterizado pelo fato de que** ainda compreende:

transmitir, pelo hospedeiro, para o cliente, uma

lista que identifica as opções para uma atualização disponível, a lista compreendo informação que identifica uma atualização parcial, uma atualização completa e um estado de um componente atualizável por meio da  
5 atualização parcial; o indicador indicando a um cliente se um componente do cliente que tem um estado que corresponde ao estado esperado é atualizável por meio da atualização parcial, de tal modo que o cliente recebe para fazer uma seleção entre pelo menos a atualização  
10 parcial e a atualização completa baseada pelo menos em parte no indicador;

gerar pelo hospedeiro, a lista de pares identificando uma pluralidade de pares e uma pluralidade de partes (409, 411), a pluralidade de partes (409, 411)  
15 definindo coletivamente as opções de atualização requeridas; e

transmitir pelo hospedeiro, para o cliente, a lista de pares, em que a lista de pares identifica pelo menos um par para cada parte da pluralidade de partes  
20 (409, 411).

7. Método, de acordo com a reivindicação 6, **caracterizado pelo fato de que** as opções de identificação da lista para a atualização disponível identificam uma pluralidade de opções de atualização disponíveis.

25 8. Método, de acordo com a reivindicação 6, **caracterizado pelo fato de que** as opções de atualização completas compreendem um patch completo (305) e a atualização parcial compreende pelo menos um patch delta

(307, 309, 311).

9. Método, de acordo com a reivindicação 6, **caracterizado pelo fato de que** a recepção no hospedeiro, a partir do cliente, de uma solicitação para a lista de pares inclui o recebimento de uma solicitação de lista de pares para uma parte da pluralidade de opções de atualização.

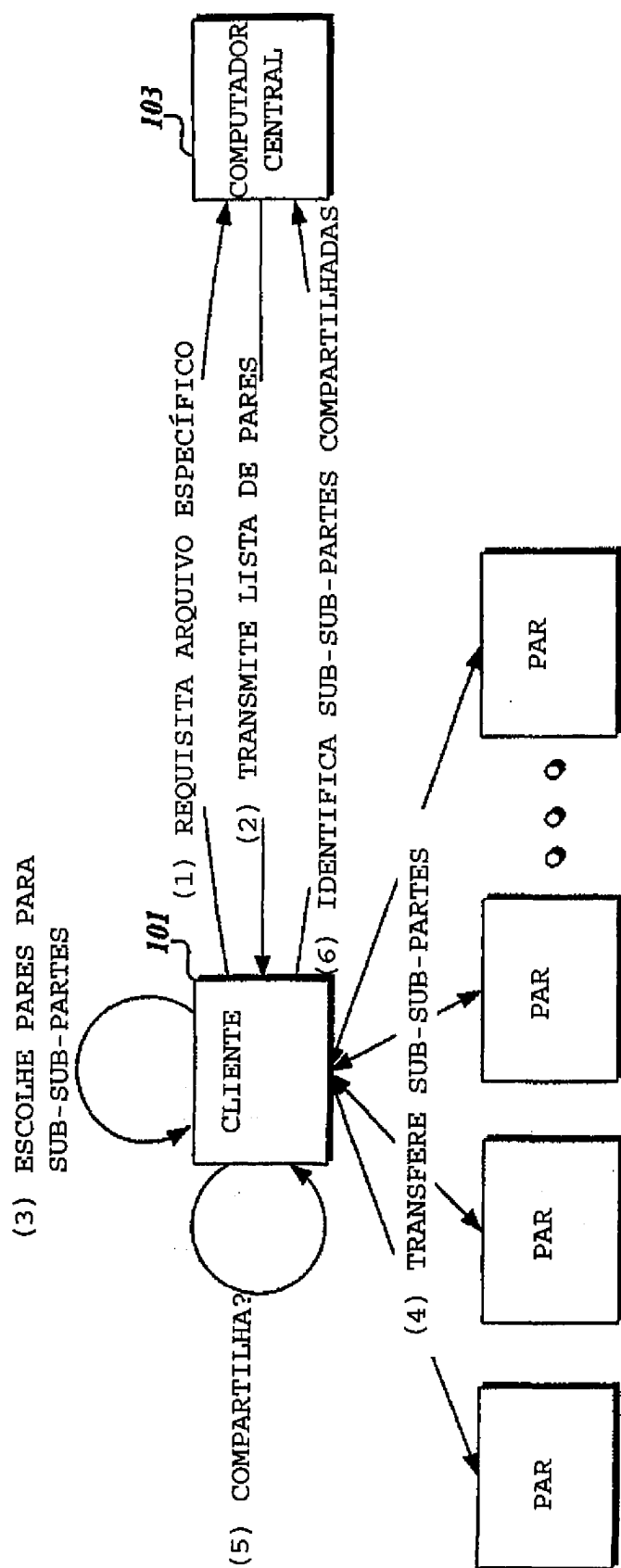
10. Método, de acordo com a reivindicação 6, **caracterizado pelo fato de que** compreende ainda:  
10           receber uma identificação de uma parte em cache da atualização disponível; e atualizar uma lista de pares principais para a atualização disponível.

11. Método, de acordo com a reivindicação 6, **caracterizado pelo fato de que** um dos pares identificados  
15   é um computador hospedeiro.

12. Método, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a informação que identifica o estado do componente atualizável por meio da atualização parcial compreende um valor de hash esperado  
20   do componente atualizável por meio da atualização parcial.

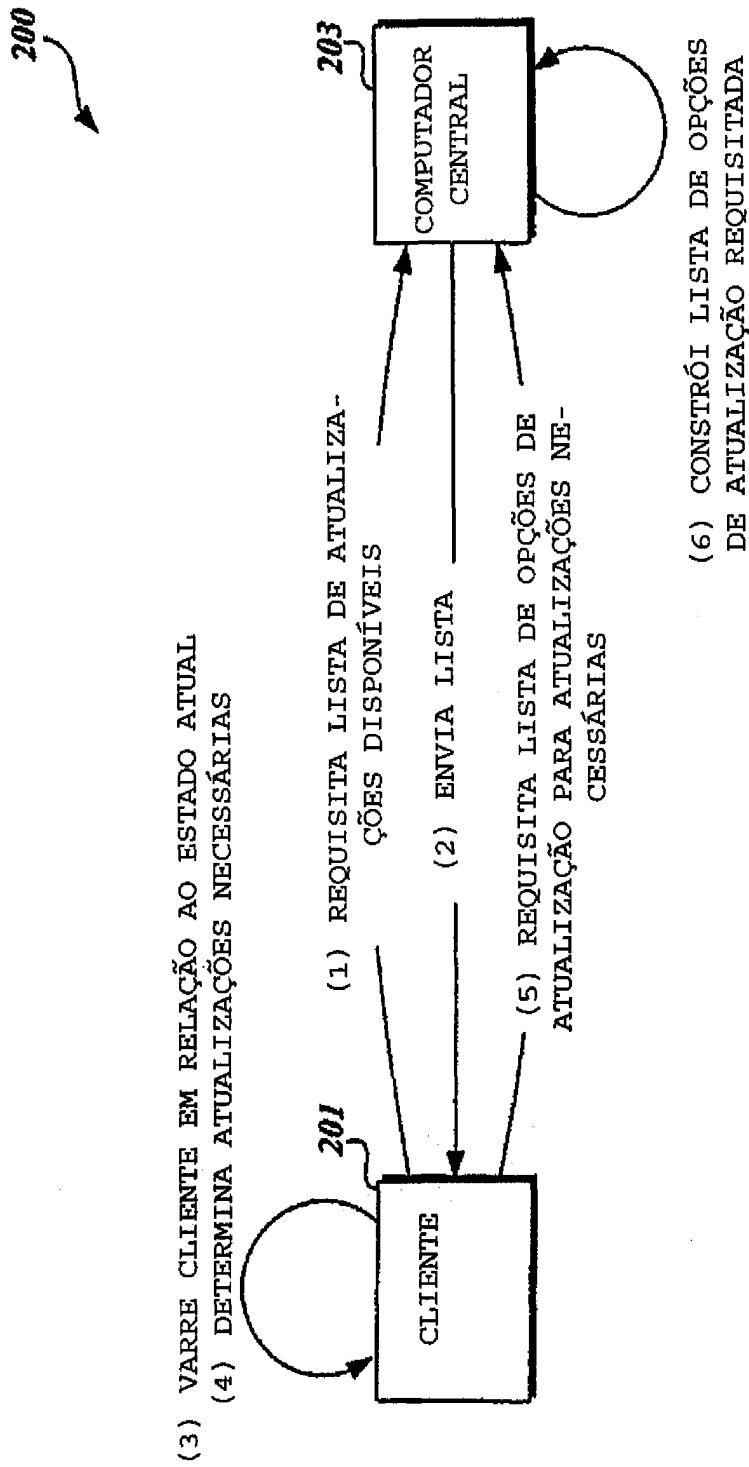
13. Método, de acordo com a reivindicação 12, **caracterizado pelo fato de que** a lista de pares solicitada pelo cliente compreende:  
25           a atualização parcial como a opção de atualização quando o cliente compreende um componente para o qual um valor de hash corresponde ao valor de hash esperado; e

a atualização completa como a opção de atualização quando o cliente não inclui um componente para o qual um valor de hash coincide com o valor de hash esperado.



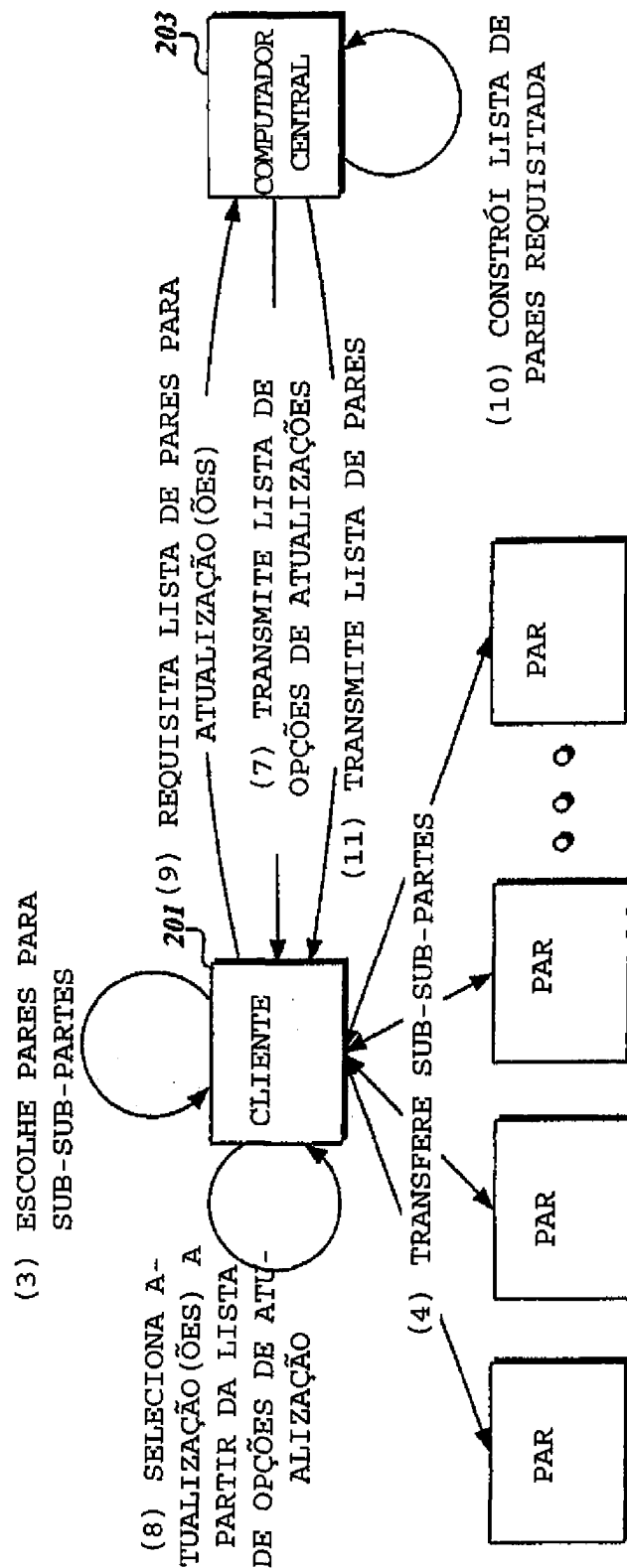
TÉCNICA ANTERIOR

*Fig.1.*



**Fig.2A.**

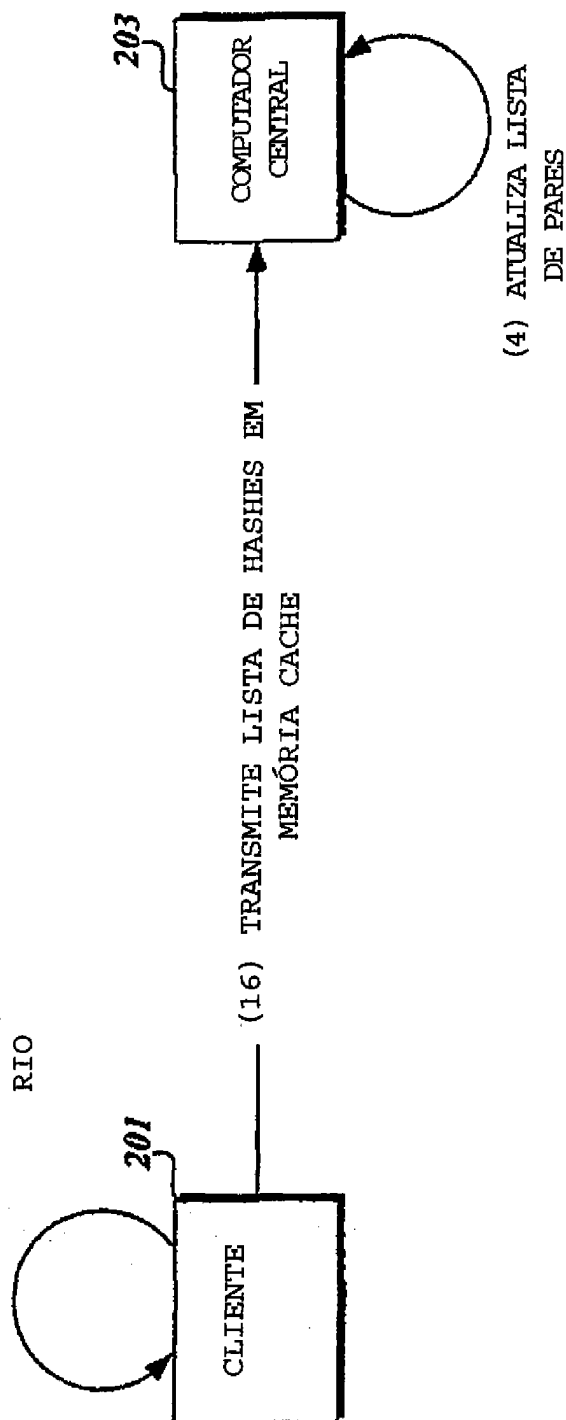
200



TÉCNICA ANTERIOR

Fig. 2B.

(14) MONTA E INSTALA CORREÇÕES  
(15) REINICIALIZA, SE NECESSÁ-  
RIO



*Fig. 2C.*

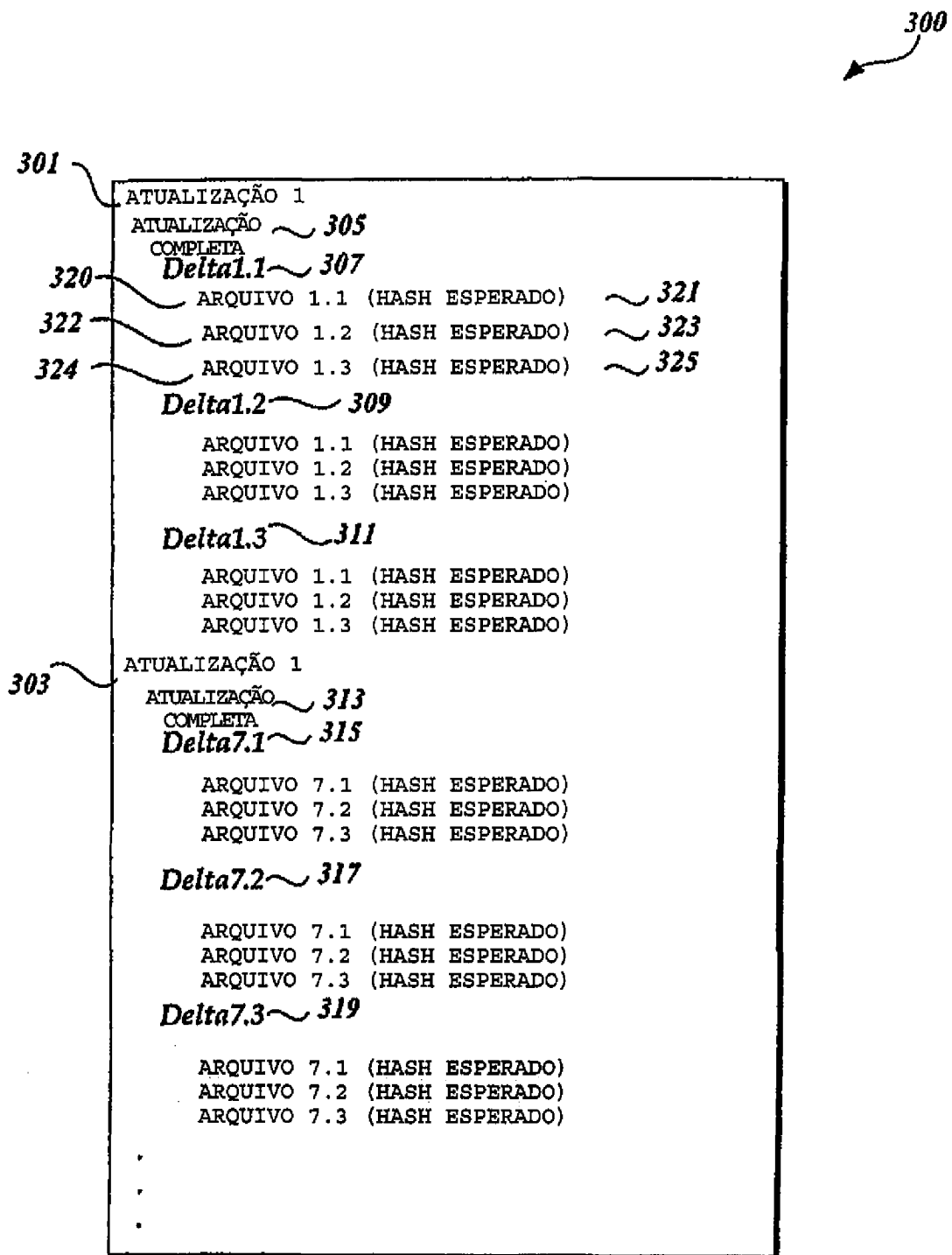


Fig.3.

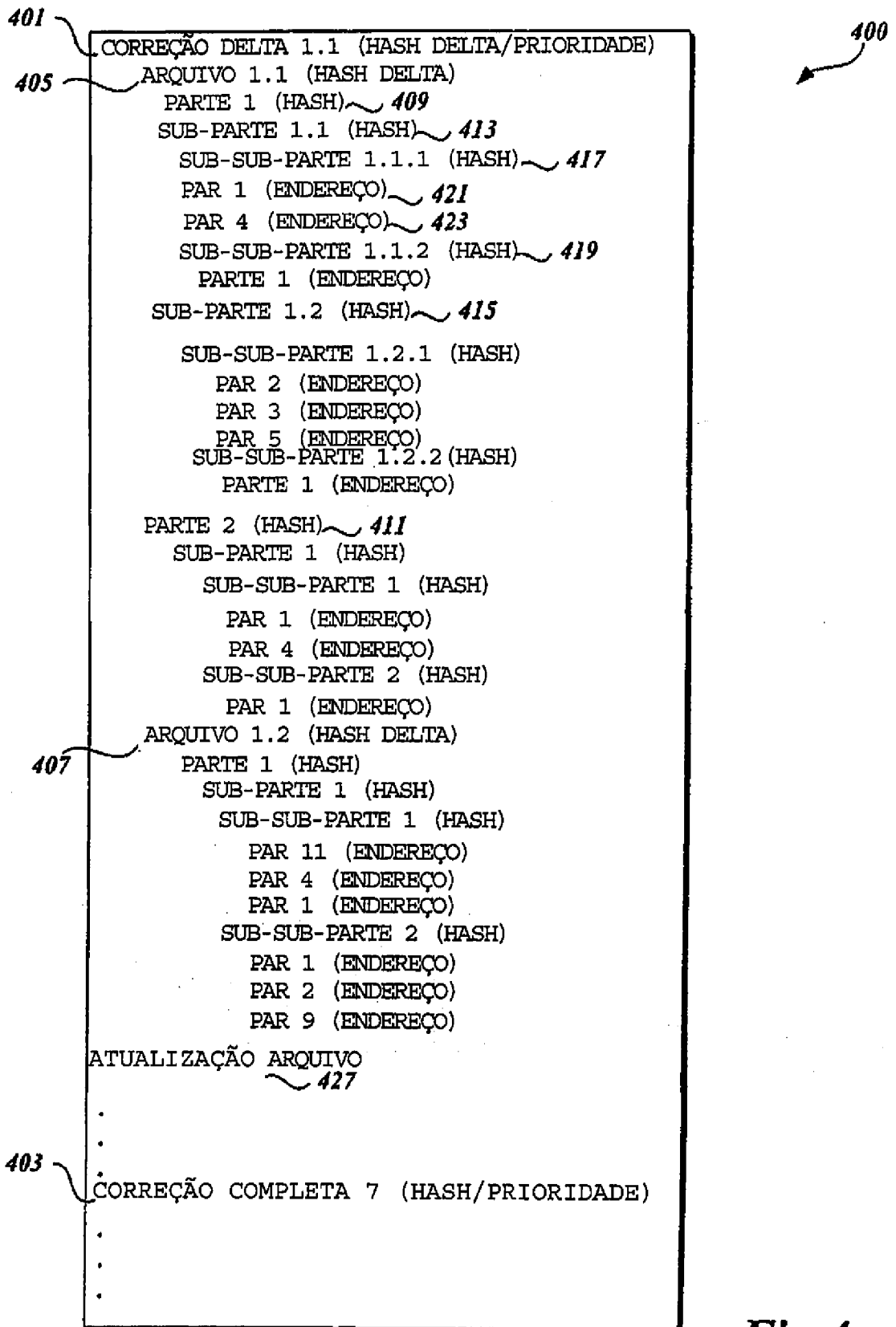
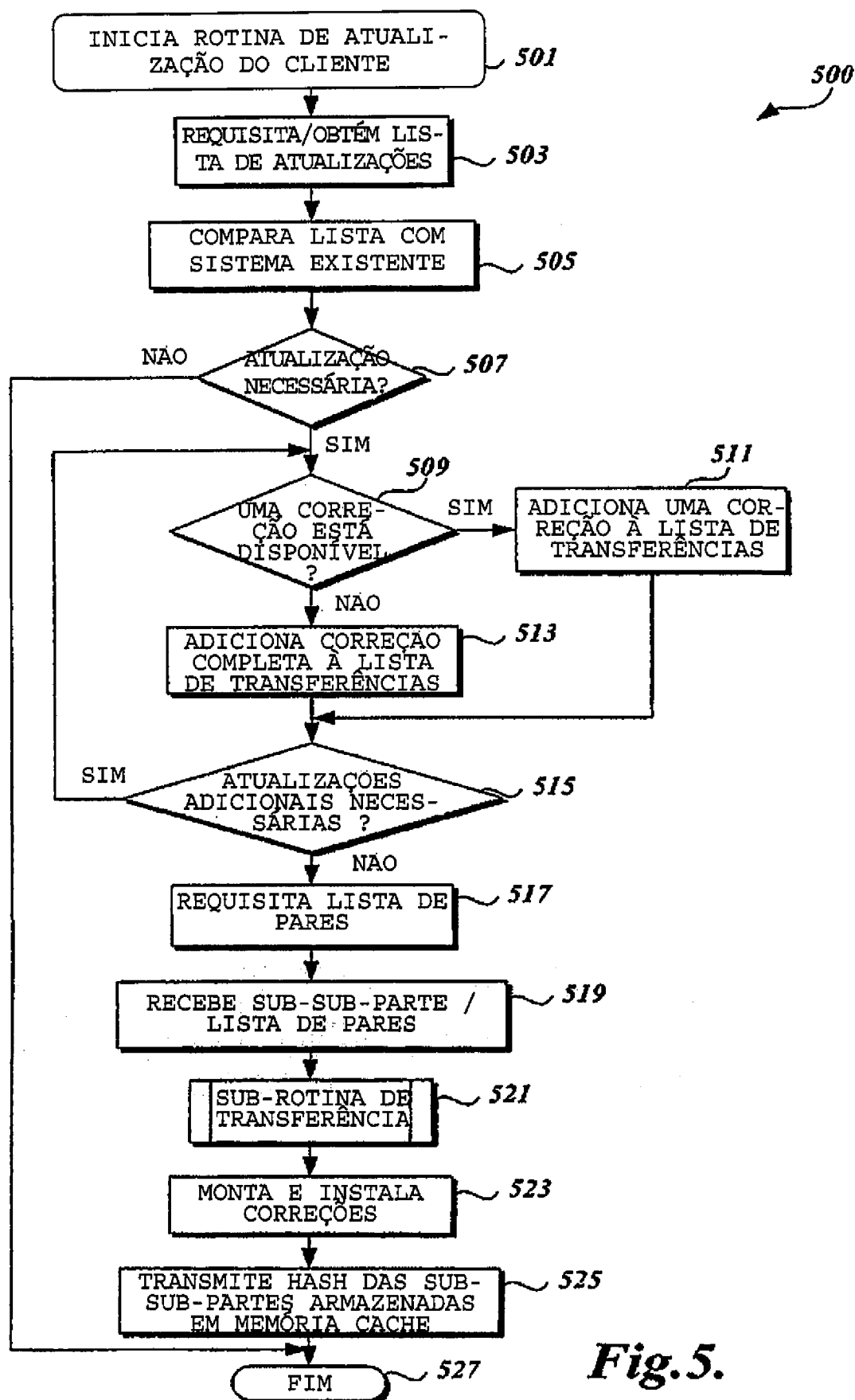


Fig.4.



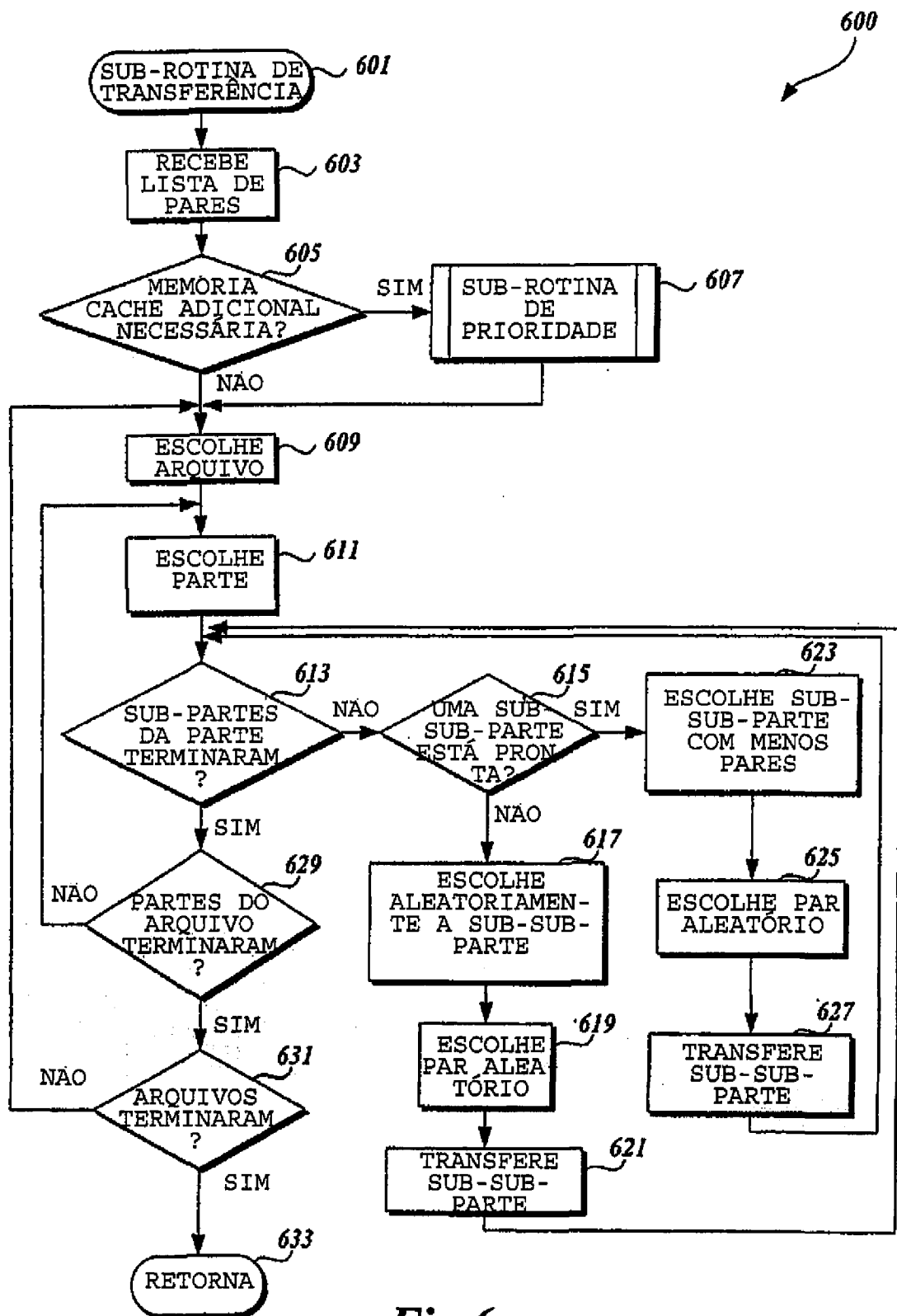
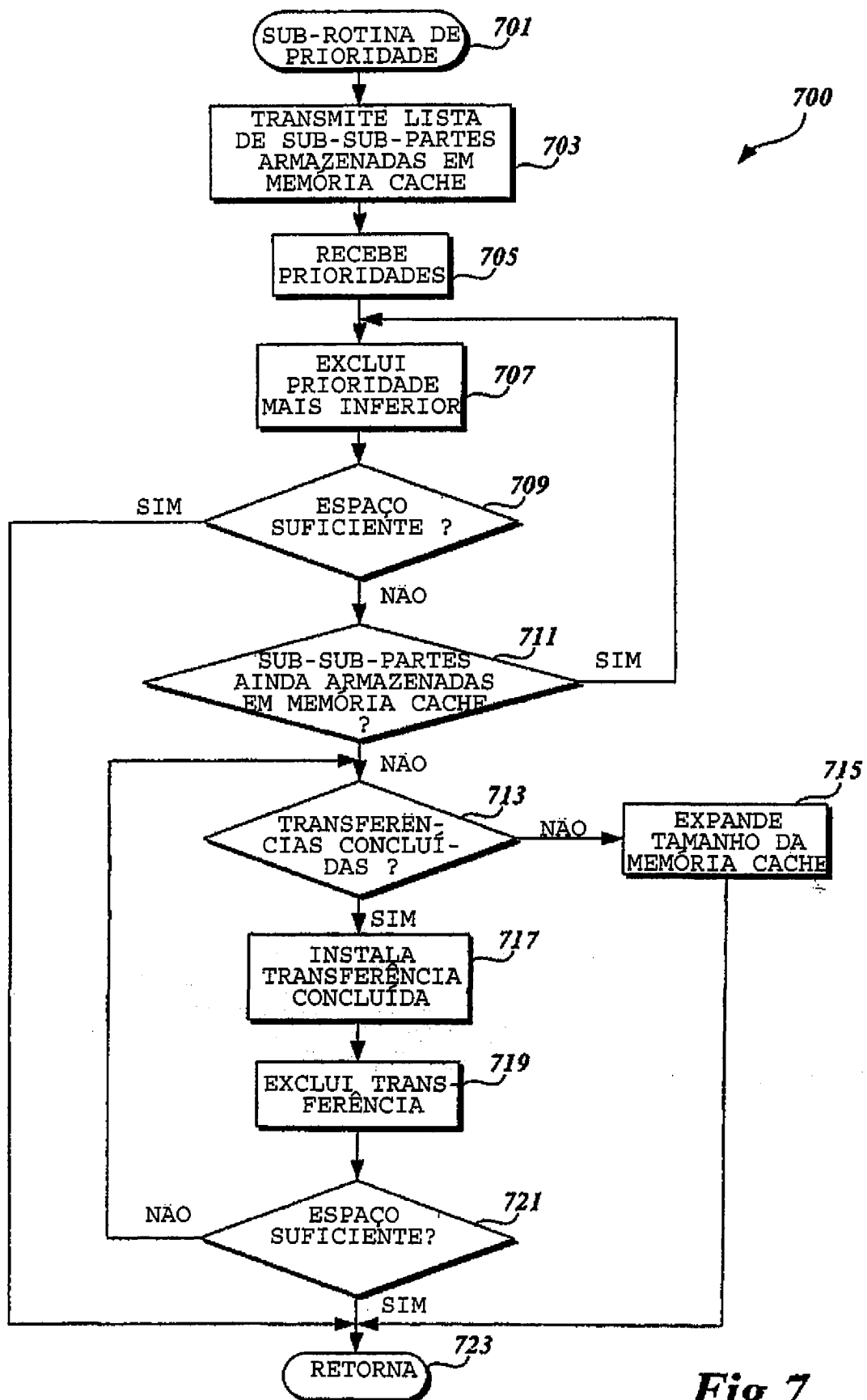
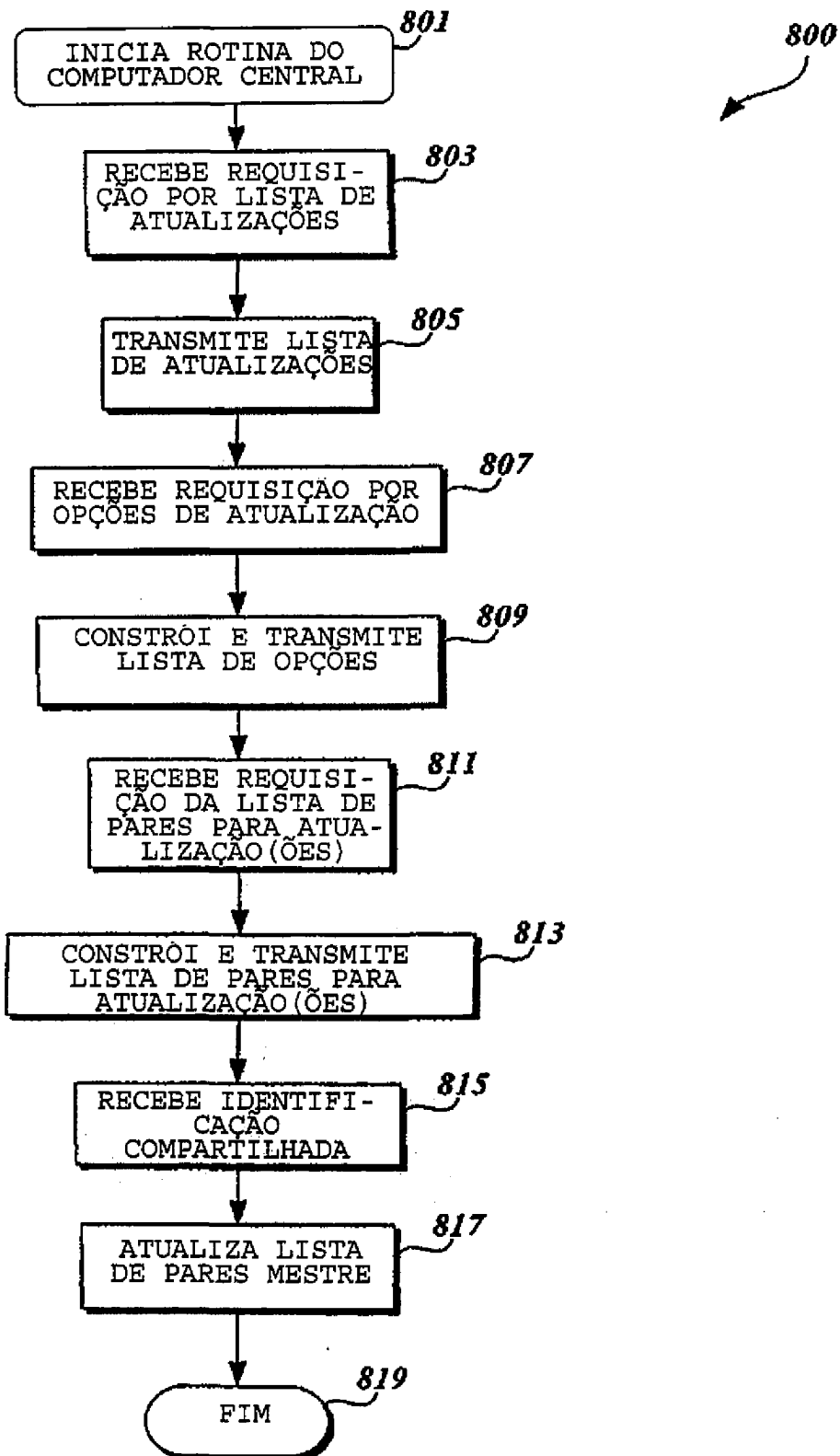


Fig.6.



*Fig.8.*