

Escalabilidade, Autonomia e Segurança em Redes Peer-to-Peer: repensando a P2PSL

Giovani Facchini, Marinho Pilla Barcellos¹

Universidade do Vale do Rio dos Sinos (Unisinos)
Av. Unisinos, 950 – São Leopoldo – RS – Brasil

facchini@gmail.com, marinho@acm.org

Relatório de Andamento do Trabalho de Conclusão – junho 2006

1. Introdução

Nos últimos anos, redes Peer-to-Peer (P2P) aparecem como um paradigma importante de comunicação e de distribuição de dados. Seu crescimento se deve, principalmente, pela popularização de aplicativos para compartilhamento de arquivos.

O aplicativo que iniciou esse processo foi o Napster [1], que permitia que usuários trocassem arquivos de música em formato MP3 [2] alheios a questões de *copyright*. Dado o enorme crescimento visto em número de usuários, logo surgiram outras alternativas para compartilhamento de arquivos. Apesar de Napster não adotar uma filosofia P2P na sua essência, o crescimento desta aplicação reacendeu a questão da **autonomia** e **interação direta entre usuários** como alternativa aos modelos então existentes, predominantemente baseados no paradigma cliente/servidor.

Assim, pesquisadores logo identificaram os principais desafios para essa classe de aplicações: a eficiência na buscas de dados, a formação de uma rede P2P (*overlay*) eficiente (considerando proximidade de rede) e a manutenção dessa rede face à entrada e saída de nós, ambos em um contexto de larga escala.

Com o tempo, surgiram outras aplicações além do compartilhamento de arquivos. Entre elas, pode-se citar conferência de áudio e vídeo, *Instant Messaging* (IM) e aplicações colaborativas.

A implantação de mecanismos de segurança em redes P2P enfrenta uma série de desafios. Embora aplicações P2P possam contribuir para o compartilhamento de recursos e colaboração em larga escala, em ambientes geograficamente distribuídos com controle descentralizado e acoplamento fraco, a sua diversificação e disseminação são dificultadas pela atual **falta de segurança**. Ainda é difícil desenvolver aplicações P2P que atendam a múltiplas combinações de aspectos de segurança, a saber, confidencialidade, autenticidade, integridade, autorização, auditoria, não-repúdio, reputação e anonimato. Há quatro razões para tal. Primeiro, os esquemas atuais para segurança de aplicações P2P cobrem apenas aspectos específicos (como por exemplo autenticação e reputação) e não podem ser facilmente integrados em um sistema único.

Segundo, não conseguem isolar os aspectos de segurança da aplicação. Ao invés disso, obrigam o usuário ou desenvolvedor a lidar com uma interface de programação potencialmente complexa, e passar por um penoso processo de configuração.

Terceiro, as abordagens na literatura demandam um comportamento simétrico e uniforme de todos os *peers* que executam uma aplicação. Esta limitação é indesejável em algumas aplicações P2P, quando os requisitos de segurança podem variar significativamente entre usuários. Para ilustrar com um exemplo trivial, tome-se o Skype, uma aplicação de Voz-sobre-IP: um dado *peer* pode estabelecer diferentes tipos de comunicação com outros *peers*, cada um com seus requisitos de segurança próprios (por exemplo, usuários domésticos e corporativos podem ter diferentes necessidades).

¹ Orientador.

A quarta e última razão é que os esquemas atuais oferecem pouco ou nenhum suporte para implantação gradual, porque eles precisam estar disponíveis em todos os *peers* de uma aplicação. É muito difícil, se não impossível, impor uma adoção abrupta de um novo esquema de segurança em um sistema de larga escala, de acoplamento fraco. Ao invés disso, é importante para o sucesso de sistemas P2P que os mesmos permitam a coexistência entre pares com e sem suporte de um esquema de segurança.

Para atacar esse problema, foi proposta a Peer-to-Peer Security Layer, ou P2PSL [22], uma camada de segurança visando abstrair e desacoplar os conceitos de segurança da comunicação em redes P2P. Ela permite a inclusão de funcionalidades de segurança em aplicações P2P e trata dos problemas de falta de integração, isolamento, assimetria e implantação gradual, recém mencionados. A P2PSL isola a implementação de aspectos de segurança e sua configuração tanto da aplicação P2P como do *middleware* de comunicação subjacente. Cada par pode especificar requisitos de segurança distintos (de acordo com diferentes graus de restrição) para cada canal de comunicação estabelecido com outros pares. Além disso, a implantação da P2PSL pelos pares que compõem a aplicação pode ser feita gradualmente. A implementação está baseada em JXTA, um conjunto consolidado de protocolos para o desenvolvimento de sistemas P2P, que tem sido amplamente usado pela comunidade.

Este Trabalho de Conclusão tem por objetivo **repensar** a arquitetura da P2PSL em função de aspectos de escala e transiência de grupos de pares. Para tanto, será necessário fazer um levantamento do estado da arte em redes P2P e fundamentos de segurança de forma a repensar a P2PSL propondo melhoramentos. Vislumbra-se o desacoplamento total do P2PSL dos mecanismos de comunicação entre pares e a sua inserção de forma modular em um substrato de rede P2P de forma a avaliar o seu desempenho. Este relatório de andamento tem por objetivo apresentar uma síntese dos assuntos estudados (nas Seções 2 e 3), bem como a descrição do andamento das etapas.

2. Redes Peer-to-Peer

As redes P2P diferem em vários aspectos do modelo tradicional cliente/servidor. Conceitualmente, a sua principal diferença está na ausência de um servidor central, porém algumas redes P2P apresentam servidor para fazer a inicialização (*bootstrap*) e realizar algumas tarefas de controle.

Na literatura não existe consenso na conceitualização de uma rede P2P. Segundo [3] existem caracterizações que consideram redes P2P apenas aquelas cujos nós são completamente equivalentes em termos de funcionalidade e das tarefas que eles executam. Porém essa definição falha ao não descrever redes que utilizam o conceito de Supernós e as que utilizam servidores para algumas tarefas como inicialização de nós na rede. Em [4] encontra-se uma definição parecida, pois define P2P como sendo uma rede sem organização hierárquica e sem controle centralizado.

Em [5] é proposta uma caracterização com maior aceitação. Então sistemas P2P são definidos como “uma classe de aplicações que tira vantagens de recursos – armazenamento, ciclos de CPU, conteúdo, presença humana – disponível nas bordas da internet”. Porém, essa definição engloba algumas redes que não são P2P, como as que utilizam conceitos de *bag-of-tasks*. Com isso, percebe-se que não existe uma definição precisa sobre o que é ou não uma rede P2P.

Em [6] são descritos os requisitos que uma rede P2P deve suportar:

- possibilidade de incorporação de nós localizados nas bordas das redes;
- suporte à conectividade variável ou temporária dos nós, bem como a utilização de endereços variáveis;
- capacidade de lidar com diferentes taxas de transmissão entre nós;

- autonomia parcial ou total dos nós em relação a um servidor centralizado;
- escalabilidade;
- possibilidade de comunicação direta entre nós.

O modelo P2P é atrativo pois agrega uma série de funcionalidades ao sistema. Um aspecto importante neste modelo é a inexistência de servidor único que configura um ponto central de falhas, dessa forma o sistema se torna mais robusto contra falhas e ataques externos. Se um nó é atacado, somente este nó será prejudicado, mas o restante do sistema continuará respondendo. Outra funcionalidade importante é a capacidade de compartilhamento de recursos dos nós conectados na rede, que fornecem de forma transparente suas capacidades.

Adaptabilidade à transiência de nós na rede faz com que o sistema continue a prover as suas funcionalidades mesmo com alta taxa de entrada e saída de nós (*churn*). Essa adaptabilidade torna o sistema estável e disponível para os usuários conectados. Isso tudo contribui para auto-organização da rede. Segundo [7], a distribuição da responsabilidade de fornecer serviços por todos os pontos da rede é uma das grandes vantagens das redes P2P.

Para formação de uma rede P2P um *overlay* é criado. Um *overlay* é uma rede de sobreposição que é criada sobre a rede de IP tradicional. O *overlay* é utilizado para prover abstração para a camada de comunicação, pois essa considera somente a conectividade 'virtual' dos nós (para contagem de *hops*) e não como estes estão ligados a rede física. Contrastando com essa abstração, foi proposta [25] uma arquitetura que tenta fazer as ligações entre os nós do *overlay* de maneira mais próxima possível da rede física real, dessa forma evita-se que informações que deveriam percorrer apenas um *hop* no *overlay* acabem percorrendo múltiplos *hops* na rede real.

A seguir, são apresentadas algumas classes de aplicações para redes P2P e então os modelos de organização: estruturada e não-estruturada.

2.1. Aplicações Peer-to-Peer

Para melhor contextualização, será representada abaixo os tipos de aplicações P2P existentes atualmente:

- **Compartilhamento de arquivos:** É o sistema mais disseminado atualmente, pois permite que qualquer usuário da rede possa publicar arquivos (conteúdo) de forma bastante simples. Usuários também podem fazer buscas e obter arquivos sem qualquer restrição. Dessa forma, a rede funciona como um grande sistema de distribuição. Tipicamente o conteúdo dos arquivos publicados é imutável, fazendo que com o tempo ele se torne mais 'popular' e mais disponível na rede. Exemplos desses são: Napster [1], Gnutella [10], KaZaA [11], BitTorrent [12] e DirectConnect [13];
- **Comunicação entre usuários:** essa categoria de aplicações está caracterizada como trocam mensagens entre duas ou mais pessoas na rede. Essas mensagens podem ser de desde texto até vídeos (passando por imagens e voz). Isso ocorre sem a necessidade de um servidor central, mas algumas das aplicações dadas como exemplo utilizam servidores para autenticação de usuários e algumas outras tarefas de controle. Essas aplicações permitem criação de conferência para múltiplos usuários. Exemplos de aplicações são: MSN Messenger [14], ICQ [8] e Skype [15];
- **Computação distribuída:** classe de aplicações que visam utilizar a capacidade de processamento dos nós conectados na rede. Geralmente, essas se utilizam do tempo ocioso dos processadores das máquinas conectadas para computar tarefas que necessitam grande capacidade de processamento. Para redes P2P, o modelo *bag-of-tasks* de computação paralela é o mais adequado, pois os dados das tarefas são divididos de maneira independente e podem ser processados em qualquer nó. Pode-se utilizar outros modelos de computação paralela, porém com alta transiência de nós e

interconexões de alta latência dificultam o controle e podem tornar alguns algoritmos ineficientes. Como exemplo, pode-se citar o OurGrid [16];

- **Armazenamento distribuído:** essa abordagem é análoga ao sistema de arquivos NFS [17], mas ao invés dos dados estarem armazenados em apenas um servidor, os mesmos estão distribuídos entre os nós e portanto precisam de controle de acesso. Se houver replicação é necessário fazer controle de consistência entre as réplicas. Deste modelo, podemos citar o OceanStore [18] como implementação de armazenamento distribuído;
- **Jogos on-line:** essa classe de aplicação tem grande potencial na área de P2P pela escalabilidade, porém praticamente inexistem jogos que utilizam essa abordagem. Os grandes produtores de jogos preferem optar pelo modelo tradicional cliente/servidor (de implementação mais simples) e ter um maior controle do jogo e dos usuários, evitando assim, fraudes e trapaças. Entretanto, na área de pesquisa existem tentativas de criação de jogos utilizando redes P2P como em [19]. Nesse cenário, P2P seria uma escolha natural para jogos massivamente paralelos.

2.2. Redes Não-Estruturadas

Nesse tipo de rede, a estrutura de conexão e organização dos nós é dada de maneira quase randômica, ou seja, quando um nó entra em uma rede recém formada, não existe uma posição pré-determinada para ele. Assim, podemos ter redes desbalanceadas e não temos nenhuma pista sobre onde um nó ou objeto pode se encontrar na rede.

Porém, essas redes tem a vantagem de serem de simples implementação e fácil controle (tanto que se tornaram as mais disseminadas). Elas facilitam a difusão de arquivos de grandes tamanhos (vídeos) que ficam muito difíceis de controlar e distribuir nas redes estruturadas. Mas as desvantagens dessas redes são a falta de escalabilidade dos mecanismos de busca, pois em alguns casos elas recorrem a inundação da rede (alta utilização de banda) ou consulta a servidores (gargalos de comunicação). Alguns exemplos de estrutura serão dados a seguir.

Centralizada. Um exemplo dessa estrutura é o Napster [1], que utiliza um servidor central que faz todas as tarefas de gerência, consulta e conexão. Quando um novo nó entra na rede P2P, ele conecta-se com o servidor e passa para o mesmo a sua lista de arquivos. O servidor então armazena e indexa essa lista de forma que todas as consultas dos nós são feitas para o servidor e respondida por ele. O servidor indica para o nó que fez a pesquisa, quais os nós que contém a informação desejada. Dessa forma, o nó que deseja a informação tenta se conectar com os nós que tem a informação diretamente para a troca de arquivos.

As desvantagens desse modelo aparecem constantemente na literatura, pois como opta pela utilização de um servidor central, este fica como um ponto central de falha. Além disso, o servidor se torna um gargalo dificultando a escalabilidade e se torna um ponto da rede que pode ser atacado prejudicando toda a funcionalidade da rede. A centralização facilita a censura, pois como trata-se de um servidor fixo, o mesmo é de responsabilidade de uma entidade que pode ser responsabilizada judicialmente pelo conteúdo distribuído.

Parcialmente Centralizada. Como exemplo dessa estrutura, podemos citar o KaZaA [11], que utiliza o conceito de Supernós. Um Supernó é um nó pertencente a rede, mas que tem funções equivalentes a de um servidor da estrutura centralizada. Nessa rede, temos alguns nós que se tornam Supernós, então temos uma rede com hierarquia. Nós ditos “comuns” irão apenas contactar o Supernó ao qual se conectaram. O Supernó manterá a lista de arquivos dos nós conectados a ele. Quando um nó dispara uma consulta para um Supernó, este irá verificar localmente em sua base se existe o dado requisitado e depois verificará com outros Supernós se estes conhecem alguma entidade com o conteúdo requisitado. Depois desse passo, o nó que requisitou a consulta, será informado dos nós que dispõem o conteúdo desejado.

Essa abordagem é mais escalável que na estrutura centralizada, pois faz uma divisão das tarefas de consulta. A consulta não fica a cargo de apenas um servidor central que necessita de muita banda e muito processamento, mas de um conjunto de servidores que são nós da rede que se dispõem a essa tarefa. A censura nessa rede torna-se mais complexa, porém os ataques de negação de serviço (DoS) podem se focar nos Supernós. Mesmo para ataques de DoS, essas redes conseguiriam manter seu serviço pelo menos parcialmente, pois seria muito difícil conseguir atacar todos os Supernós simultaneamente.

Descentralizada. O representante mais conhecido desse tipo de rede é o Gnutella [10]. Ele é totalmente descentralizado e utiliza apenas uma lista com alguns *hosts* (gnutellahosts.com) para fazer o *bootstrap* na rede. A partir desse momento, qualquer consulta feita por um nó é requisitada a todos os seus vizinhos. Cada vizinho que recebe uma consulta, passa a mesma para todos os seus vizinhos (menos o origem). Esse algoritmo é conhecido como inundação e se utiliza de TTL (Time To Live) nas mensagens para garantir que a mesma não se propague para toda a rede.

Essa abordagem evita boa parte das tentativas de censura e dificulta alguns ataques, porém o sistema consome bastante recursos de rede propagando mensagens. Se o TTL for muito grande, toda a rede será inundada tornando-a bastante ineficiente. Se as informações requisitadas por um nó estiverem muito distantes do mesmo, as requisições podem não alcançar o nó detentor do conteúdo desejado. Por fim, algoritmos de *random-walks* podem ser usados para diminuir a quantidade de tráfego e ainda conseguir boa qualidade na consulta.

2.3. Redes Estruturadas

Nesse tipo de rede os nós são organizados em um grafo onde objetos são mapeados para chaves através de funções *hash* e cada nó da rede fica responsável por um subconjunto do total global de chaves. Entretanto, essa estrutura não permite a busca através de consultas complexas pois os armazenamentos e consultas são feitos através das chaves e não através do conteúdo. Com isso, conteúdos similares podem estar em lugares totalmente diferentes na rede, pois tem chaves diferentes. Para exemplificar esse tipo de abordagem, algumas redes que implementam a mesma serão explicadas a seguir.

CAN (Content Addressable Network) [20]. Essa rede utiliza um espaço d-dimensional para mapear nodos e chaves. Quando o primeiro nó entra na rede, ele é responsável por todo o espaço, mas a medida que mais nós vão entrando, esse espaço vai sendo particionado entre os nós de forma a fazer balanceamento de carga. Cada área que um nó é responsável é chamada de zona onde objetos são mapeados. O mapeamento ocorre da seguinte forma: Um objeto com valor V é mapeado para uma chave K ; a chave K é mapeada para um ponto P no espaço d-dimensional; essa zona em que o ponto P foi mapeado é de responsabilidade de um nó N ; Assim, o nó N fica responsável pelo objeto de valor V .

Para fazer consultas na rede tem-se um procedimento com passos semelhantes a inserção: um nó N_1 fornece para a rede a chave K para recuperar o objeto; essa chave é mapeada para um ponto P na rede; esse ponto P fica na zona Z que é de responsabilidade do nó N_2 ; N_1 envia uma mensagem em direção a N_2 requisitando o objeto, mas como N_1 não sabe o endereço de N_2 , N_1 envia para uma região vizinha que irá passar essa mensagem de região para região até chegar no nó N_2 responsável pela região Z .

Chord: Essa rede utiliza um espaço circular para fazer o mapeamento de nós e objetos. O posicionamento de nós e objetos no anel é dado por uma função *hash* consistente e então ordenada fazendo modulo 2^m (que indica o tamanho do anel). Cada nó que entra na rede é conectado no anel mapeado através da função *hash*. Esse nó contém ponteiros para seu sucessor e uma tabela chamada "*Finger Table*" (utilizada para melhorar o desempenho não necessitando rotear mensagens por todos os nós da rede) onde cada elemento é o endereço do nó $n+2i$ (onde n é o número do próprio nó e i é número da entrada na tabela). Com isso, mensagens podem dar 'saltos' na rede.

Para mapear um elemento sua chave é computada e com isso sabe-se em que posição esse objeto deveria estar no anel. O objeto é roteado até sua posição e se não existe um nó responsável pela posição que o objeto foi mapeado, o nó sucessor fica responsável pelo objeto. Para requisições o funcionamento é análogo, pois com a chave do objeto sabe-se a posição do mesmo na rede. Dessa maneira, uma mensagem é roteada até o nó responsável pelo objeto.

3. Fundamentos de Segurança em P2P

Como visto em [22], a segurança em redes de computadores abrange diversos aspectos que atendem a diferentes objetivos dos pontos de vista dos usuários das aplicações. Esses mesmos conceitos de aspectos de segurança podem ser transportados diretamente para redes P2P ocasionando os seguintes requerimentos:

- **Confidencialidade:** Essa característica, quando aplicada, evita que terceiros consigam identificar a informação trocada entre entidades. Para realizar esse procedimento, duas classes de aplicações são utilizadas: criptografia simétrica e criptografia assimétrica. Do ponto de vista de redes de computadores a criptografia pode ser aplicada na comunicação como um todo ou apenas em alguns campos (dados, cabeçalhos, etc). Em redes P2P, implica que a comunicação entre dois nós da rede não possa ser identificada por outros nós ou por entidades fora da rede.
- **Autenticação:** Requisito que é utilizado quando existe a necessidade de verificar se uma entidade é realmente quem ela afirma ser. Atualmente, para fazer essa verificação utiliza-se assinaturas digitais com algoritmos de criptografia assimétricos, ou seja, um transmissor pode assinar uma mensagem e enviar ao receptor, então o receptor verificará se a assinatura do transmissor está correta. Em P2P isso garante que certo usuário ou nó está sendo identificado e pode-se confiar nas informações provenientes do mesmo.
- **Integridade:** A integridade é a garantia de que uma informação não tenha sido modificada voluntária ou involuntariamente. Dessa maneira, se uma mensagem é enviada de um nó A para um nó B, a integridade garante que no caminho de A para B a mensagem não foi modificada. Assim, pode-se saber se houve corrupção da mensagem (por fatos isolados como links defeituosos) ou se alguma entidade maliciosa está tentando injetar informações errôneas na rede. Para garantir integridade é utilizado um algoritmo de *Hash*.
- **Não-Repúdio:** Garante que uma entidade não possa negar operações (informações) feitas por ela. Essa característica já está, em parte, presente quando utilizamos autenticação, pois garantimos que uma entidade é quem ela realmente diz ser. Em redes P2P, isso pode ser utilizado como penalização, ou seja, se um nó A envia informação inválida para o nó B, esse nó B pode assegurar-se que isso trata-se de uma operação de 'má fé' do nó A, pois o nó B tem certeza que a mensagem é do nó ^a
- **Autorização:** Define recursos que certa entidade pode acessar. A autorização faz o controle de acesso de algum usuário (ex: nó) sobre recursos disponíveis (ex: arquivo, processamento) de forma a limitar o acesso apenas a quem a usuários autorizados. Uma das formas de fazer esse controle é através de *Access Control List (ACL)*.
- **Auditoria:** Capacidade de analisar o comportamento do sistema ou de entidades do sistema. No caso de redes P2P, capacidade de verificar como certo nó está agindo. Para esse tipo de análise é muito utilizado o artifício de *log*. Os *logs* são informações sobre o que está ocorrendo em um nó em um determinado momento (suas ações).
- **Anonimidade:** Permite que uma entidade não possa ser identificada em uma comunicação com outra, ou seja, sua identidade é desconhecida. Se um nó A quer contactar um nó B, sem que B saiba quem o está contactando, A pode utilizar-se de nós

intermediários para fazer a comunicação para ele.

- **Reputação:** Indica a confiabilidade de um nó na rede, ou seja, se o mesmo contribui com a rede fornecendo recursos ou apenas onera o sistema sem contribuir com o mesmo. Existem muitos trabalhos sobre esse tema, pois como as políticas dependem de informações dadas por vários nós, o sistema tem uma grande quantidade de vulnerabilidades. De maneira geral, um nó tem melhor reputação se compartilha e fornece bastante recursos e o faz de maneira correta, enquanto terá uma reputação ruim caso contrário.
- **Disponibilidade:** Em sistemas computacionais, essa propriedade refere-se ao fato do sistema estar pronto para responder a requisição de clientes quando for solicitado. Em redes P2P significa que os serviços da rede continuarão disponíveis, mesmo com a alta transiência de nós.
- **Negabilidade:** Permite a uma entidade negar a responsabilidade sobre dados por ela armazenada. Isso ocorre quando uma entidade A (nó) armazena dados de outra entidade B mas esses dados estão cifrados, ou seja, a entidade A não consegue identificar os dados armazenados. Isso tem por objetivo garantir que uma entidade não seja processada por armazenar conteúdos com *copyright* que não pertencem a mesma.

4. P2PSL

A P2PSL (*Peer-to-Peer Security Layer*) visa prover uma camada de segurança modular para integração com aplicações P2P. Como pode ser visto em [24], a P2PSL foi implementada independentemente da aplicação sob a qual ela irá executar e do substrato de rede subjacente. Sua API é de fácil manipulação e permite a inserção gradual de requisitos de segurança em aplicações.

Uma rede P2P tem natureza assimétrica de operações, pois cada nó é independente e pode exigir ou não certos níveis de segurança que cada um dos requisitos citados anteriormente pode prover. Pensando nisso, a P2PSL foi concebida de maneira modular e com configuração individual em cada cliente para que as aplicações possam optar por seus requisitos.

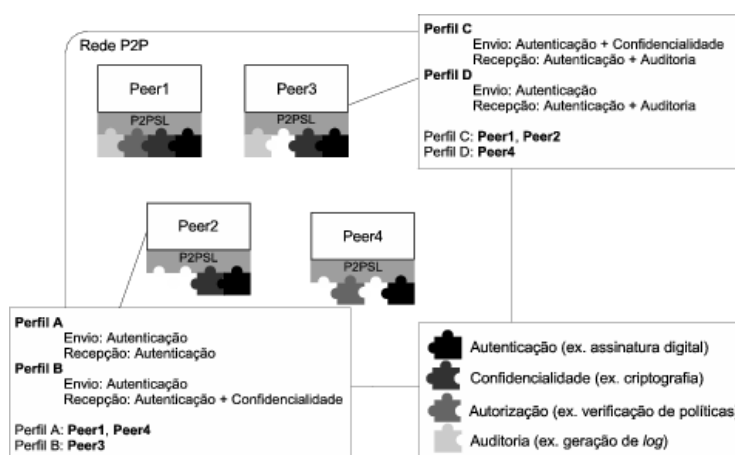


Figura 1: Rede de nós usando P2PSL

P2PSL pode ser visualizada como a montagem de vários módulos. Como pode-se observar na Figura 1, há uma rede P2P com quatro nós. Cada nó tem em seu corpo alguns dos módulos para incorporação de segurança e cada um deles define políticas para utilização dos módulos. Para evitar a necessidade de configurar políticas de segurança para cada nó da rede, a camada oferece o conceito de perfis. Tome-se como exemplo o Peer2 da Figura 1, ele tem dois perfis configurados. O Perfil A requisita autenticação para envio e recebimento de mensagens, o Perfil

B requisita autenticação para envio e autenticação mais confidencialidade para recebimento de mensagens.

Pode-se ver que os nós Peer1 e Peer2 foram colocados no Perfil A e o nó Peer3 foi colocado no Perfil B. Isso simplifica a configuração, pois podemos atribuir alguns perfis básicos e selecionar os nós que desejamos colocar nesses perfis. Pode-se criar um perfil “*default*” e atribuir a ele todos os novos nós (desconhecidos) e então ter um outro perfil *legancy* para os nós que não estão utilizando a camada de segurança. Isso facilita o suporte aos sistemas legados. Dessa forma pode-se migrar gradativamente para uma solução com incorporação de segurança.

A camada funciona como um *wrapper* entre a aplicação e o substrato de rede. Ela recebe os dados vindos pela rede (de algum nó), aplica os módulos necessários e entrega os dados para a aplicação. Quando a aplicação deseja se comunicar com algum nó, essa contacta a camada de segurança enviando os dados e o destino. Dessa forma, a camada aplica os módulos necessários para a mensagem poder ser entregue a aplicação do nó destino e envia através do substrato de rede. Esse procedimento é executado pois existem módulos, como o de criptografia, que necessitam modificar o conteúdo da mensagem e no outro lado o mesmo módulo precisa ser aplicado para recuperação dos dados.

Com sua implementação modular utilizando carga dinâmica de módulos, fica simples a adição de novos módulos de segurança na ferramenta de maneira a atender aos requisitos das aplicações. Para isso, um novo módulo basta implementar a interface definida pela P2PSL e colocá-lo junto aos outros módulos no diretório de repositório.

Cada módulo deve ser caracterizado de forma a estabelecer suas necessidades. Isso é feito através de um arquivo XML que precisa definir algumas informações para a camada poder controlar o comportamento do módulo. São apenas quatro parâmetros que precisam ser informados para a camada no arquivo XML, os outros são referentes aos parâmetros necessários ao funcionamento do módulo. Como exemplo de parâmetros pode-se citar a indicação de chave pública e privada (no caso do módulo de criptografia) ou o arquivo para gravar a geração do módulo de *logs*.

Para cada módulo desenvolvido, P2PSL associa quatro características. São quatro atributos que podem ser utilizados para indicar o comportamento do módulo:

- **export_requirement**: indica se o nó remetente da mensagem necessita modificar a mensagem sendo enviada a outro nó para que este possa aplicar o mesmo módulo e recuperar o conteúdo original da mensagem.
- **obligatory_if_applied**: indica se o nó receptor de uma mensagem necessita utilizar o módulo para recuperar a conteúdo original (que fora modificado pelo remetente utilizando *export_requirement*).
- **allow_on_bcast_sending**: indica se o módulo pode ser utilizado em transmissões *broadcast*.
- **discard_on_failure**: indica se o módulo deve descartar a mensagem silenciosamente se a verificação aplicada pelo módulo falhar.

Para configuração da camada é empregado um arquivo XML. Nele estão descritos como são compostos os perfis definidos pelo usuário e a ordem de aplicação dos módulos para envio e recebimento de mensagens. Um atributo importante para declaração no perfil é o *respect_remote_requirements* que indica se o nó respeitará os pedidos de requisitos de outros nós aplicando os módulos necessários para esse fim.

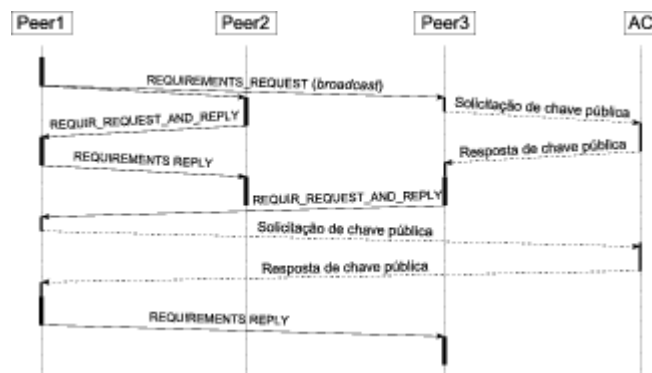


Figura 2: Diagrama representando descoberta de requisitos

Para que os nós da rede façam a troca de requisitos de segurança, foi criado um protocolo. Primeiramente um nó A que deseja entrar na rede envia uma mensagem de *broadcast* com a tag REQUERIMENTS_REQUEST requisitando para os outros nós quais são seus requerimentos. Se um nó B que recebe a mensagem já conhece a chave pública do remetente A, ele envia uma mensagem para A com a tag REQUERIMENTS_REQUEST_AND_REPLY. Isso indica quais são os requerimentos B para com A e pergunta quais os requerimentos de A para com B. Se A conhece a chave pública de B, A responde com tag REQUERIMENTS_REPLY indicando quais são seus requerimentos com B. Nesse fluxo descrito, caso algum nó não conheça a chave pública do nó que enviou a mensagem, o mesmo contacta uma AC (Autoridade Certificadora) para requisitar a chave pública. Um exemplo está descrito na Figura 2, onde Peer1 entra na rede e ambos Peer1 e Peer2 se conhecem, porém Peer3 e Peer1 não se conhecem.

A Figura 3 mostra como é a implementação e funcionamento da camada dentro de um nó. Como pode-se perceber a comunicação é feita baseada em JXTA/JAL permitindo fácil integração da camada com aplicações feitas sobre essa plataforma.

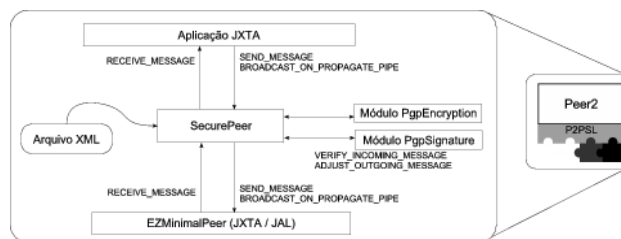


Figura 3: Instância da P2PSL em um nó

Atualmente, existe quatro módulos de segurança disponíveis na camada:

- **Assinatura PGP:** módulo utilizado para assegurar autenticidade e integridade de mensagens;
- **Criptografia PGP:** esse módulo é responsável por garantir confidencialidade nas mensagens;
- **Verificação de políticas de acesso:** responsável por controle de acesso a recursos. Módulo baseado em *Role-Based Access Control* (RBAC);
- **Geração de logs:** Usado para gerar log das ações e mensagens trocadas. Pode ser usado para auditoria.

5. Descrição do Modelo Preliminar

Com base na observação da implementação original e como a mesma funciona, foram

detectados alguns problemas para integração da mesma com qualquer substrato de rede atualmente existente. Isso ocorre pois a camada responsável pela comunicação atualmente é JXTA. Isso faz com que qualquer aplicação que queira utilizar as funcionalidades providas pela camada necessite usar uma rede baseada em JXTA e em código Java.

Pensando nessas limitações, propõe-se transformar a camada P2PSL em um tipo de infraestrutura de comunicação que seja independente de linguagem e que não seja responsável por fazer comunicação. Dessa forma, seria possível adicionar aspectos de segurança gradativamente em uma rede P2P existente que não fosse baseada em JXTA e que não fosse implementada em Java.

A nova proposta ainda está em fase de modelagem, mas para fazer o isolamento proposto seria ideal que essa nova camada fosse um *daemon* que ficasse em execução na máquina que roda o substrato P2P esperando por ações deste. Uma abordagem possível é a criação de um protocolo de comunicação entre a camada e uma entidade desejando utilizá-la via *sockets* locais. Com isso, um substrato existente poderia contactar a camada para realizar as tarefas de prover os requisitos de segurança e continuar com sua camada de comunicação subjacente sem a intervenção da P2PSL.

6. Andamento das Etapas

6.1. Revisão Bibliográfica

Durante o semestre foi realizada a leitura de artigos sobre redes peer-to-peer e aspectos de segurança nas mesmas. Juntamente com isso, foi estudado o comportamento da camada de segurança P2PSL, algumas dificuldades e limitações da mesma. Os estudos nessa área seguem, para tentar identificar possíveis pontos de melhoria na camada e como estes podem ser implementados.

6.2. Definição de Modelo

Esse processo de definir e repensar como a P2PSL será modificada se encontra em andamento e está sendo colocado em questionamento junto com o orientador e alguns mestrandos para que possa-se obter um modelo flexível e com alta capacidade de integração.

6.3. Outras

As outras atividades como implementação do modelo proposto, análise de melhorias e desempenho das soluções ainda não iniciaram, pois as mesmas dependem de ter-se uma definição formal do modelo a ser seguido.

6.4. Estimativas

Algumas das tarefas previstas no cronograma inicial tiveram alguns atrasos para sua conclusão, mas isto não inviabiliza a conclusão das outras tarefas. A única tarefa prevista e ainda não iniciada é a de implementação, pois ainda não foi definido o modelo. Em Julho espera-se, com a definição do modelo, começar a implementação do mesmo e a integração deste com alguma rede P2P para prova de conceito, fazendo testes quanto ao funcionamento da rede composta com nós com e sem a camada de segurança.

7. Referências Bibliográficas

[1] OpenNap: Open Source Napster Server (2006). website. <http://opennap.sourceforge.net/>

[2] Fraunhofer IIS – Audio & Multimedia. <http://www.iis.fraunhofer.de/amm/techinf/layer3/>

[3] Theotokis, S. A. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371.

- [4] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005). **A survey and comparison of peer-to-peer overlay network schemes**. *IEEE Communications Surveys & Tutorials*, 7(2):72–93.
- [5] Shirky, C. 2000. **What is p2p... and what isn't**. <http://www.oreillynet.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html>. O'Reilly.
- [6] Rocha, J.; Domingues, M.; Callado, A.; Souto, E.; Silvestre, G.; Kamienski, C.; Sadok, D. **Peer-to-Peer: Computação Colaborativa na Internet**. Minicurso, *Simpósio Brasileiro de Redes de Computadores*, maio 2004.
- [7] Sousa, Nuno Miguel Tavares de. **Peer-to-Peer Computing**. Universidade do Porto. http://gnomo.fe.up.pt/~eol/MEMBERS/nuno_sousa/old/ppc/artigo.html
- [8] **ICQ.com** (2006). website: <http://www.icq.com/>
- [9] **Sourceforge.net** (2006). website: <http://sourceforge.net/>
- [10] **Gnutella – A protocol for a revolution** (2006). website: <http://rfc-gnutella.sourceforge.net/>
- [11] Liang, J., Kumar, R., and Ross, K. W. (2004). **The kazaa overlay: A measurement study**. In *19th IEEE Annual Computer Communications Workshop (CCW 2004)*.
- [12] **BitTorrent** (2006). website: <http://www.bittorrent.com>
- [13] **DirectConnect** (2006). website: <http://www.dcpp.net>
- [14] **MSN Messenger** (2006). website: <http://messenger.msn.com/>
- [15] **Skype – The whole world can talk for free** (2006). website: <http://www.skype.com>
- [16] Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2004). Discouraging free riding in a peer-to-peer cpu-sharing grid. In *13th IEEE Symposium on High Performance Distributed Computing (HPDC'04)*.
- [17] Shepler, S.; Callaghan, B.; Robinson, D.; Thurlow, R.; Beame, C.; Eisler, M.; Noveck, D. **Network File System (NFS) version 4 Protocol**. RFC 3530.
- [18] OceanStore (2006). **The OceanStore Project website**. <http://oceanstore.cs.berkeley.edu/>
- [19] Knutsson, Bjorn; Lu, Honhui; Xu, Wei; Hopkins, Brian. **Peer-to-Peer support for massively multiplayer games**. In *INFOCOM 2004*, volume 1, paginas 96-107, Março 2004.
- [20] Ratnasamy, S., Francis, P., Handley, M., Karp, R. **A scalable content-addressable network**. In *Proceedings of SIGCOMM 2001*.
- [21] Stoica, L., Morris, R., et al., **Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications**. *IEEE/ACM Transnet*. vol .11, no. 1, 2003, pp. 17-32.
- [22] Detsch, André. **Uma Arquitetura para Incorporação Modular de Aspectos de Segurança em Aplicações Peer-to-Peer**. Dissertação de Mestrado (Unisinos), 2005.
- [23] Barcellos, Marinho P., Gaspary, Luciano P. **Fundamentos, Tecnologias e Tendências rumo a Redes P2P Seguras**. Em *Simpósio Brasileiro de Redes de Computadores 2006*. Minicurso.
- [24] Detsch, A., Gaspary, Luciano P., Barcellos, Marinho P., Sanchez, Ricardo N. **Uma Abordagem para Incorporação Flexível de Aspectos de Segurança em Aplicações Peer-to-Peer**. Em *Simpósio Brasileiro de Redes de Computadores 2006*.

[25] Madruga, M., Batista, Thais V., Guedes, Luiz A. **Uma Arquitetura P2P Baseada na Hierarquia do Endereçamento IP**. Em *Simpósio Brasileiro de Redes de Computadores 2006*.