TÉCNICO LISBOA

Estudo sobre Mecanismos de Segurança Aplicáveis à Difusão de Conteúdos Multimédia Adaptativos e Escaláveis em Redes Peer-to-Peer

Nkusu Felicidade Domingos Quivuna

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientador: Prof. Rui António dos Santos Cruz

Júri

Presidente: Prof. Luís Manuel Antunes Veiga
Orientador: Prof. Rui António dos Santos Cruz
Vogais: Prof. Miguel Filipe Leitão Pardal
Prof. Daniel Jorge Viegas Gonçalves

Dezembro de 2017

Agradecimentos

Agradeço em primeiro lugar a Deus, pelo dom da saúde e pela proteção concedida durante a fase realização desta formação. A Instituição, Universidade Técnica de Angola (UTANGA), através do Departamento de Tecnologia de Informação e Comunicação, por me terem dado a oportunidade de obter a bolsa de estudo para fazer esta formação em (Portugal).

Agradeço ao meu orientador, o Prof. Dr. Rui Santos Cruz, pela disponibilidade, paciência, dedicação e por todo apoio concedido durante a elaboração da dissertação e o acompanhamento.

Aos meus colegas em geral sem distinção, agradeço pelo carinho e por todo apoio, nos bons e nos maus momentos que partilhamos, foi uma experiência agradável que passamos juntos.

Agradeço de forma especial, à minha família, ao meu pai, Manuel Quivuna e a minha mãe Isabel Domingos Quivuna, por todo o apoio durante a frequência no Ensino Superior, e, à minha irmã Ângela e ao seu esposo. Minha família meu porto segura, nada seria possível sem a sua ajuda incondicional.

A todos que contribuíram de forma direita ou indireta para a realização e concretização dessa conquista muito obrigada.

Resumo

O crescimento das redes de computadores, aliado a redução de custos dos recursos computacionais, motivou a proliferação das redes e desencadeou no surgimento de aplicações, tais como as aplicações em modo *Peer-to-Peer* (P2P), que utilizam recursos distribuídos para executar funções de modo descentralizado. Estas aplicações, facilitam o transporte de conteúdos multimédia de características escaláveis, que se adaptam às condições da rede e do recetor, fazendo-os chegar a um número muito elevado de participantes com a melhor qualidade possível. A distribuição de informação em formato de vídeo sobre as redes baseadas em *Internet Protocol* (IP), quer pré-gravada quer transmitida em tempo-real, é cada vez mas frequente, tanto devido às melhorias de qualidade proporcionadas pelos *Codecs* mais recentes, como ao aumento de largura de banda das redes de acesso e também à capacidade de processamento dos dispositivos terminais. Estas aplicações onde a difusão de conteúdos é feita de forma cooperativa por nós participantes (tipicamente) anónimos, levanta preocupações a nível de segurança, nomeadamente, aspetos relacionados com a integridade e a confidencialidade dos conteúdos, bem como a autenticação e controlo de acesso dos vários intervenientes. Com este trabalho pretendemos efetuar um estudo abrangente dos mecanismos de segurança aplicáveis a sistemas de difusão multimédia baseados nas especificações normativas do protocolo *Peer-to-Peer Streaming Protocol* (PPSP) do *Internet Engineering Task Force* (IETF).

Palavras Chave

Redes *Peer-to-Peer* (P2P), conteúdos multimédia adaptativos, difusão de conteúdos, segurança da informação, *Peer-to-Peer Streaming Protocol* (PPSP).

Abstract

The growth of computer networks, allied with the reduction in costs of computational resources, motivated the proliferation of networks and triggered the emergence of applications, such as *Peer-to-Peer* (P2P) applications, that use distributed resources to perform functions in a decentralized way. These applications facilitate the transport of multimedia contents with scalable characteristics, which adapt to the network and receiver conditions, reaching a very high number of participants with the best possible quality. The distribution of information in video format over *Internet Protocol* (IP) based networks, whether pre-recorded or transmitted in real-time, is becoming more frequent, due to the quality improvements provided by the most recent Codecs, as well as to the increase of bandwidth on the access networks and also to the processing capacity of terminal devices. These applications where the diffusion of content is done in a cooperative way by the peer nodes anonymously (typical), raises security concerns, namely on aspects related with the integrity and confidentiality of the content, as well as the authentication and access control of the various stakeholders. With this work we intend to carry out a comprehensive study of the security mechanisms applicable to multimedia streaming systems based on the normative *Peer-to-Peer Streaming Protocol* (PPSP) protocol specifications from *Internet Engineering Task Force* (IETF).

Keywords

Networks *Peer-to-Peer* (P2P), adaptive multimedia content, content dissemination, information security, *Peer-to-Peer Streaming Protocol* (PPSP).

Conteúdo

1 Introdução **1**

1.1 Objetivos do trabalho 4

1.2 Organização do documento 4

2 Fundamentos **7**

2.1 Fundamentos das redes P2P e suas arquiteturas 9

2.2 Organização das redes sobrepostas 11

2.2.1 Arquiteturas estruturadas 12

2.2.2 Arquiteturas não estruturadas 13

2.2.3 Sistemas híbridos 14

2.3 Vulnerabilidades, ataques, riscos e defesas 15

2.3.1 Vulnerabilidades e ataques 16

2.3.2 Mecanismos de defesa e proteção de um sistema 18

3 Difusão Multimédia em P2P e Segurança **21**

3.1 Difusão multimédia em P2P 23

3.1.1 Operação de difusão multimédia em P2P 24

3.2 Segurança das aplicações P2P de *Live streaming* 24

3.2.1 Ataques de poluição de conteúdos 25

3.2.2 Ataques de *routing* 26

3.3 Autenticação em sistemas P2P de *Live Streaming* 27

3.3.1 Esquemas de amortização de assinaturas 28

3.3.2 Esquemas leves de assinaturas Digitais 28

3.3.3 Proteção de conteúdos por DRM 30

3.3.4 Tipos de proteção por DRM 31

3.3.5 Encriptação *HTTP Live Streaming* (HLS) 33

3.4 Arquiteturas de *Streaming* P2P 33

4 Mecanismos de Segurança em implementações baseadas em PPSP **37**
4.1 Protocolos *Standard* de *Streaming* em PPSP 39
4.2 Mecanismos de segurança aplicáveis ao Protocolo PPSP 41
4.2.1 Identificação e Autenticação dos *Peers* 41
4.2.2 Controlo de acesso 42
4.2.3 Integridade dos conteúdos 43
4.2.4 Confidencialidade dos conteúdos 44
4.2.5 Disponibilidade 44
4.2.6 Anonimato 45
4.2.7 Auditoria 46
4.2.8 Outras formas de segurança Aplicáveis ao Protocolo PPSP 46
4.3 Demonstração de um mecanismo de segurança em PPSP 47
4.3.1 Implementação funcional 48
4.3.2 Procedimento operacional 59
4.4 Avaliação operacional dos mecanismos 60
4.4.1 Negação de acesso 63
4.4.2 Identificador do *Peer* 64
4.4.3 Download do conteúdo de vídeo 66
5 Conclusão **67**
5.1 Discussão 69
5.2 Recomendações 69

Lista de Figuras

2.1 Estrutura de uma Rede P2P. 9
2.2 Estrutura de uma Rede Centralizada Cliente/Servidor. 11
2.3 Necessidade de se garantir a segurança de um sistema informação 16

3.1 Arquitetura SEA 34
3.2 Arquitetura NAPA-WINE 34
3.3 Arquitetura SARACEN 35

4.1 Componentes do sistema baseado em PPSP 40
4.2 Cenário do sistema PPSP Implementado (em execução) 49
4.3 Arquitetura de *Streaming* 58

Acrónimos

AAA	*Authentication, Authorization, Accounting*
AES	*Advanced Encryption Standard*
ALPS	*Authenticating Live P2P Streams*
BiBa	*Bins and Balls signature*
CDN	*Content Distribution Network*
CORE	*Common Open Research Emulator*
CPU	*Central Processing Unit*
DHT	*Distributed Hash Table*
DoS	*Denial of Service*
DRM	*Digital Rights Management*
HLS	*HTTP Live Streaming*
HORS	*Hash to Obtain Random Subset*
HTL	*hops-to-live*
HTTP	*Hypertext Transfer Protocol*
HTTPs	*Hyper Text Transfer Protocol Secure*
ID	*Identifier*
IDS	*Intrusion Detection System*
IETF	*Internet Engineering Task Force*
iOS	*iPhone Operating System*

IPTV *IP Television*

IP *Internet Protocol*

ISO *International Organization for Standardization*

MAC *Message Authentication Code*

MPD *Media Presentation Description*

NAPA-WINE *Network-Aware P2P-TV Application over Wise Networks*

P2P *Peer-to-Peer*

PARM *Polluion-Attack Resistant Multicast Autentication*

PEAC *Probabilistic, Efficient, and resilient Authentication protocol for broadcast Communications*

PPSP *Peer-to-Peer Streaming Protocol*

PPSTP *Peer-to-Peer Streaming Tracker Protocol*

PPSPP *Peer-to-Peer Streaming Peer Protocol*

QoE *Quality Of Experience*

RTMP *Real-Time Messaging Protocol*

SARACEN *Socially Aware, collaboRative, scAlable Coding media distributioN*

SEA *SEAmless Content Delivery*

SQL *Structured Query Language*

SSL Secure Socket Layer

SVC *Scalable Video Coding*

TLS *Transport Layer Security*

URL *Uniform Resource Locator*

UTANGA Universidade Técnica de Angola

VoD *Video on Demand*

VPN *Virtual Private Network*

XML *Extensible Markup Language*

1

Introdução

Conteúdo

1.1 Objetivos do trabalho 4
1.2 Organização do documento 4

A crescente evolução das telecomunicações e consequentemente das redes de computadores, permitiu a redução dos recursos computacionais, fato este que levou à proliferação e à expansão do uso das redes em todos os setores da sociedade. Isso trouxe consigo um aumento na diversidade de recursos e serviços oferecidos, o que, por sua vez, tem aumentado a complexidade das redes. Os sistemas computacionais por outro lado também apresentam grande heterogeneidade em termos de padrões de redes suportadas, sistemas operacionais, capacidade, etc. [1].

Nos últimos anos, tem-se também assistido a um enorme aumento de diferentes tipos de aplicações, em especial as baseadas em mecanismos da *Web* e que proporcionam quer a simples partilha de ficheiros quer a transmissão de vídeo em tempo-real, ou mesmo vídeo-conferência. Estas aplicações, ganharam mais interesse devido à sua flexibilidade e fácil acessibilidade.

Muitas dessas aplicações/serviços, com especial relevo, a distribuição/difusão em tempo-real de conteúdos multimédia, envolvem tipicamente uma fonte (servidor/produtor) e vários destinatários (recetores/consumidores). Devido à dificuldade no suporte para *Multicast* através da *Internet*, estas aplicações/serviços sofrem com os problemas de escalabilidade e de controle, o que limita o número de recetores envolvidos e o domínio geográfico da sua utilização.

Dadas essas limitações têm proliferado as redes *Peer-to-Peer* (P2P) como a tecnologia que permite implementar *Multicast* não na rede mas na camada de aplicação. Ao incorporar a tecnologia P2P em aplicativos para difusão em tempo-real de conteúdos, resolve-se o problema de escalabilidade da rede mas não necessariamente o de controle.

Outras razões pelas quais estes modelos de computação distribuído se tornaram atrativos, desdobram-se em três aspetos. Primeiro, as redes P2P são escaláveis, ou seja, lidam bem (eficientemente) tanto com grupos pequenos quanto com grupos muito grandes de participantes. Segundo, ao não possuírem tipicamente, um ponto central de falhas, resistem melhor quer à variação do número de participantes quer a ataques intencionais, tais como os de negação de serviço. Terceiro, as redes P2P oferecem autonomia a seus participantes, possibilitando que entrem e saiam da rede de acordo com seu interesse e disponibilidade, tomando as suas decisões sem depender de entidades terceiras [2].

Não tendo havido contudo um padrão normativo neste tipo de aplicativos P2P, nomeadamente para a distribuição/difusão em tempo-real de conteúdos multimédia, a sua heterogeneidade estrutural, mesmo naqueles que utilizam o protocolo globalmente mais utilizado, o *BitTorrent* tem ocasionado sérias dificuldades na integração e interoperação deste tipo de soluções.

Para colmatar essa lacuna, foi desenvolvido pelo *Internet Engineering Task Force* (IETF) um conjunto de protocolos normativos e abertos, designados *Peer-to-Peer Streaming Protocol* (PPSP), especificamente vocacionados para a distribuição/difusão em tempo-real de conteúdos multimédia [3].

Mantém-se porém em aberto, a questão da segurança neste tipo de rede e com estes protocolos, pois para além da integridade dos conteúdos estruturados transmitidos, da sua autenticidade e do

respetivo controlo de acesso, há também que preservar os aspetos de confidencialidade quer dos conteúdos quer dos participantes da sessão e a sua disponibilidade.

1.1 Objetivos do trabalho

O principal objetivo do trabalho, consiste no estudo sobre os mecanismos de segurança aplicáveis à difusão de conteúdos multimédia, adaptativos e escaláveis, em redes P2P, nomeadamente em sistemas implementados com base nas especificações normativas PPSP do IETF [3–5]. Com base no estudo, demonstra-se a implementação de um mecanismo de segurança, que permite a identificação, a autenticação e a autorização de um "*peer*" neste tipo de sistemas[1].

1.2 Organização do documento

Após terem sido feitas várias pesquisas a volta do assunto em questão, por várias obras publicadas e escritas por diferentes autores, fez-se uma análise dos mesmos conteúdos para podermos entender como os mecanismos de segurança são aplicados em redes P2P atualmente ou como devem ser aplicados, para resolver a questão da segurança nestes sistemas e posteriormente a sua implementação, baseadas nas especificações normativas do PPSP.

Nesta onda de ideias, o presente trabalho, apresenta estes assuntos, repartindo-os em cinco capítulos. Cada capítulo aborda um assunto específico, de acordo com a distribuição dos mesmos:

Capítulo 2: Fundamentos.

Neste capítulo, fez-se uma resenha, sobre alguns conceitos fundamentais para o desenvolvimento do trabalho, onde encontraremos uma descrição, dos fundamentos de rede P2P e suas características, entre outros assuntos que nos ajudaram a compreender melhor o funcionamento destes tipos de aplicações, fazendo referência ao modelo da arquitetura centralizada Cliente/Servidor, identificando, as caraterísticas que este modelo apresenta, em relação ao modelo P2P.

Capítulo 3: Difusão multimédia em P2P e segurança.

A segurança na transmissão dos conteúdos multimédia em redes de natureza P2P, apesar de ser ainda um desafio, para os pesquisadores, vários estudos, foram feitas e continuam sendo feitos, nesta área, com vista a contribuir para a resolução deste problema, contando, com a contribuição de vários autores. Neste ponto, mostramos estas contribuições, em termos do que existe, no que tange a segurança na partilha de dados usando o meio de transmissão em causa.

[1] O sistema P2P usado para demonstrar a implementação do mecanismo de segurança, teve como base o código utilizado pelos autores do RFC 7846 [3] no processo de desenvolvimento dessa especificação e no âmbito do Projeto Europeu SARACEN [6]

Capítulo 4: Mecanismos de Segurança em implementações baseadas em PPSP.

Nesta secção, apresentamos, os mecanismos de segurança aplicáveis em rede P2P, baseados nos protocolos PPSP, seus conceitos e técnicas, utilizadas para resolver os diferentes problemas que podem causar a falta de segurança e colocar em perigo o funcionamento do sistema e os recursos dos utilizadores. Neste capítulo apresentamos também uma demonstração de implementação de um mecanismo simples de segurança baseando-se na identidade e autenticação dos "*Peers*" no sistema.

Capítulo 5: Conclusão.

Finalmente, após um estudo árduo, no que tange aos mecanismos de segurança aplicados aos protocolos PPSP sobre as redes ou em sistemas P2P, fizemos uma breve descrição conclusiva, sobre as pesquisas realizadas, como um consenso final desta fase do trabalho.

2

Fundamentos

Conteúdo

2.1 Fundamentos das redes P2P e suas arquiteturas 9

2.2 Organização das redes sobrepostas 11

2.3 Vulnerabilidades, ataques, riscos e defesas 15

De forma geral, este capítulo explana, os fundamentos de uma rede P2P, também designadas por ≪redes sobrepostas≫ (*overlay* na terminologia inglesa). Ainda neste capítulo apresentamos os elementos e as caraterísticas das arquiteturas *overlay* estruturadas e não estruturadas, com alguns exemplos de infraestruturas que se enquadram em cada uma dessas arquiteturas, com enfoque nos princípios da segurança subjacentes, nas vulnerabilidades que um sistema de informação pode apresentar, e nos tipos de ataques que estes sistemas podem sofrem e quais as suas defesas.

2.1 Fundamentos das redes P2P e sua arquitetura

Uma rede P2P, é um sistema distribuído composto de nós interligados, em que cada nó funciona tanto como cliente, quanto como servidor, capazes de se auto-organizar em topologias de rede diversas, com objetivo de partilhar recursos, com capacidade de adaptação a falhas de rede e acomodação a um número variável de nós, simultaneamente, mantendo conectividade e o desempenho em níveis aceitáveis, sem a necessidade de suporte ou intermediação de uma coordenação central [7].

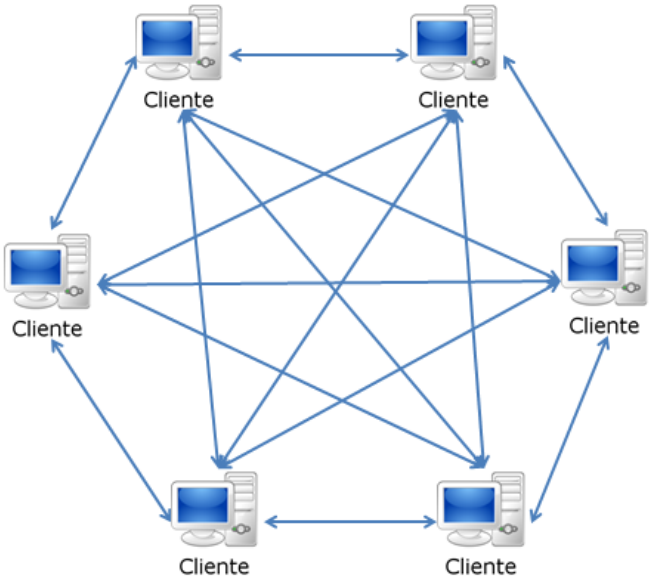

Figura 2.1: Estrutura de uma Rede P2P.

Por outro lado, considera-se um sistema como *P2P* se os elementos que formam o sistema, possuírem a capacidade de partilhar seus recursos, para fornecer os serviços que o sistema foi projetado para fornecer. Os elementos do sistema fornecem serviços para outros elementos e solicitar serviços de outros elementos.

Em princípio, todos os elementos do sistema devem atender ao anterior critérios para que o sistema possa ser considerado como P2P. No entanto, na prática, um sistema pode ter algumas exceções

e ainda ser considerado P2P. Por exemplo, um sistema P2P pode ainda ser considerado como tal, mesmo que tenha uma solução centralizada como um servidor.

Por outro lado, alguns sistemas dividem os pontos finais entre os *Peers* e os clientes. Os *Peers* solicitam e prestam serviços, enquanto que os clientes geralmente só solicitam serviços. Um sistema onde a maioria os pontos de extremidade comportaram-se apenas como clientes não podem ser considerados como P2P, os elementos que formam o sistema P2P que são chamados *Peers*, devem ser capazes de se comunicar diretamente entre eles sem passar por intermediários, e ser capaz de se auto-organizar para o seu funcionamento [8].

As redes P2P assentam numa arquitetura distribuída (Figura 2.1), contrastando com as topologias clássicas, cliente/servidor também designadas por centralizadas, que envolvem dois elementos:

1. **Servidor:**
    - Oferece serviços requisitados pelos clientes através da rede;
    - A comunicação entre ambos, é feita quando se recebe uma requisição.
2. **Cliente:**
    - Requisita um determinado serviço, fornecido por um servidor.
    - Inicia a interação com o servidor;
    - Fornece as interfaces de utilizador.
    - A execução é finalizada, após a sua utilização pelo utilizador final.

Um exemplo da arquitetura centralizada Cliente/Servidor é ilustrada na Figura 2.2, onde os vários clientes efetuam pedidos ao único servidor, que por sua vez responde aos clientes.

Numa arquitetura P2P, os nós podem desempenhar as duas funções, tanto cliente como servidor e cada nó possui autonomia para decidir quais os recursos que deve/pode partilhar com os restantes elementos da rede, e impor restrições de acesso.

A arquitetura P2P, apresenta características chave para distribuição de conteúdos na rede que são:

- Partilha direta de recursos entre os nós, sem intermediação de um servidor centralizado, embora se admita o uso de nós servidores em tarefas com menor exigência computacional ou comunicacional e não diretamente envolvidos na partilha de recursos;
- Capacidade de auto-organização tolerante a falhas, assumindo conectividade variável e conjunto transitório de nós que se adaptam automaticamente, tanto às falhas de conexões de rede, como dos próprios nós.
- Outra característica chave em P2P é o facto do sistema ser constituído com base na ≪colaboração≫ entre os nós participantes.

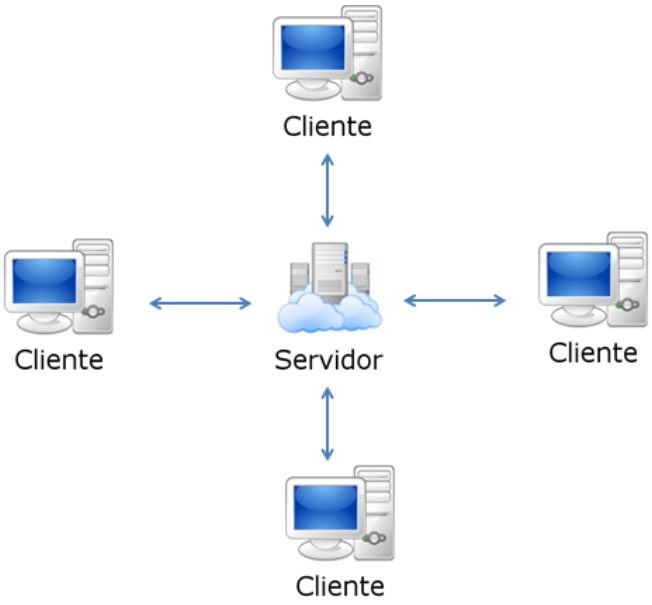

Figura 2.2: Estrutura de uma Rede Centralizada Cliente/Servidor.

Com base no exposto, e segundo [9], poderemos definir as Redes P2P como sistemas distribuídos constituídos por nós intercetados, capazes de se auto-organizar em topologias sobrepostas, com o objetivo de partilhar recursos, em formato de (áudio e vídeo), ciclos de *Central Processing Unit* (CPU), armazenamento e largura de banda, capazes de se adaptaram a conjuntos transitórios de nós mantendo conectividade e desempenho aceitáveis, sem a necessidade de intermediação ou apoio de uma entidade central.

2.2 Organização das redes sobrepostas

Podemos dividir os sistemas P2P em duas categorias principais, no que concerne ao *Overlay*: estruturado e não-estruturado.

Em sistemas não estruturados, a topologia é determinada de forma *ad hoc*, ou seja, à medida que os nós entram e saem do *Overlay*, estabelecem ligações arbitrárias com outros nós. O posicionamento de objetos ou serviços, é neste caso, completamente independente da topologia do *Overlay*. Uma dificuldade associada a este tipo de *Overlay* é o processo de busca de objetos ou serviços. Métodos básicos como a inundação (*flooding*) do *Overlay* (cada nó difunde a mensagem a todos os outros), foram utilizados nos primeiros sistemas P2P, tais como no *Gnutella* original. A ineficácia deste método conduziu ao desenvolvimento com base na escalabilidade. Daí, resultaram estratégias mais consistentes, tais como trajetória aleatória (*random walk*), índices de encaminhamento (*routing*) [10], e *Overlay* escaláveis via Tabelas de Espalhamento–*Distributed Hash Table*s (DHTs).

Nos sistemas estruturados, a topologia do *Overlay* está de acordo com o esquema de alocação de chaves a *Identifier*s (IDs) de nós, por forma a associar determinado objeto ou serviço a um nó específico de forma determinística e conhecida globalmente, no *Overlay*. As DHT funcionam como tabelas de *routing* distribuídas, permitindo que um objeto/serviço seja encontrado com um pequeno número de etapas.

2.2.1 Arquiteturas estruturadas

As redes P2P estruturadas, assentam numa topologia determinística. Os principais exemplos são os seguintes:

- **Chord**: Infraestrutura de *routing* em anel que utiliza *hash SHA-1*, para associar chaves de objetos aos nós, no espaço circular de IDs de nós de *m bits*. Os identificadores de nós são obtidos via *hash* do endereço *Internet Protocol* (IP), ao passo que as chaves de objetos são criadas a partir de um *hash* da descrição de objeto.

  Uma chave *k* é associada ao nó de identificador igual a *K*, ou se o mesmo não existe, ao próximo nó do anel, denominado nó sucessor. O *hashing* permite aos nós entrarem e saírem da rede causando pouca entropia no *overlay*. Quando um nó ingressa na rede depois do nó *n*, ele assume a responsabilidade sobre uma parcela das chaves que pertenciam a *n*. Cada nó mantêm um apontador para os *N* nós imediatamente sucessores e uma *finger table* com até *m* apontadores para outros nós (logaritmicamente espalhados no anel).

  O *routing* de procura é unidirecional ao longo do anel, podendo ser recursivo ou iterativo. No modo recursivo, a mensagem vai sendo encaminhada de nó em nó, aproximando-se do predecessor do objeto. Quando chega ao nó com o objeto, a busca retorna recursivamente à origem. No modo iterativo, o nó requisitante vai questionando os nós que ficam cada vez mas próximos do nó com o objeto. Quando o nó com o objeto é questionado, responde com os dados. A falha de nós não causa uma falha global podendo haver replicação de objetos em nós consecutivos [11].

- **CAN**: A ≪*Content-Addressable Network*≫ é uma infraestrutura descentralizada que assenta no uso de um espaço de coordenadas cartesianas virtual de *n* dimensões, numa topologia *Multi-Torus*. Trata-se de um espaço lógico que serve para a identificação dos nós e respetiva localização via tabelas de *routing* distribuído. Cada nó é responsável por uma zona do espaço, determinada dinamicamente, e mantém uma tabela de *routing* com o endereço IP e as coordenadas de cada um dos seus vizinhos no espaço. O protocolo de pesquisa, utiliza pares de chave de objeto para mapear um ponto *P* no espaço de coordenadas usando uma função *hash* uniforme, e coloca estas coordenas nas mensagens. Uma mensagem é encaminhada para o destino, usando uma

rota simples, até ao nó mas próximo das coordenadas. A falha de nós não causa uma falha global. Múltiplos nós são responsáveis por um objeto e quando há falha a aplicação tenta de novo [12].

- **Tapestry**: É uma infraestrutura P2P, que permite o encaminhamento de mensagens até ao objeto, de uma forma distribuída, auto-administrada e tolerante a falhas. As informações de *routing* e localização são distribuídas entre os nós da rede. A consistência da topologia é verificada dinamicamente e pode ser reconstruida em caso de perda. A *Tapestry* é baseada nos mecanismos de localização e de *routing* em malha. Esta estrutura distribuída, permite que os nós localizem os objetos na rede, usando pequenos mapas de *routing*. Na malha original os nós podem assumir papel de servidores (que armazenam objetos), *routers* (que encaminham mensagens) e clientes (que originam pedidos).

- **Pastry**: Tal como na *Tapestry*, a *Pastry* é uma infraestrutura de *routing*, baseada em malha. A principal diferença, está na fórmula de localização na rede e na replicação de objetos. A *Pastry* é utilizada pelo sistema de armazenamento persistente de larga escala, *PAST*, um sistema de comunicação em grupo e de comunicação de eventos.

- **Kademlia**: É uma infraestrutura de *routing*, para o encaminhamento de busca de objetos segundo uma métrica de distância entre identificadores de nós baseada em operação *XOR*. Nesta topologia, toda a mensagem trocada transporta ou reforça informações úteis de contacto. O sistema explora essa informação, para enviar mensagens de busca assíncronas e paralelas, que toleram falhas de nós sem impor atrasos nem *timeouts* aos utilizadores. Inúmeras aplicações P2P têm utilizado o algoritmo *Kademlia*, tais como: *Overnet*, *eDonkey*, *eMule*, além do *BitTorrent* que usa a *Kademlia* para proporcionar o uso de *torrent* sem um *Tracker* [13].

2.2.2 Arquiteturas não estruturadas

Exemplos de redes P2P, não estruturadas:

- **Naspter**: Foi uma aplicação muito importante, para a partilha de ficheiros, apesar de depender de um servidor central para o seu funcionamento.

- **Gnutella**: É um sistema de partilha de ficheiros em topologia *ad hoc*. Todos os nós são funcionalmente idênticos, denominados *servents*, por serem simultaneamente *servers* e *clients*.

- **FastTrack/KaZaa**: É um sistema de partilha de ficheiros, assente numa arquitetura de dois níveis, com nós normais e super nós. Os nós normais ligam-se a um super nó e os super nós interligam-se entre si. Um nó normal mantém uma lista de endereços de até 200 super nós, enquanto um super nó pode manter uma lista com milhares de super nós.

- **BitTorrent**: Sistema de partilha de ficheiros baseado em ≪enxames≫ (*swarms*) de nós, que permutam diretamente blocos de ficheiros tipicamente coordenados por um nó central, o *Tracker*. A lista de ficheiros e respetivas propriedades, bem como o endereço do (ou dos) *Tracker* responsável pelo *swarm*, são especificados através de um ficheiro denominado *torrent*. Este ficheiro é preparado pelo utilizador que deseje publicar conteúdo, podendo ser disponibilizado num *web site* especializado. Um *Torrent* tem que ser localizado por utilizadores interessados, via busca em *site* com *Torrents* ou através de motores de busca na *web*.

  Assim que tiver obtido o ficheiro *torrent*, o utilizador fornece-o ao *software* cliente, que então se liga ao *Tracker*. O nó informa o *Tracker* sobre o seu interesse naquele objeto descrito no *torrent* e este responde com uma lista aleatória de nós ativos no *swarm*. O nó contacta então múltiplos outros nós, solicitando blocos (*chunks*) tipicamente de 512 KB do conteúdo.

  O *BitTorrent* utiliza uma política de incentivos baseada no *toma-lá-dá-cá* (*tit-for-tat*) [14], tal que de início o nó possui um número de vizinhos no *swarm* que em princípio estão todos colocados no estado ≪sufocado≫ (*choked*), donde escolhe um número de nós (tipicamente 4), para os quais irá fazer *upload*. Na terminologia *BitTorrent*, esses nós serão então *unchocked*.

  A escolha baseia-se na taxa de *download* obtida dos vizinhos com quem o nó interage, e também num *unchoking* otimista, em que um outro nó (independentemente da sua taxa de download) é escolhido periodicamente de forma aleatória, dando assim oportunidade a novos nós para a cooperação na partilha. O *Tracker* monitoriza a disponibilidade dos nós e dos fragmentos do objeto dos mesmos. Os nós são testados periodicamente e um nó que não responde ou não coopera é sucessivamente trocado.

- **Overnet/eDonkey2000**: É uma arquitetura híbrida de duas camadas, composta por nós clientes e nós servidores, estes responsáveis por indexar informações sobre ficheiros e participar nas operações de busca. Tanto o cliente como o servidor são executados por qualquer utilizador.

- **Freenet**: É um sistema de partilha de ficheiros com garantia de anonimato. Baseia-se em DHT, usa palavras-chave e texto descritivo para identificar objetos. As buscas são efetuadas de nó em nó através de chaves ou *strings* contendo texto descritivo. O *routing* assegura a localização de objetos usando uma chave, até aos pedidos que excedam o limite de *hops-to-live* (HTL).

2.2.3 Sistemas híbridos

Nas estruturas híbridas, conjugam-se os mecanismos e construção dos *Overlay* estruturado e não estruturado por forma a obter otimizações funcionais.

Um dos exemplos mais conhecidos e usados é o próprio *BitTorrent*. O ficheiro *torrent* é aberto no *software* cliente, que então poderá obter os endereços dos *Tracker* que controlam o *swarm*, e nesta

situação o nó constrói um *Overlay* malhado e não estruturado de outros nós, ou então obtém os ID numa DHT, fazendo assim (também) uma construção de *Overlay* estruturado.

2.3 Vulnerabilidades, ataques, riscos e defesas

A segurança da informação, é um mecanismo utilizado com a finalidade de proteger os ativos de um sistema. No âmbito da segurança de um sistema, podem-se considerar três grandes áreas de atividade, todas relevantes e com as suas especificidades: defesa contra catástrofes, defesas contra falhas previsíveis e defesa contra atividades não autorizadas.

Um dos motivos que causa a falta de segurança de um sistema informático, é a **vulnerabilidade**. Pode-se considerar a vulnerabilidade como uma característica de um sistema, que o torna sensível a certos ataques. A vulnerabilidade que os sistema apresentam, permitem acesso de certos ataques, causando a insegurança desse sistema.

Considera-se um **ataque**, o conjunto de passos executados no âmbito da exploração de vulnerabilidades, permitindo a concretização de ações ilícitas. Um **risco** ou **ameaça**, é o dano resultante da execução bem-sucedida de um ataque. Por último, considera-se de **defesa**, o conjunto de políticas e mecanismos definidos de modo a impedir ações ilícitas para:

- Diminuir as vulnerabilidades de um sistema;
- Detetar, controlar e anular ataques passados ou atuais;
- Minimizar os riscos decorrentes de ataques bem-sucedidos.

Um sistema considerado seguro deve contemplar as três vertentes referidas e não parte delas. Deve-se considerar que os ataques podem ter sempre sucesso, o que torna fundamental a criação de planos de contingência por forma a minimizar os riscos causados por ações ilícitas aos sistemas. O plano de contingência deve garantir a recuperação de dados ou serviços, que tenham sido causados por eventuais ações ilícitas dos atacantes ou intrusos num sistema de informação.

Para melhor perceção e compreensão da necessidade de se garantir a segurança num sistema de informação, podemos resumir usando o diagrama de conceitos relacional ilustrado na Figura 2.3.

A necessidade de se garantir a segurança num sistema de informação está relacionado com diversos aspetos que podem afetar direta ou indiretamente os vários elementos relacionados com um sistema de informação.

A segurança é uma atividade dedicada à proteção de ativos, contra acessos não autorizados, alterações indevidos ou indisponibilidade das informações pertencentes a um individuo ou empresa, viabilizada por meios de políticas e procedimentos de segurança que permitam a identificação e controlo

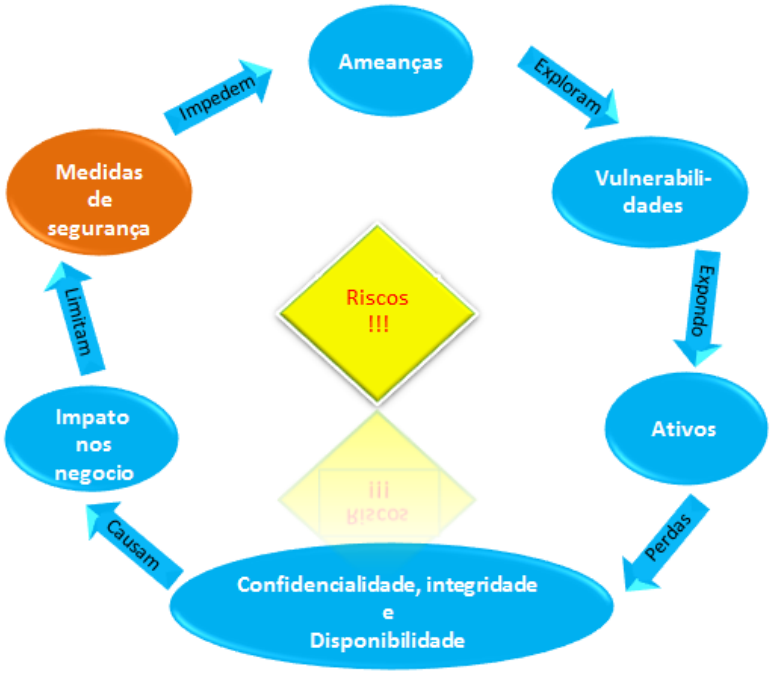

Figura 2.3: Necessidade de se garantir a segurança de um sistema informação

de ameaças e vulnerabilidades, visando a preservação da confiabilidade, integridade e da disponibilidade de um sistema de informação.

2.3.1 Vulnerabilidades e ataques

Para executar alguma ação maliciosa, num determinado sistema de informação, os *hackers*, utilizam formas de ataques, que exploram as vulnerabilidades que os sistemas apresentam [15]:

1. ***Hacking**:* O *Hacking*[1], é um tipo de ataque, que permite o acesso não autorizado, a um determinado sistema de informação, na tentativa de recolha de informações privadas pertencentes a outrem. O *Hacker* faz isso explorando a vulnerabilidade ou fraqueza de um sistema de destino.

   Um *Hacker*, tem a capacidade de usar as técnicas de inoculação de vírus e *worms*[2], no entanto, o *Hacker* pode evitar a deteção de *Intrusion Detection System* (IDS) ao disfarçar habilmente o ataque.

2. **Ataques de nível de aplicação:** Este tipo de ataque ocorre porque hoje, por um lado, os desenvolvedores de *software* na maior parte das vezes, trabalham sob pressão, para entregar produtos a tempo, por outro lado, a demanda por programação extrema dentro da metodologia de enge-

[1]Hacking: é um comportamento, que os *hackers* apresentam, na tentativa de invadirem um sistema informático, enganando as suas características de segurança
[2]Worm: é um programa auto-replicante, diferente de um vírus que não necessita de outro programa para se propagar [15].

nharia de *software*, cria problemas no cumprimento de todas as fases que seriam requeridas. Todas estas necessidades, fazem com que por vezes, não seja possível realizar todos os testes necessário afim de descobrir vulnerabilidades, antes da entrega do produto final.

3. **Ataques de Sistema Operativo**: Por causa da complexidade que os sistemas apresentam hoje, com o funcionamento de diversos serviços com diferentes formas/permissões de acesso, permite a possibilidade de violações de segurança, tornando-se difícil garantir que todos os serviços sejam seguros, dado ser uma tarefa dispendiosa e extremamente difícil, todo o processo de manutenção da grande variedade de serviços.

4. ***SQL injection***: Neste tipo de ataque, o invasor, injeta código *Structured Query Language* (SQL), através de uma caixa de entrada num formulário da *Web*, para obter acesso a recursos ou fazer alterações nos dados. O invasor injeta comandos SQL, para explorar vulnerabilidades de entrada não validadas em componentes de *Backend* do sistema (tipicamente uma base de dados de aplicativos *Web*), e consequentemente executar comandos arbitrários. Como os programadores utilizam comandos sequenciais com a entrada do utilizador, torna mais fácil para os invasores injetarem estes comandos.

5. ***Phishing***: Esta é uma forma de tentar adquirir informações, tais como nomes de utilizador, senhas e detalhes de cartão de crédito, mascarando-se como uma entidade confiável numa comunicação eletrónica. As comunicações que ≪pretendem≫ ser de *sites* sociais populares, *sites* de leilão, processadores de pagamentos *on-line*, por serem réplicas quase perfeitas desses *sites*, são comummente usadas para atrair o público desavisado que, consequentemente, é iludido e acaba por dar informações pessoais valiosas a terceiros, sem perceber que o está a fazer.

6. **Engenharia social**: Este é o lado humano de inadvertidamente permitir a entrada numa rede corporativa. Empresas com processos de autenticação, *firewalls*, *Virtual Private Network*s (VPNs) e *Software* de monitorização de rede podem estar abertas a ataques, por exemplo, quando um funcionário transmite informações-chave num *e-mail*, ou responde a perguntas por telefone com alguém que não conhece, ou mesmo ao falar sobre um projeto com os colegas de trabalho num bar local após as horas de trabalho.

   É o truque de obter informações sensíveis explorando aspetos da natureza humana básica, tais como confiança, medo e desejo de ajudar. Estas técnicas de engenharia social tentam tipicamente coletar informações confidenciais, de autorização e de detalhes de acesso. A engenharia social é a forma mais difícil de atacar, porque não pode ser defendida com *hardware* ou *software* e porque as pessoas são o elo mais fraco na cadeia de segurança, uma defesa para ser bem sucedida terá que ter boas políticas e a educação dos funcionários.

7. ***Trojan***: O *Trojan*, é um aplicativo mal-intencionado que não se consegue espalhar por si próprio, sendo transportado disfarçadamente num outro (aplicativo ou ficheiro de utilizador aparentemente confiável) que quando executado ou aberto permite a execução do *Trojan*. Historicamente, o termo tem sido utilizado para se referir a aplicativos que parecem legítimos e úteis, mas realizar atividades maliciosas e ilícitas num computador afetado.

2.3.2 Mecanismos de defesa e proteção de um sistema

DE uma forma não exaustiva, descrevem-se em seguida os principais mecanismos de defesa e proteção que devem ser considerados.

1. **Contra catástrofes físicas:** Este tipo de defesa objetiva-se, permitir que um sistema computacional, ou serviço que esse sistema presta, consiga sobreviver a catástrofes onde existam consequências ao nível físico. As catástrofes físicas podem ser resumidas em três exemplos comuns:

   - Catástrofes ambientais: tremores de terra, incêndios, inundações, quedas de raios, tempestades magnéticas, etc;
   - Catástrofes políticas: ataques terroristas, motins, etc;
   - Catástrofes materiais: degradação irreparável ou perda/ roubo de equipamentos computacionais, tais como discos magnéticos, computadores portáteis, etc.

   Tais catástrofes, constituem causas de dano físico irreparável de equipamentos informáticos, e também potenciais causas de perda irreparável de informação armazenada.

   Para garantir ou assegurar a sobrevivência, pode-se usar *hardware* com redundância ou equipamentos redundantes com informação replicada (como por exemplo, dois sistemas iguais e com a mesma informação em locais diferentes e suficientemente distantes). Também uma forma de assegurar a informação consiste na realização periódica de cópias de salvaguarda (*backup*).

2. **Defesa contra falhas previsíveis:** Este tipo de defesa, minimiza o impacto de problemas que ocorrem com uma frequência maior, cujo impacto global é normalmente menor. Definir exatamente o que são falhas não é fácil, mas são consideradas tipicamente as seguintes:

   - Quebra no fornecimento de energia elétrica ou falha na fonte de alimentação de um equipamento computacional;
   - Bloqueio na execução de aplicações ou sistemas operativos;
   - Falhas temporais de conectividade em troços de rede.

Estes três itens representam o conjunto de eventos expectáveis no funcionamento diário de um sistema computacional.

Uma das soluções concernente as falhas no fornecimento de energia elétrica podem ser mitigada com sistemas alternativos funcionando com baterias ou com geradores a combustíveis e as falhas nas fontes de alimentação podem ser resolvidas com fonte redundantes.

Para resolver o problema de bloqueio de aplicações ou sistemas operativos deve-se recorrer a conjuntos de máquinas, organizadas por exemplo em *cluster*, o que obriga a reiniciar a aplicação ou o sistema bloqueado. Os exemplos típicos de bloqueios de sistemas operativos são os ≪ecrãs azuis≫ dos sistemas *Windows* e as mensagens de pânico dos sistemas *UNIX/Linux*. O reinício inesperado e forçado do sistema causa falha na persistência da informação do sistema computacional, causando a incoerência da informação, e para se evitar esta incoerência em informações críticas são usados sistemas transacionais, por forma a garantir que a transformação da informação guardada se faz de forma coerente.

A falha temporal de conectividade em troços de redes pode ser tolerada usando troços e caminhos alternativos para a informação. As informações na *Internet*, fluem de um ponto para outro isto é de uma origem para um destino, utilizando vários caminhos alternativos, e em caso de falhas de um caminho podem-se usar outros, garantindo que a informação possa chegar ao seu destinatário a qualquer altura, usando vários caminhos alternativos.

3. **Defesa contra atividades não autorizadas:** Proteger as informações conta ações não autorizadas é um dos objetivos subjacentes na realização deste trabalho, utilizando mecanismos que permitam proteger o sistema, adotando medidas para evitar a concretização de ameaças que podem afetar o sistema de informação. Para tal, diversas medidas são adotadas com a finalidade de salvaguardar informação em ambos sentidos tais como:

   - **Acesso a informação:** são levados em consideração todos os aspetos relacionados com o acesso a informações reservadas ou confidencias, impedindo que as mesmas sejam tornadas públicas, guardadas em sistemas computacionais ou em trânsito.
   - **Alteração de informação:** consideram-se neste caso todas as atividades que de qualquer forma alterem ou eliminem sem autorização as informações pertencentes a terceiros.
   - **Utilização exagerada ou abusiva de recursos computacionais:** diversos problemas relacionados com a disponibilidade, podem estar relacionados com este tipo de atividade ilícita, na defesa contra o acesso não autorizado de informação, causado pela utilização exagerada ou abusiva de recursos. Isto porque, os recursos computacionais podem ser diversos tipos como:
     - Tempo de processamento;

  - Memória primária ou secundária;
  - Tempo e material de impressão;
  - Ocupação de redes de comunicação etc.

  Estes recursos possuem uma gama de utilização limitada, onde o excesso, o exagero ou a utilização não controlada dos mesmos culminam em problemas que afetam a disponibilidade do sistema.

- **Impedimento de prestação de serviço – *Denial of Service* (DoS):** O impedimento de prestação de serviço é um caso extremo do uso excessivo ou abusivo de recursos computacionais. Considera-se assim porque o principal objetivo da atividade ilícita é apenas impedir que terceiros tenham acesso aos recursos afetados sem qualquer beneficio próprio dos mesmos.
- **Vandalismo:** um dos casos extremos de ataques DoS, esta diretamente relacionado com a interferência com o funcionamento normal do sistema sem qualquer beneficio para o sujeito causador.

3

Difusão Multimédia em P2P e Segurança

Conteúdo

3.1 Difusão multimédia em P2P 23

3.2 Segurança das aplicações P2P de *Live streaming* 24

3.3 Autenticação em sistemas P2P de *Live Streaming* 27

3.4 Arquiteturas de *Streaming* P2P 33

Neste capítulo, apresentamos trabalhos relevantes relacionados com a segurança de informações em sistemas P2P, que tratam aspetos de segurança tais como autenticação e controle de acesso.

3.1 Difusão multimédia em P2P

Vários estudos tem sido feitos, investigando problemas relacionados com a segurança em redes P2P, de maneira contribuir com o estudo e implementação de um mecanismo de segurança nestas aplicações.

Nos sistemas de difusão (*Streaming*) em P2P todos os nós deverão cooperar na replicação de *Streams*, alocando alguns dos seus próprios recursos ao sistema, tal como a largura de banda para *upload* de *Streams* em si residentes, para propagação a outros nós. É também assumido que cada *Stream* corresponde ao fluxo contínuo de dados multimédia codificados pela fonte de *Streaming*. A fonte necessita de subdividir este fluxo em segmentos menores (*Chunks*), que correspondem tipicamente a alguns poucos segundos de *Streaming*, ou a determinado tamanho físico (bloco) de *bytes*, identificado no ficheiro de meta-dados correspondente. Estes *Chunks* (ou partes de si, designadas por *Pieces*) correspondem a unidades de *Stream* que serão propagadas e disponibilizadas aos *Peers* para replicação do *Stream*. Existem dois cenários aplicacionais típicos para P2P *Streaming*:

1. *Live* (difusão em tempo-real): Quando os nós aderem à rede podem apenas aceder ao que está a ser transmitido nesse momento.

2. *Video on Demand* (VoD) (vídeo a pedido): Quando os nós aderem à rede podem aceder a todo o conteúdo, bem como interagir com o mesmo, navegando para quaisquer pontos temporais.

Para serviços como *IP Television* (IPTV), o VoD é um elemento chave, dado permitir a visualização de qualquer parte específica de vídeo, em qualquer momento. No entanto o P2P em VoD é mais difícil de implementar que o serviço P2P *Live Streaming*. Enquanto num cenário ao vivo um grande número de utilizadores segue o mesmo *Stream*, de uma forma quase sincronizada, no VoD os utilizadores podem seguir o mesmo *Stream* mas temporalmente de uma forma assíncrona, entre si [16].

As arquiteturas para P2P media *Streaming* adotam tipicamente as topologias em árvore (*Tree*) e em malha (*Mesh*).

Na topologia de árvore, são formadas árvores *Multicast*, em que o media *Bitstream* é descarregado ao longo da árvore, encaminhado a partir da fonte do media. As árvores *Multicast* são construídas ao nível da camada aplicacional.

Na topologia em malha não existem caminhos estruturados e os *Peers* organizam-se por si próprios em enxames (*Swarms*), para partilharem o conteúdo. Nesta aproximação cada nó mantém uma lista de *Peers* com quem troca informação sobre a disponibilidade de dados, e a quem recorre para acesso a conteúdos.

A topologia de árvore tem sido utilizada sobretudo em aplicações comerciais, tais como o *PeerCast*. Mas as soluções mais populares de P2P *Streaming* assentam na topologia em malha, tais como a *Sop-Cast* e a *PPLive*.

3.1.1 Operação de difusão multimédia em P2P

As redes P2P, possuem maior eficiência, para a distribuição de conteúdos multimédia, através da *Internet*, do mesmo conteúdo para um grande número de utilizadores, ao mesmo tempo.

Na tecnologia P2P o mesmo conteúdo é armazenado num largo número de dispositivos homólogos e partilhada de forma cooperativa entre eles. As operação de uma rede P2P podem, em geral, ser descrita como se segue, a partir do ponto de vista do utilizador e, do ponto de vista da rede *Overlay*:

- **Ponto de vista do utilizador:** Um *software* de aplicação de *downloads* P2P é instalado a partir de um *site* na *Internet* para um computador pessoal que está ligado através de uma ligação de banda larga. Após a inicialização da aplicação P2P, ele tenta conectar-se a determinados *hosts* na *Internet* (geralmente pré-configurado na aplicação) para efetuar o designado *bootstrapping*. A aplicação P2P usa estes anfitriões para encontrar outros *Peers* para conectar-se (juntar-se ao *Overlay*)

- **Ponto de vista do *Overlay*:** Para qualquer tipo de rede *Overlay*, inicialmente, podem existir apenas alguns dispositivos de computação na *Internet* disponíveis para formar a rede P2P. Estes dispositivos precisam de alguma forma de descobrir o outro dispositivo e formar o *Overlay* inicial, isto é, efetuar o *Bootstrap* do *Overlay*. O mecanismo de *Bootstrap* permite que um novo *Peer* se possa juntar para poder identificar um ponto no *Overlay*, a fim de emitir um pedido de adesão, visto que o novo *Peer*, ao não fazer ainda parte do *Overlay*, não pode utilizar os mecanismos de pesquisa, para localizar outros *Peer* vizinhos.

3.2 Segurança das aplicações P2P de *Live streaming*

O uso de aplicações de *live streaming* que permitem a transmissão de áudio e vídeo em tempo real, de uma fonte de dados a vários recetores em redes P2P, já é uma realidade e dominarão o mercado nos próximos anos. A segurança dos dados no que tange a autenticação dos mesmo é um problema desafiador, o que faz com que estas aplicações sofram com o problema de ataques, como os conhecidos ataques de poluição dos conteúdos e ataques de *routing*.

Tais problemas de ataques, surgem devido a vulnerabilidades que os sistemas de informação apresentam.

Existem quatro vulnerabilidades fundamentais que são exploradas por atacantes em sistemas P2P de *Live Streaming* devido a ausência de controle ao nível de:

- acesso a rede *Overlay*;
- comportamento dos *Peers*;
- incentivo a cooperação entre os *Peers*;
- autenticidade e integridade dos *Chunks*.

Em ausência de controlo no acesso a rede *Overlay*, um *Peer* malicioso pode interferir no funcionamento de construção do *Overlay* e na gestão dos participantes da rede. Uma conhecida ameaça proveniente desta vulnerabilidade é o ataque *eclipse* [1].

A nomenclatura deste ataque deve-se ao facto de um ataque bem sucedido ser capaz de esconder um segmento de rede de outro segmento. Neste tipo de ataque a construção da rede *Overlay* é comprometida, e o efeito é tanto mais devastador quanto maior for a combinação entre nós maliciosos.

A falta de controle no comportamento do *Peer* possibilita que ataques de DoS sejam facilitados, dado que os *Peers* não maliciosos podem ser sobrecarregados com dados inválidos ou com pedidos excessivos de *Chunks*. Como consequência a disponibilidade dos dados pode ser comprometida e a qualidade do vídeo recebido pode ser muito má. Além disso, não existindo cooperação entre os *Peer* é possível que alguns nós decidam não contribuir mais para o sistema; um ataque que explora esta vulnerabilidade é o ataque de **omissão** [17].

A a última vulnerabilidade é a ausência de autenticação e integridade dos *Chunks* que são gerados e transmitidos originalmente pela fonte de dados. Isto proporciona brechas de segurança necessárias para a implantação de **ataques de poluição de conteúdos**.

3.2.1 Ataques de poluição de conteúdos

A poluição de conteúdos em *Live streaming* é um ataque que pode ter um efeito devastador num sistema P2P de *Live Streaming*, [18]. Parte-se do pressuposto, que os participantes na rede P2P de *Live Ltreaming* são de três tipos principais:

1. **Fonte**: entidade única que codifica o vídeo e o transmite originalmente aos demais *Peers*; pressupõe-se que a fonte de dados é sempre confiável
2. **Recetores**: *Peers* altruístas (executam o protocolo de forma correta), interessados em receber ou difundir os dados transmitidos.

[1] Ataques eclipse: um ataque eclipse consiste em fazer com que um *peer* correto seja rodeado por *peers* maliciosos, de forma que toda a sua comunicação possa ser intercetada isolando ou levando o mesmo a crer que está corretamente conectado com o resto da rede.

3. **Poluidores ou atacantes**: desviam-se da especificação do protocolo de forma arbitrária ou racional (desviam-se da especificação do protocolo em busca de benefícios próprios), que pretendem distribuir conteúdos poluídos.

A criação de mecanismos que reduzam os níveis de poluição é um desafio bastante relevante aos olhos da comunidade científica.

Desta feita, muitos estudos tem sido feitos nesta área, por vários autores, com vista a desenvolver mecanismos para solucionar tal situação que na essência assentam numa estratégia de autenticação de dados, baseada no uso de assinaturas digitais, para evitar, ou diminuir o impacto desses ataques, detetando os conteúdos poluídos.

Assim, o primeiro passo que deve ser dado, e que se torna o alicerce para se poder considerar um sistema de informação seguro é utilizando o método de autenticação dos dados, em que os dados transmitidos são assinados com uma chave privada a partir da fonte, permitindo que os recetores verifiquem a autenticidade dos dados recebidos através de uma chave pública da fonte, esta chave é obtida de forma segura pelos seus destinatários. Assim, os *Peers* recetores poderão descartar os dados falsos ou corrompidos enviados pelos Peers maliciosos, de formas a evitar ou a diminuir o impacto desse ataque. Resumindo, a detenção de conteúdos poluídos é feita usando o meto de autenticação por assinatura digitais, aplicadas com uma função de *Hash* sob criptografia.

3.2.2 Ataques de *routing*

Este tipo de ataques, significam, anomalias de *Routing*, em que mensagens transmitidas através do *Overlay* são desviadas para longe do seu destino para nós maliciosos, ou mesmo descartadas.

Tais ataques aumentam a probabilidade de falhas nas buscas, e têm um impacto negativo no custo e desempenho do sistema. Para que este tipo de ataque ocorra, as informações de *Routing* ou sobre outros nós, em nós corretos, são envenenados por nós maliciosos. Isto ocorre quando por exemplo, nós maliciosos respondem com rotas incorretas às mensagens de busca.

Segundo [19], temos 3 grande categorias de ataques de *Routing*:

1. *Routing* incorreto de procura: quando um nó malicioso encaminha mensagens de busca para um nó malicioso ou inexistente;

2. Atualizações incorretas de rotas: geradas por nós maliciosos que fornecem falsas informações sobre rotas, a nós corretos;

3. Alteração do *Overlay*: que pode ocorrer quando um nó correto ao entrar no *Overlay*, carrega a sua tabela de rotas a partir de um nó malicioso de *Bootstrap*, criando uma partição separada, controlada pelos nós maliciosos, mas contendo nós corretos.

De acordo com [20], são referenciados ataques a mecanismos de *Routing* que procuram assumir o controlo da comunicação do *Overlay* num ataque de tipo *Eclipse*. O atacante possui um grande número de nós, ou de identificadores no *Overlay*, de tal forma que os corretos são *eclipsados* pelos maliciosos e o *Overlay* deixa de funcionar corretamente.

3.3 Autenticação em sistemas P2P de *Live Streaming*

A autenticação de dados em sistema de P2P *Live Streaming*, surge como solução para não adulteração dos conteúdos por *Peers* maliciosos, que poderiam alterar o valor dos *chunks* recebidos antes que estes fossem partilhados aos *Peers* vizinhos da rede. Para tal, diversas abordagens existem como solução para estes males:

- Utilização de algoritmos clássicos de criptografia assimétrica;
- Esquemas baseados em *Message Authentication Code* (MAC);
- Esquemas leves de assinatura.

Para garantir a autenticação dos dados recebidos, através da distribuição de uma chave pública entre os recetores, usa-se o método de **algoritmos clássicos** de criptografia assimétrica, reservando uma chave privada na fonte, (exemplo do RSA) [2]. Este método, apesar de fornecer a autenticação de dados através do uso da complexidade matemática, representa uma sobrecarga que não permite o seu uso direito em sistemas de *Live streaming* devido às restrições temporais do processo de *streaming*.

Os esquemas baseados em MAC, permitem a autenticação de dados entre uma fonte e um recetor a partir de uma chave privada partilhada entre os mesmos. Como usam os algoritmos simétricos, são mais rápidos. No entanto este tipo de solução permite que recetores também possam assinar *Chunks*, o que permitiria a um atacante adulterar um *Chunk* e assinar o mesmo com a chave partilhada. Desta forma os MACs não podem ser utilizados para difusão de dados em sistemas nos quais os recetores não sejam confiáveis, tais como os de P2P *Live Streaming* [21].

Além das soluções descritas anteriormente, existem também, outros métodos que garantem maior eficiência ao contrário dos métodos tradicionais, através da diminuição das sobrecargas proporcionadas, tais como (atraso, tamanho de assinatura, custo computacional, etc.) Podendo com isto ser utilizadas por sistemas cujos requisitos sejam diferenciados (baixa latência ou alta largura de banda). As principais soluções de segurança, sobretudo ao nível de autenticação, em *Live streaming* são baseadas em duas abordagens principais que são os **esquemas de amortização de assinatura** e os **Esquemas leves de Assinaturas Digitais**.

[2] RSA é um algoritmo de criptografia de dados, que deve o seu nome a três professores do Instituto de Tecnologia de Massachusetts (MIT), RonaldRivest, Adi Shamir e Leonard Adleman. (Foi o primeiro algoritmo a possibilitar criptografia e assinatura digital, e uma das grandes inovações em criptografia de chave pública)

3.3.1 Esquemas de amortização de assinaturas

Os esquemas de amortização de assinatura, consistem na amortização das sobrecargas ao longo do tempo [22], para tal gera-se uma única assinatura para um bloco de *chunks* em vez de se gerar uma assinatura para cada *chunk*, o que nos proporciona a otimização no custo de geração e na conferência de assinaturas, em função da quantidade de dados.

Para se evitar a manipulação dos *chunks* por *peers* intermediários, é possível fazer a assinatura de cada *chunk* diretamente com a chave privada da fonte. Mas este tipo de solução apesar de segura, significa grandes sobrecargas e custos.

Uma das formas de otimização foi o desenvolvimento de um esquema bastante simples de assinaturas, no qual deve ser aplicada uma função de *hash* sobre cada *chunk* que será enviado, e o resultado deste *hash*, deve ser codificado com a chave privada da fonte, gerando assim, o respetivo *Digest*. O *Digest* deve ser enviado juntamente com o *Chunk* correspondente. Assim os recetores podem verificar a autenticidade dos dados recebidos, através da comparação entre o *hash* sobre os dados e a descodificação do *Digest*.

A coincidência entre eles, indica que os dados recebidos são válidos; caso contrário os dados não deveram ser difundidos, pois foram comprometidos ou corrompidos. Este esquema é designado por ***Hash Chaining*** .

3.3.2 Esquemas leves de assinaturas Digitais

Os esquemas leves de assinatura digitais, proporcionam a redução dos custos computacionais através da utilização de funções unidirecionais de *hash* para a criação e a verificação de assinaturas.

Os algoritmos anteriormente descritos amortizam a sobrecarga de funções convencionais de assinatura digital. Os esquemas leves de assinaturas digital utilizam as funções de *hash* de menor custo computacional que as funções convencionais. Pertencem a esta família, nomeadamente os esquemas *Bins and Balls signature* (BiBa), *Hash to Obtain Random Subset* (HORS), *Polluion-Attack Resistant Multicast Autentication* (PARM) ,*Polluion-Attack Resistant Multicast Autentication* (PARM)2, *Authenticating Live P2P Streams* (ALPS):

- ***Bins and Balls signature* (BiBa):** É um protocolo de autenticação de *Broadcast* que utiliza um esquema de assinatura digital para fazer a verificação da origem de uma sequência de *chunks* [23]. Este método, baseia-se na tentativa de encontrar colisões para criação de assinaturas. Uma colisão corresponde a sinalização de duas evidências (*Balls*) que correspondem ao mesmo valor (*Bins*) através de uma função aleatória de *hash*.

  O principio do funcionamento do esquema de assinatura BiBa compreende 3 fases:

  1. Criação de evidências pela fonte

2. Procura de K colisões (criação da assinatura)
3. Verificação da assinatura no recetor.

O esquema de assinatura *BiBa* tem por objetivo proporcionar assinaturas menores e um processo de verificação mas rápido que os esquemas referidos anteriormente.

- ***Probabilistic, Efficient, and resilient Authentication protocol for broadcast Communications* (PEAC):** O PEAC é um protocolo probabilístico de autenticação baseada em funções unidirecionais de *hash* que pode ser usado para *brodcast* em redes *wireless*, comunicação via satélite ou *IP Multicast* [24].

  O PEAC apresenta quatro momentos:

  1. inicialização pela fonte;
  2. seleção das evidências por *chunk* pela fonte;
  3. verificação da autenticidade das evidências;
  4. renovação temporal de chaves.

- ***Polluion-Attack Resistant Multicast Autentication* (PARM):** O PARM é um esquema de autenticação de conteúdos baseados em função de *hash* unidirecionais para a aplicação *multicast*, com o grande propósito de ser rápido, resistente a ataques de poluição e a ataques de negação de serviço.

  O PARM pode ser usado conjuntamente com o protocolo *SAIDA*, o sistema de amortização de assinatura desenhado para utilização em ambientes sujeitos a altas percas de *chunks*, através de esquemas de codificação. O esquema baseia-se na criação de evidências, que são verificadas *chunk* à *chunk* pelos recetores.

  O principio de funcionamento do PARM é o seguinte:

  1. inicialização na fonte de dados;
  2. geração de evidências por *chunk* pela fonte;
  3. validação de evidências por *chunk* no recetor;
  4. renovação parcial de chaves.

  De acordo com [25], um dos procedimentos essenciais do funcionamento do PARM é a renovação parcial de chaves, para que a comunicação entre a fonte e os recetores continue segura.

  Diversas propostas foram feitas por [26], para a melhorar o esquema de assinatura PARM, que resultou no surgimento de uma nova versão denominada por PARM2. Este novo modelo teve as seguintes alterações:

  - Os recetores não utilizam tabela de uso para permitir o uso de uma camada de transporte subjacente sem entrega ordenada de pacotes. Para tal, a fonte inclui em cada *chunk*, o numero de *hashes* para chegar até a chave pública (equivalente aos valores da tabela de uso) e um número de sequência;
  - Na tentativa de se reduzir o tempo de verificação da assinatura, os recetores utilizam w *buffers* contendo evidências recentes recebidas.
  - e por fim, quando a metade de suas evidências tenham sido enviadas, a fonte renova uma coluna de *hashes*.

- ***Authenticating Live P2P Streams* (ALPS):** O ALPS é um esquema de assinatura digital desenvolvido para autenticação de *stream*, em aplicações *P2P* de *Live streaming* [27].

  Esta proposta surge com a finalidade de ultrapassar algumas lacunas, tais como atrasos, tamanho da assinatura e complexidade computacional, impostas por abordagens baseadas em esquemas clássicos de assinatura digital (tais como RSA) e por esquemas de assinaturas anteriormente propostos, tais como BiBa.

  Os princípios fundamentais do ALPS são os seguintes:

  1. inicialização pela fonte (criação da chave privada, matriz de evidências e chave pública);
  2. seleção das colunas das evidências na matriz pela fonte;
  3. seleção das linhas das evidencias na matriz pela fonte;
  4. atualização dos índices das sub-matrizes tendo como base as colunas selecionadas no passo 1 (pela fonte);
  5. criação da assinatura pela fonte (conjunto de evidências selecionadas com base nos passos 2, 3 e 4);
  6. verificação da assinatura no recetor.

3.3.3 Proteção de conteúdos por *Digital Rights Management* (DRM)

A segurança nas redes P2P também estão relacionadas com a proteção de conteúdos e gestão de direitos digitais DRM. A partilha de informações nesta era digital e a distribuição não autorizada de conteúdos tem crescido bastante. Os editores de conteúdo e fornecedores de *Software* acham cada vez mais difícil proteger seu trabalho e direitos autorais.

Os livros eletrónicos e os arquivos multimédia são distribuídos livremente pela *Internet* e cada vez mais têm sido estudadas novas tecnologias para tornar seguras as informações contra a pirataria ou a cópia ilegal de informações, que são distribuídas livremente através de aplicações P2P, restringindo

o seu acesso por utilizadores não autorizados. Assim, Os fornecedores de *Software* são forçados a adaptar fortes estratégias e tecnologias de prevenção anti-pirataria para prevenir tais riscos.

A gestão de DRM, funciona como um conjunto de tecnologias de controlo de acesso que usam técnicas criptográficas que suportam a proteção de conteúdos digitais, podendo desta forma, ser utilizada pelos detentores de direitos de autor, editores e fabricantes de *Hardware* para limitar a utilização de conteúdos e dispositivos digitais. A *framework* DRM também acomoda o uso *off-line* de conteúdo protegido por uma configuração única [28].

3.3.4 Tipos de proteção por DRM

Existem vários níveis de proteção de conteúdos, nomeadamente a restrição de acesso, proteção básica e gestão de direitos digitais (DRM).

1. **Restrição de acesso**: São providenciadas diversas funcionalidades restritivas de acesso, que podem ser utilizadas para assegurar, que os conteúdos apenas poderão ser visualizados de novo, a partir de uma conta autorizada e assegurar também que o conteúdo apenas poderá ser visualizado em certas zonas geográficas. Estas funcionalidades incluem a aplicação de *tokens* a *Uniform Resource Locator*s (URLs) e GEO-Restrição de conteúdo.

   A ≪tokenização≫ por URL restringe o acesso a partir de uma fonte de dados, através da validação via *token* (ou valor-chave similar) apenso ao URL. A *Content Distribution Network* (CDN) que faz a distribuição, ou sistema similar, é responsável pela validação do *token* gerado pelo *player*, o qual pode representar regras lógicas, baseadas em restrições geográficas.

   Este método é considerado uma aproximação simplista, pois que possui o seu foco no acesso ao vídeo, e não na distribuição de conteúdo.

2. **GEO-Restrição de conteúdos**: As funcionalidades de GEO-restrição, restringem o acesso ao conteúdo com base na localização geográfica do visualizador. A GEO-restrição pode incluir um conjunto de vídeos.

3. **Proteção básica de conteúdos**: A proteção básica de conteúdos, envolve a encriptação da transmissão do vídeo *stream*, ou dos pacotes de vídeo à medida que são transferidos através da rede, e as opções principais são a Encriptação *HTTP Live Streaming* (HLS) e a Verificação *Real-Time Messaging Protocol* (RTMP) para garantir a proteção da informação, utilizando o *Digital Rights Management* (DRM).

De acordo com [28], os fornecedores de *Software*, combinam técnicas de ofuscação de código, junto com soluções de licença de *Software* para proteger seus produtos contra engenharia reversa, adulteração e exploração.

Os protetores de *Software*, utilizam técnicas de codificação e técnicas de encriptação, para ocultar e rastrear o código fonte. Esta estratégia de proteção dos dados por DRM, também apresenta algumas limitações, tais como:

- Ofuscantes de código podem resistir a técnicas de engenharia reversa, em certa medida, mas não pode oferecer uma proteção infalível;
- As técnicas de proteção baseadas em senha não são robustas e muitas vezes apresentam falhas, tais como a partilha de senhas ou a reutilização de números de série;
- As soluções de marca d'água, podem atuar como dissuasores, mas não podem prevenir ativamente o uso indevido do *Software*.

Além das limitações citadas a cima, a *framework* DRM oferece:

- Uma estrutura comum para proteger os objetos digitais e bibliotecas de *software*;
- Extensibilidade da estrutura DRM para proteger vários formatos digitais e suporte para a variedade de clientes em diferentes plataformas;
- Proteção de bibliotecas de *software* de engenharia reversa;
- Execução de *Software* controlado por DRM e a implementação de restrições do fornecedor de *Software*;
- Extensibilidade do design para suportar outros tipos de modelo de licenciamento, como assinatura, aluguer, licenças de teste, *software* de uso único e cópias de avaliação;
- Suporte para modelos *on-line* e *off-line*, para que o tipo apropriado de entrega de conteúdo e métodos de licenciamento possam ser usados [28].

A "vídeo cloud"proporciona um amplo suporte de DRM:

- Suporte de pacotes, licenças e *playback*;
- Uso adaptativo da apropriada DRM para um dado dispositivo;
- Suporte de encriptação *Advanced Encryption Standard* (AES).

- **Considerações úteis na utilização do DRM**

1. se optarmos por terminar um "vídeo cloud DRM"os novos vídeos carregados não serão protegidos por DRM, pelo que o processo terá que começar por desencapotar os títulos referenciados como DRM, um de cada vez;

2. Os vídeos que na nuvem estão protegidos por DRM podem ser partilhados com uma conta não ativada para DRM, mas apenas, após a conta ser ativada para *playback* DRM;

3. o conteúdo DRM, é protegido por Adobe *primetime* e não pode ser visualizado de novo no respetivo módulo de visualização;

4. A encriptação HLS utiliza os mesmos princípios do DRM e da partilha do meio de visualização;

5. o DRM não é suportado nos *players* inteligentes embutidos em *Flash* com utilização de publicação de código *ActionScript*;

6. O DRM não é mantido em vídeos distribuídos no *youtube*.

3.3.5 Encriptação *HTTP Live Streaming* (HLS)

A encriptação HLS, é um protocolo adaptativo criado pela *Apple* para comunicar com os dispositivos MAC e *iPhone Operating System* (iOS). A encriptação HLS, proporciona a salvaguarda da integridade do *Stream*, em segmentos de média e coloca a chave de descodificação diretamente no ficheiro gerado. Foram criados esquemas adicionais para proporcionar melhor trabalho ao nível da invisibilidade da chave. Por exemplo, a codificação rígida da chave na aplicação ou apenas a permissão de um único acesso, na obtenção da chave, a partir da URL. Apesar disso a encriptação HLS não deverá ser considerada segura, porque proporciona um nível mínimo de segurança nos *Streams*.

3.4 Arquiteturas de *Streaming* P2P

Algumas iniciativas de investigação, nomeadamente as promovidas e apoiadas pela União Europeia no Sétimo Programa-Quadro (FP7), endereçam a diferentes níveis as questões da difusão de Multimédia escalável ou adaptativo em tecnologias P2P. Duas dessas iniciativas, os projetos *Socially Aware, collaboRative, scAlable Coding media distributioN* (SARACEN) e **P2P-NEXT** foram percursores no desenvolvimento de propostas para normalização ao nível do IETF dos protocolos para difusão em P2P, nomeadamente o PPSP.

***SEAmless Content Delivery* (SEA)**

Neste projeto foi concebida uma arquitetura (Figura 3.1) que cria um *Overlay* baseada em árvore para *Streaming* de vídeo, otimizando a recuperação de *Peers*, garantindo a entrega de conteúdo multimédia e oferecendo qualidade visual através de exploração de técnicas de codificação avançadas para redes de banda larga [29].

***Network-Aware P2P-TV Application over Wise Networks* (NAPA-WINE)**

Neste projeto (Figura 3.2) o sistema baseia-se na tecnologia P2P adequada para alta qualidade

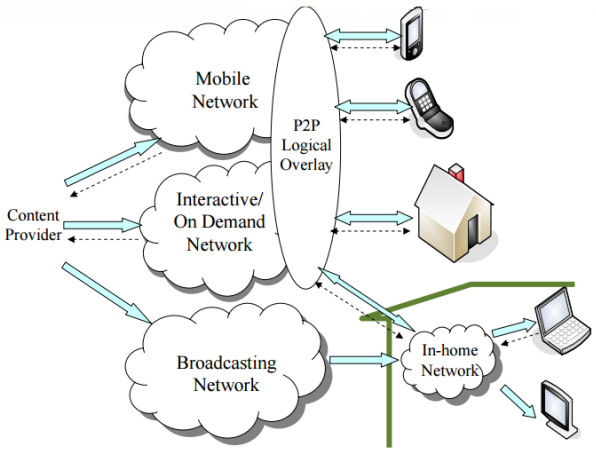

Figura 3.1: Arquitetura SEA

de transmissão de TV ao vivo, utilizando padrões de conceção que associam a comunicação e primitivas para uma nova classe de rede de *Streaming* de conteúdos multimédia *P2P-aware* [30].

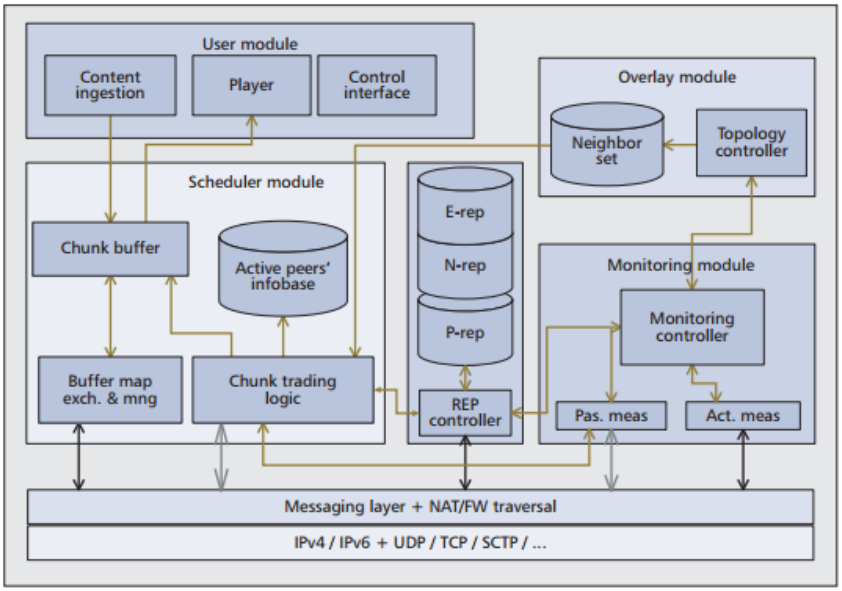

Figura 3.2: Arquitetura NAPA-WINE

***Socially Aware, collaboRative, scAlable Coding media distributioN* (SARACEN)**

Esta rede é considerada descentralizada ou híbrida e consiste em nós *Peer* distribuídos, capazes de transmitir fluxos e arquivos multimédia em fluxo contínuo e localizar conteúdo por meio de um repositório central de meta dados (abordagem baseada em *trackers*) e/ou metadados distribuídos numa abordagem baseada em *Distributed Hash Table* (DHT) [6].

Este projeto implementa uma plataforma protótipo para oferecer *Quality Of Experience* (QoE) personalizado através da integração de técnicas de *Scalable Video Coding* (SVC), protocolos de trans-

porte de multimédia avançados e tecnologias P2P garantindo a privacidade dos utilizadores.
A plataforma utiliza o envolvimento ativo dos utilizadores finais na cadeia de distribuição de média através de sua auto-organização em um ambiente de rede distribuída P2P, para desfrutar os serviços com custos mais baixos.

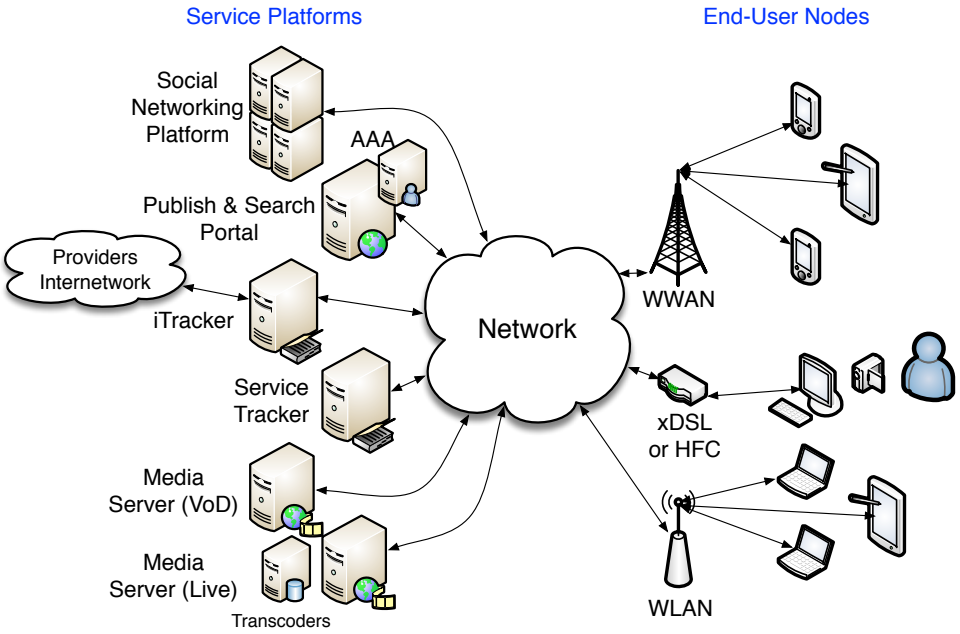

Figura 3.3: Arquitetura SARACEN

P2P-NEXT - *Next Generation Peer-to-Peer Content Delivery Platform*

Neste projeto, permite-se a construção de um sistema *open-source* eficiente e com personalização, adequado para difusão de média com conotação social e colaborativa usando o paradigma P2P.
A arquitetura do sistema é desenvolvida numa perspetiva *user-centric* assegurando que a rede é capaz de suportar o tráfego futuro da Internet para habilitar a partilha de média, através da fusão perfeita de conteúdo, comunidades, comunicação e comércio. A abordagem técnica incluiu a pesquisa e desenvolvimento de modelos de negócios sustentáveis para a cadeia de valor, a pesquisa e especificação de um conjunto de modelos de pagamento, e de um conjunto ferramentas adequados a transformar os serviços não lineares [3] em lineares [4] via P2P [31].

Os aspetos de segurança nestas soluções não são explorados em detalhe dado que os objetivos relacionavam-se essencialmente com o mecanismo de difusão em P2P de conteúdos Multimédia adaptativos e escaláveis em modo linear (tipo TV em direto) ou não-linear (tipo VoD).

[3] *non-Linear Service*: serviço de Televisão a pedido, onde os conteúdos são solicitados (*pulled*) pelos utilizadores
[4] *Linear Service*: serviço tradicional de Televisão, onde os contéudos são difundidos (*pushed*) aos utilizadores

Uma característica comum de todas as soluções investigadas que abordam *streaming* com P2P é a de proporem algoritmos de escalonamento para a distribuição de blocos de dados de média entre vários *Peers* (*Piece-picking* no modo de *Pull*, difusão de origem em modo *Push*, ou híbridos) mas com base no conhecimento das características de codificação dos Média para *Streaming* de modo que as estratégias de propagação seletiva baseadas em prioridades possam ser usados para a transmissão.

4

Mecanismos de Segurança em implementações baseadas em PPSP

Conteúdo

4.1 Protocolos *Standard* de *Streaming* em PPSP 39

4.2 Mecanismos de segurança aplicáveis ao Protocolo PPSP 41

4.3 Demonstração de um mecanismo de segurança em PPSP 47

4.4 Avaliação operacional dos mecanismos 60

A informação constitui nos dias de hoje, um dos ativos mais importantes para as organizações, elevando a segurança da informação para um nível estratégico e até mesmo de sobrevivência, assim torna-se fundamental a utilização dos meios tecnológicos para o armazenamento e a transmissão segura da mesma, de forma a contribuir positivamente para o intercâmbio de informação entre as pessoa.

As tecnologias *Peer-to-Peer* (P2P), surgem como solução para a implementação de *Multicast* na camada de aplicação, resolvendo o problema da escalabilidade apresentadas nas soluções anteriores, mas ainda assim é necessário garantir a questão de controlo nestas rede. Conforme vão surgindo as situações de ataques em rede, torna-se imprescindível a criação de mecanismos de proteção, que consistem na provisão e políticas adotadas para prevenir tais problemas e monitorizar o acesso não autorizado ou a utilização incorreta da informação e dos seus serviços associados.

Os mecanismos de segurança, aplicáveis em implementações baseadas em *Peer-to-Peer Streaming Protocol* (PPSP), devem garantir a segurança não só nos aspetos de integridade dos conteúdos estruturados transmitidos, da sua autenticidade e do respetivo controlo de acesso, mas também nos aspetos de confidencialidade dos conteúdos, da identificação e autenticação dos participantes da sessão e da sua disponibilidade. Assim, torna-se necessário fazer um estudo, para podermos entender de que formas devem ser implementados os mecanismos de segurança aplicados em sistema PPSP, como forma de melhorar a qualidade de transmissão de dados e garantir o acesso seguro das transações feitas nestes tipos de sistemas.

4.1 Protocolos *Standard* de *Streaming* em PPSP

Os sistemas *Peer-to-Peer* (P2P) receberam uma grande atenção nos últimos anos, por oferecerem melhor qualidade de tráfego de *Streaming* na *Internet*, proporcionando alta escalabilidade e tolerância a falhas, capazes de distribuir conteúdos *Streaming* em grande escala, para uma grande audiência. Facto este que tem levado provedores de serviço de *Streaming* a utilizarem o paradigma P2P para distribuir seus *Streaming* de conteúdo, sem ou com associação a *Content Distribution Network*s (CDNs).

Contudo, a maioria dos sistemas P2P para este tipo de finalidade, baseiam-se em protocolos proprietários, ou recorrem a arquiteturas proprietárias e protocolos que, embora executem funções semelhantes, resultam em repetitivos esforços de desenvolvimento e de integração para novos sistemas, aumentando a complexidade de interação com diferentes aplicações de *Streaming* em P2P [4].

Dada a crescente dificuldade de integração dos diferentes sistemas de *streaming* P2P na infraestrutura global da Internet para entrega de conteúdos multimédia, foi sentida a necessidade de normalização deste modelo, e daí o surgimento de um protocolo padrão de *streaming*, que é o *Peer-to-Peer Streaming Protocol* (PPSP), que tem por objetivo resolver os problemas mencionados assim como a padronização das operações no sistema de transmissão P2P. Esta iniciativa de normalização, aberta, de mecanismos

de *Streaming* na *Internet* foi encetada pelo IETF, com a publicação das especificações PPSP.

O PPSP é composto pelo *Peer-to-Peer Streaming Tracker Protocol* (PPSTP), especificado no RFC 7846 [3] e pelo *Peer-to-Peer Streaming Peer Protocol* (PPSPP), especificado no RFC 7574 [5].

O protocolo *Peer*, controla o intercâmbio de dados de média diretamente entre *Peers* e o protocolo *Tracker* permite a troca de informações meta-dados entre os *Peers* e *Tracker* reportando o estado de *Streaming*. A arquitetura requer *Peers* capazes de comunicarem-se com um *Tracker* afim de participarem de um *swarm* que distribua um determinado conteúdo, tal como ilustrado na Figura 4.1.

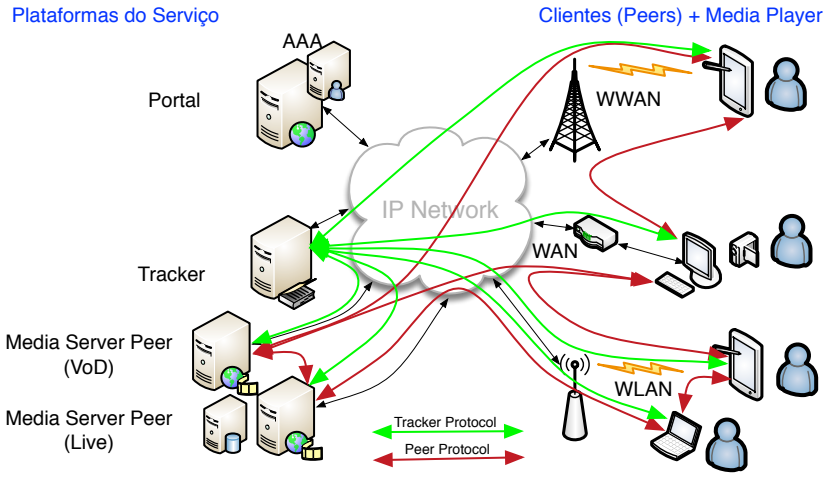

Figura 4.1: Componentes do sistema baseado em PPSP

O Protocolo de *Tracker* é usado para o *Bootstrapping* (na perspetiva de ≪registo≫) e para o controle de um *Peer*, enquanto estiver ativo na distribuição de conteúdo em *Streaming*. Os *Peers* são organizados em *Swarms* que correspondem a grupos de *Peers* partilhando um determinado conteúdo.

Em termos de partilha de dados Multimédia o *Peer* possui duas particularidades:

- O *Peer* ≪semeador≫ (*SEED*): é o *Peer* que contém o conteúdo completo, ou seja, a estrutura dos ficheiros de Multimédia com todos os segmentos (*Chunks*).

- O *Peer* ≪sugador≫ (*LEECH*): é um *Peer* que ainda não concluiu a transferência de todos os segmentos do conteúdo de Multimédia e que contribui com os seus recursos para o mecanismo de *Streaming* no *Swarm* com os segmentos já obtidos.

A arquitetura *PSPP* oferece suporte compatível com a infraestrutura *Web* e foi concebida como uma abordagem em camadas ou seja, uma camada de pedido/resposta, uma camada de mensagem e uma camada de transporte.

4.2 Mecanismos de segurança aplicáveis ao Protocolo PPSP

A segurança da informação, é motivo de preocupação para as organizações, visto que a informação tornou-se um ativo de grande valia para a continuidade dos negócios, o que fez com que alguns organismo, nomeadamente a a *International Organization for Standardization* (ISO) editasse normas técnicas, que se complementam nas definições de diretrizes e recomendações de boas práticas para a segurança da informação [32].

Quando se fala da segurança em rede P2P, pode-se olhar em diversos aspetos ou caraterísticas que devem ser levadas em consideração para podermos então considerar um sistema seguro, e que vão da autenticação dos utilizadores/participantes no processo, à integridade e disponibilidade dos conteúdos, assim como de aspetos de "legalidade" na disponibilização ou no consumo, e de direitos de autor sobre esses conteúdos.

Para que a transmissão de conteúdos seja considerada segura utilizando os protocolos PPSP, é necessário a implementação de mecanismos de segurança em soluções baseadas nesses protocolos, capazes de garantir que tanto os conteúdos como os participantes da sessão estejam não apenas protegidos, mas também operando em condições funcionais seguras. Neste contexto, os mecanismos de segurança que podem ser aplicáveis a implementações baseadas em PPSP, serão os descritos nas secções seguintes.

4.2.1 Identificação e Autenticação dos *Peers*

A autenticação é um dos principais mecanismos de segurança, dentro de um sistema, que tem como função verificar se alguém é de fato quem afirma ser, em uma transação ou em uma conexão mantida entre entidades participantes no sistema. Primeiro, no estabelecimento de uma conexão, o serviço deve assegurar que ambas as entidades são autênticas. Segundo, o serviço deve assegurar que a conexão não sofre interferência de forma que uma terceira entidade possa se fazer passar por uma das partes, com o propósito de envio ou recebimento não autorizado de mensagens quer e sinalização/controle quer de conteúdos [33]

Para garantir que a informação não sofra interferências, o sistema deve permitir a validação das credencias de um utilizador ou participante, de forma a verificar se o mesmo é quem diz ser. Uma vez identificado e validado, a autorização determina o que utilizador ou participante ≪pode fazer≫, isto é, quais os recursos do sistema que pode utilizar ou a que tem acesso.

O PPSP deve permitir *Swarms* de tipo fechado, onde somente os *Peers* considerados como autênticos e autorizados terão acesso, garantindo assim que somente os *Peers* confiáveis possam acessar os conteúdos originais no sistema. Isso pode ser alcançado por um mecanismo de segurança, tal como a autenticação dos *Peers* que atuam como "agentes" na operação desencadeada pelo respetivo "utiliza-

dor" do serviço, ou por um esquema de gestão de chaves. Por falta de controlo de acesso ao sistema, pode ocorrer o ataque de tipo *Sybil*[1], gerando a falsificação de múltiplas identidades no sistema.

Para se evitar a existência deste tipo de ataque, deve ser necessário, utilizar uma entidade fraca para fazer *Hash* de um identificador, tal com o seu endereço IP, ao gerar o ID [11]. Este mecanismo é responsável por determinar aleatoriamente um identificador para cada *Peer* que deseja participar no *swarm*, isto porque um *Peer* não deve poder alocar o seu próprio ID, evitando assim o ataque anteriormente descrito.

No contexto de uma implementação com base em PPSP e dado que as entidades lógicas que efetuam a gestão dos *Peer*—o Portal com funções *Authentication, Authorization, Accounting* (AAA) e o *Tracker*—são entidades de serviços baseados na *Web*, a criação do *Hash* já é parte do protocolo *Hyper Text Transfer Protocol Secure* (HTTPs) configurado nessas entidades lógicas e que desempenha a função de uma Agência Certificadora, em que se confie na atribuição de identidades.

4.2.2 Controlo de acesso

O controle de acesso, está ligado ao conceito de autorização, onde são definidas regras para limitar os participantes no acesso ao sistema. Boa parte das aplicações existentes concentra-se na partilha de recursos de forma aberta, ao invés de implementar regras que permitem definir quem, quando e o que pode ser acedido [2].

A utilização mecanismo de autorização restringe o acesso aos recursos com base em informações sobre o requisitante. Para que se possa garantir o mecanismo de controlo de acesso, utilizando o protocolo PPSP, cada *Peer* deve possuir um identificador global enquanto participante ativo, o qual está relacionado com a definição de um *Peer ID*, que é um requisito básico para que um *Peer* seja identificado de forma exclusiva num sistema de transmissão de P2P para que outros *Peer* ou o *Tracker* possam se referir a esse participante no sistema utilizando apenas o respetivo *ID*, sem necessidade de envolvimento de informação do "utilizador/Cliente" do serviço.

Outra forma de se garantir a autorização segura no controlo de acesso é na utilização do método conhecido como *OAuth* [34], geralmente, o *OAuth* fornece aos clientes um acesso seguro aos recursos do servidor em nome do proprietário do recurso. Este especifica um processo para o proprietários do recursos, para autorizar o acesso de terceiros aos seus recursos de servidor sem compartilhar suas credenciais.

Este método é utilizado especificamente para trabalhar com o protocolo de *Hypertext Transfer Protocol* (HTTP), emitindo *Tokens* de acesso para os utilizadores, mediante autorização do servidor, com a aprovação do proprietário do recurso.

[1] Ataques Sybil: um único nó pode utilizar várias identidades diferentes em conluio para confundir a rede. Os mecanismos de reputação dificilmente lidam com este tipo de ataque, uma vez que os votos das identidades em conluio facilmente deturpam os mecanismos de votação

O *Token*, é uma chave associada a um utilizador e são utilizados durante um certo período de tempo, ao fim do qual o *Token* perde validade e o utilizador recebe do servidor outra chave para a continuidade das suas operações.

4.2.3 Integridade dos conteúdos

Para que um conteúdo seja integro, precisa estar protegido contra qualquer tipo de alteração não autorizada. Tal como é garantida a confidencialidade, a integridade é aplicada a um fluxo de mensagens, a uma mensagem individual ou a alguns campos da mensagem. Este mecanismo, pode ser orientado à conexão, em que pode ser aplicado a um fluxo de mensagens, garantindo que as mensagens são recebidas da mesma forma como são enviadas pela fonte, sem duplicação, inserção ou modificação.

Assegurar a integridade em aplicações P2P, é indispensável para identificação de objetos corrompidos. A adulteração de objetos pode ocorrer tanto no momento do seu armazenamento, como no transito entre nós. Assim podem implementar-se diversos mecanismos baseados em algoritmos criptográficos por forma a se assegurar a integridade das aplicações [35]. Os serviços de integridade podem ser diferenciados também pela existência ou não de recuperação de dados, isto porque, quando uma violação de integridade é detetada, este mesmo serviço pode reportar a violação da mensagem.

A integridade dos conteúdos de *Streaming* no PPSP [4,5], deve ser apoiado para fornecer aos *Peers* a possibilidade de identificar conteúdo não autêntico, assim como modificações indesejáveis por outras entidades utilizando a sua fonte genuína.

Segundo [11], o mecanismo que assegura a integridade de um objeto, pressupõe a verificação pelo nó que o recuperou. Para tal, o nó que deseja inserir o objeto na rede calcula o *Hash* do seu conteúdo para produzir a chave. Quando um nó recupera um objeto, calcula o *Hash* do objeto e compara-o com a chave utilizada na procura. Caso sejam iguais, o objeto pode ser considerado integro.

As especificações do protocolo PPSP resolve essas ameaças utilizando o métodos de mitigação ao combinar ambos os protocolos em um aplicativo. Isso incluirá privacidade do utilizador e proteção da distribuição de conteúdo utilizando o mecanismo de criptografia, para a proteção de dados prevê a verificação de integridade dos dados transportados, através da verificação da sua assinatura *Hash*, comparando-a com o resultado do processo antes da transmissão.

A violação dos mecanismos baseados em criptografia é difícil de ser alcançado computacionalmente pelos atacantes, os ataques à integridade mais típicos quando não existe um mecanismo de segurança são os seguintes [2]:

- Ataque de poluição a arquivos: neste tipo de ataque, um nó malicioso anuncia uma cópia corrompida de um arquivo e a distribui quando solicitado por outro nó.
- Ataque de resposta falsa: este ataque sucede quando, um nó malicioso encaminha normalmente

mensagens de busca, tendo a possibilidade de identificar e intercetar uma mensagens de resposta, modifica a resposta e o nó malicioso fornece uma cópia corrompida do arquivo.

4.2.4 Confidencialidade dos conteúdos

A confidencialidade tem como função de manter o sigilo ou a privacidade de uma informação, como também pode ser resumida como a ausência de divulgação não autorizada de uma informação, evitando que pessoas ou entidades não autorizadas, tenham acesso à mesma e garantindo que somente as partes envolvidas na comunicação terão acesso à mesma. [32].

Assim, quando se garante este requisito, nenhum utilizador pode partilhar qualquer conteúdo, utilizando a identidade de outro utilizador. Isto também está relacionado com o processo de integridade, que diz respeito à ausência de alterações não autorizadas ao sistema ou à informação em causa, garantindo a proteção dos dados transmitidos.

A confidencialidade pode ser aplicada em vários níveis, começando pela simples transmissão de dados entre dois intervenientes numa comunicação, ou protegendo apenas campos específicos de uma mensagem, evitando desta forma a análise de tráfego por terceiros, através da identificação de origem ou destino, frequência ou por outras características que possa impedir a falta de confidencialidade nos dados ou serviços fornecidos pelo sistema em causa.

A garantia da confidencialidade dos dados num sistema está ligado com a utilização de algoritmo de criptografia, baseando-se no uso de cifras. Uma cifra é uma técnica concreta de criptografia, que permite ocultar uma informação. A operação de uma cifra, é definida através da especificação de um algoritmo e de uma chave. O algoritmo define o modelo genérico de transformação de dados, a chave é um parâmetro do algoritmo que permite variar o seu comportamento de forma complexa [28]. No caso do PPSP, um editor de conteúdo que deseje a confidencialidade deve distribuir o mesmo num formato ao qual as técnicas de cifra de tipo DRM tenham sido aplicadas [4]

4.2.5 Disponibilidade

A disponibilidade é a garantia de que os utilizadores autorizados obtenham acesso à informação e aos ativos correspondentes sempre que for necessário, permitindo que os *Peers* estejam acessíveis para a realização do processo esperado [32].

Nas redes P2P, a disponibilidade está relacionada com o *TimeSlot* [2], a técnica básica para incremento da disponibilidade é a replicação, impedindo a existência de pontos únicos de responsabilidade num único nó.

[2]TimeSlot: No contexto das redes P2P este termo, refere-se a parcela de tempo em que um objeto ou serviço está associado aos demais nós [2].

Várias formas de ataques podem estar relacionados com a falta de disponibilidade de um sistema, sendo que os principais ataques baseiam-se com a negação de serviço (DoS). Regista-se um caso de ataques DoS quando por exemplo um nó malicioso funciona corretamente com o objetivo de fazer buscas, mas quando é solicitado a partilhar, nega o serviço, mantendo o objeto ou serviço sob sua responsabilidade, recusando-se no envio de respostas.

A sobrecarga de um nó, é também outra forma de ataque DoS, em que nós corretos tornam-se involuntariamente incomunicáveis, o que culmina na sua eliminação do *overlay* por outros nós. Também se pode verificar outra forma de ataque, como a entrada e a saída acelerada de nós no *overlay*, o designado *Churn*, o que causa a sobrecarga de determinados nós ou de recursos de rede e provocar a negação de serviços aos demais nós. O protocolo de PPSP do *Tracker* deve levar em consideração a frequência de troca de mensagens e a utilização de largura de banda eficiente ao comunicar informações de disponibilidade do *Peers*.

4.2.6 Anonimato

Este mecanismo, tem como função de garantir que as identidades reais no sistema (tipicamente os "utilizadores/Clientes") permanecerão desconhecidas, evitando que um nó possa ser identificado numa comunicação com outro nó ou um servidor qualquer.

Nas redes P2P, o anonimato deve garantir a não-identificação dos seguintes elementos:

- O autor ou o responsável pela publicação de determinado objeto ou serviço;
- A identidade do nó que está armazenar um objeto ou serviço;
- A identidade e do conteúdo do objeto;

Como cada nó na cadeia conhece apenas os seus vizinhos imediatos, não há como determinar a identidade dos nós que publicam o conteúdo original, de todos os nós que estão armazenando todos os objetos e dos nós de onde partem os pedidos para obtenção dos objetos [2].

Os principais ataques em que as soluções de anonimato estão vulneráveis são os passivos, em especial os de escuta e análise de tráfego. O objetivo desses ataques é desvendar a identidade dos elementos envolvidos nas comunicações estabelecidas.

No caso particular das aplicações P2P, há também interesse em revelar quem publicou uma informação, quem está armazenando e quem solicitou determinados objetos. Em rede P2P, a negabilidade pode ser entendida como um componente do aspeto do anonimato, referindo-se à habilidade de um nó em negar conhecer o conteúdo de objetos por ele armazenados [17].

4.2.7 Auditoria

Uma das formas de se garantir o correto funcionamento dos mecanismos de segurança, definidos num sistema de acordo com [35], é a utilização de auditoria permitindo a análise do funcionamento dos mecanismos de segurança durante ou após a sua execução. Este método tem como objetivo gerar *logs* de execução.

A análise dos *logs*, deve permitir a verificação do correto funcionamento dos mecanismos de segurança, acompanhando a sua execução, quanto possibilitar a identificação das causas, responsáveis pela falha de algum mecanismos de segurança. Dependendo da situação, esta análise pode ser feita tanto de forma manual quanto automática, sendo importante haver uma estrutura coerente na geração dos *logs*.

4.2.8 Outras formas de segurança Aplicáveis ao Protocolo PPSP

Além dos mecanismos de segurança identificados, para a construção de um sistema *streaming* seguro, utilizando o protocolo *Peer-to-Peer Streaming Protocol* (PPSP), que é garantido utilizando o mecanismo à cima referenciados, outras características são consideradas preponderantes, e estão associados aos mecanismos de segurança dos dados, como: o *Routing* seguro e a Reputação dos *Peer*.

- ***Routing* Seguro:** Os ataques coordenados aos protocolos de descobertas de recursos em *Overlays* não estruturados, afetam a disponibilidade e a autenticidade, isto a nível do envenenamento da *Cache* de *Pong* (*Pong-cache Poisoning*). A *Cache Pong* é a lista de nós conhecidos pelo nó, esta lista é usado para a descoberta de novos nós no *Overlay*, e o seu envenenamento é provocado pela contaminação com identificadores de nós adversários.

  O *Routing* seguro para *Overlay*, é descrito como uma primitiva que assegura que quando um nó operacional envia uma mensagem para uma chave k, é provável que essa mensagem chegue a todos os nós operacionais no conjunto de nós que mantêm replicas [36]. Isso depende de 3 mecanismos:

  1. Atribuição segura de ID de nós;
  2. Manutenção segura da tabela de *Routing*;
  3. Encaminhamento seguro de mensagens.

- **Sistema de Gestão de Reputação:** Para podermos entender, o funcionamento de um sistema de reputação, é importante referenciar que, a gestão de qualquer sistema, seja ele etnológico ou não é um conjunto de atividades e rotinas voltadas para o planeamento, monitoramento e controle dos serviços prestados pela infra-estrutura ou por uma determinada organização.

  O mecanismo de reputação pode substituir um sistema baseado em certificados digitais, o que permite de certa forma garantir o mecanismo de anonimato. Esta abordagem permite a atribuição

de pesos aos nós que mais interagem no *overlay* no sentido de incentivá-los a continuarem a colaborar com os outros *peers* na rede. Ataques à reputação visam a comprometer os mecanismos associados à identificação e à contabilização de nodos maliciosos [2].

Um sistema de gestão de reputação em rede P2P, visa balancear o melhor funcionamento da rede de modo geral, agindo como um mecanismo de incentivo para os *Peers*, fornecendo uma determinada recompensa pelo bom comportamento dos participantes. Para que isso aconteça o sistema de gestão de reputação supervisiona as atividades realizadas pelos *Peers*. Os *Peers* que mais participam, ganham o privilégio de terem a melhor reputação e os *Peers* menos participativos, são eliminados da rede ou de um grupo de interesse.

O sistema de gestão de reputação ou incentivos, pode ser integrado ao nível do *Tracker*, ou interagindo com este, dado que no *Tracker* as mensagens `STAT_REPORT` podem incluir parâmetros de recolha de informação [3] com base no histórico de transações passadas, tais como, contagem de segmentos (enviados e recebidos) e tempo de contribuição do *peer* no *swarm*, taxa de transmissão alcançada, etc.

4.3 Demonstração de um mecanismo de segurança em PPSP

Afim de se poder efetuar uma demonstração da incorporação de mecanismos de segurança em sistemas baseados em PPSP, foi necessário recorrer a um sistema de *Streaming* em P2P existente. Para tal, foi utilizado parte do código de referência para *Tracker* e para *Peer* (apenas na componente de interação com o sistema, ou seja, Portal e *Tracker*) desenvolvido no âmbito do projeto *SARACEN*, que foi a base para o desenvolvimento do PPSTP, e que assim permite incorporar e testar os mecanismos de segurança aplicáveis.

Dada a complexidade que os aspetos de segurança apresentam, é impraticável, ou mesmo irrealista, a implementação de todos os mecanismos de segurança anteriormente descritos, devendo contudo atender-se ao contexto, às condicionantes e ao nível de tolerância funcional do ambiente ou sistema que se pretende seguro. Um dos aspetos importantes nestes sistemas, é o de identificação, autenticação, e autorização dos *Peers* no sistema, supondo para tal um cenário de distribuição de conteúdos em ambiente controlado, ou seja, num cenário onde os clientes do serviço têm que ser conhecidos, afim de poderem aceder ao mesmo. Tomamos assim este aspeto como base para o protótipo demonstrativo de alguns dos mecanismos preconizados.

4.3.1 Implementação funcional

Para a demonstração utilizou-se um emulador de rede em tempo-real que instancia elementos de rede e nós de computação nos quais são executadas as aplicações do sistema (Portal, *Tracker*, *Peers*). Para o desenvolvimento e teste, o emulador de rede é executado em máquina virtual com sistema operativo Linux, pelo que os nós de computação da rede correspondem a *Containers* Linux que executam as aplicações nativas do sistema PPSP. As ferramentas usadas foram as seguintes:

1. **VirtualBox** é um *Hypervisor* de virtualização, que visa criar ambientes para instalação de sistemas distintos, assim a execução de aplicações (*Softwares*) para esses sistemas, compartilhando fisicamente o mesmo *Hardware*.

2. **Vagrant** é um mecanismo de lançamento automatizado de ambientes virtuais, que permite de forma programática (*scripting*) instanciar e configurar rapidamente sistemas que são reprodutíveis em ambiente real.

3. **Emulador de Rede virtual *C.O.R.E.*** o *Common Open Research Emulator* (CORE) é um emulador de redes robusto, desenvolvido em Python pela *Boeing Research and Technology* conjuntamente com o *Naval Research Laboratory* dos Estados Unidaos da América[3]. o CORE permite a emulação de uma rede em tempo real, onde modelos abstratos são utilizados. A emulação pode ser conectada a redes físicas reais, fornecendo um ambiente para executar aplicações e protocolos reais, aproveitando a virtualização fornecida pelos sistemas operativos do Linux. A escolha desta ferramenta deveu-se ao facto de proporcionar uma demonstração fidedigna para testes de aplicativos e plataformas, avaliando cenários de rede, estudos de segurança e aumento do tamanho das redes de testes físicos (em escala).

Nesta emulação do sistema (ilustrado na Figura 4.2 foi implementado um Portal, que corresponde à entidade que efetuará a gestão dos utilizadores desse sistema, que permitirá aos mesmos a seleção dos conteúdos pretendidos, e que também executa a função lógica de *Tracker* do sistema, que atribuirá chaves seguras (*Peer ID*) aos *Peers* associados à aplicação Cliente de *Streaming* em P2P no *swarm*. Assim, nesse Portal rodam os seguintes serviços:

1. Autenticação para validação do *cliente* e para que o mesmo seja considerado como uma entidade do sistema.

2. Atribuição de identificador de cliente único, o *Peer-ID*, que lhe dará acesso aos recursos do serviço.

3. O processo servidor que implementa o *Tracker* do sistema, é configurado para funcionar em modo HTTP seguro. O *Tracker*, tem também por missão fornecer a lista dos *Peer* ativos em cada *Swarm*

[3] https://www.nrl.navy.mil/itd/ncs/products/core

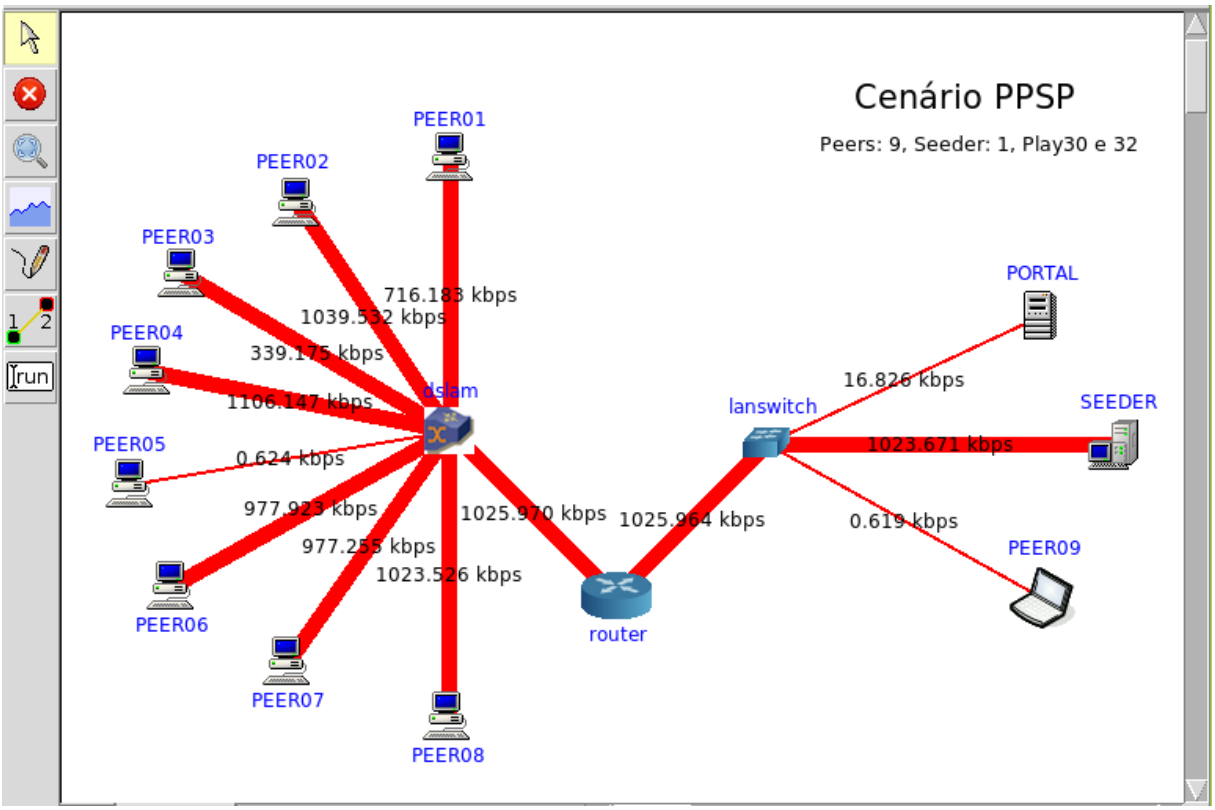

Figura 4.2: Cenário do sistema PPSP Implementado (em execução). O volume de tráfego trocado entre os nós da rede é representado por linhas vermelhas de maior espessura.

(ou seja, cada conteúdo multimédia) e outras funções desempenhados por ele no processo de *Streaming*. O protocolo de *Tracker*, o PPSTP foi implementado em HTTP, mas de forma a prover segurança nas transações efetuadas com os *Peers*, garantindo que os dados sejam transmitidos por meio de uma conexão criptografada. Esta segurança é conseguida por meio do protocolo de segurança da camada de transporte *Transport Layer Security* (TLS), que fornece proteção dos dados em três aspetos principais:

(a) **Criptografia**: os dados trocados entre *Peers* e *Tracker*/Portal estão a salvo de ataques relativamente ao conteúdo das mensagens trocadas.

(b) **Integridade**: os dados não podem ser modificados ou corrompidos durante a sua transferência.

(c) **Autenticação**: permite provar que os *peers*, comuniquem-se com o *Tracker*/Portal, com proteção aumentando a confiança.

4. Manutenção e disponibilização dos meta-dados correspondentes aos conteúdos (*Media Descriptor*) que são ficheiros em formato *Extensible Markup Language* (XML), e que são usados tanto pelos *Peers* como pelo *Tracker*.

Também foi considerado um *Peer* especial do tipo *SEED* que desempenha o papel de *Media Server*, e que funcionará como fonte garantida inicial dos *Streams*.

Na configuração do Portal apenas se implementou uma lista de *Clientes* em acesso seguro que permite a autenticação e validação posterior pelo *Tracker*, conforme verificável na Listagem 4.1.

Listagem 4.1: Script de configuração do Portal

```
# apache2.conf generated by utility.py:HttpService
PidFile ${APACHE_PID_FILE}
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}
AccessFileName .htaccess

<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy all
</Files>

DefaultType None
HostnameLookups Off
ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
Include /usr/lib/apache2/mods/mods-enabled/*.load
Include /usr/lib/apache2/mods/mods-enabled/*.conf

<IfModule mod_ssl.c>
    Listen 443
</IfModule>
<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

```
30
31 LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
      Agent}i\"" vhost_combined
32 LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i
      \"" combined
33 LogFormat "%h %l %u %t \"%r\" %>s %O" common
34 LogFormat "%{Referer}i -> %U" referer
35 LogFormat "%{User-agent}i" agent
36 ServerTokens OS
37 ServerSignature On
38 TraceEnable Off
39
40 <IfModule mod_ssl.c>
41     <VirtualHost *:443>
42         DocumentRoot /var/www
43         ErrorLog ${APACHE_LOG_DIR}/error.log
44         LogLevel warn
45         CustomLog ${APACHE_LOG_DIR}/access.log combined
46         SSLEngine on
47         SSLCertificateFile /usr/lib/apache2/ssl/server.crt
48         SSLCertificateKeyFile /usr/lib/apache2/ssl/server.key
49         <Directory />
50             Options FollowSymLinks
51             AllowOverride None
52             Require all denied
53         </Directory>
54         <Directory /usr/share>
55             AllowOverride None
56             Require all granted
57         </Directory>
58         #Autorizacao - usar o recurso
59         <Directory /var/www/>
60             Options Indexes FollowSymLinks MultiViews
61             # autenticacao
62             AuthType Basic
```

```
63                AuthName "Auth Area"
64                AuthUserFile /usr/lib/apache2/users.txt
65                Require valid-user
66            </Directory>
67        </VirtualHost>
68    </IfModule>
```

Ao nível do emulador de rede foram definidos “serviços” para cada elemento do sistema PPSP que criados em modo paramétricos, ou seja, são definidos com “variáveis” que permitem gerar “shell scripts” que são executados no arranque (*bootstrap*) de cada nó virtual e de cada componente da rede, e referem-se por exemplo a características das ligações entre nós e elementos da rede, protocolos e configurações dos elementos de rede e dos nós ligados à rede (endereços IP, identificadores, etc.), e aplicações invocadas nos nós virtuais. Na essência, cada nó da rede executa no arranque um *script* BASH com os comandos necessários e à respetiva identificação única.

A implementação, efetuada com recurso a “scripting” por questões de simplicidade, pode perfeitamente ser codificada em módulos componentes dos elementos lógicos do sistema (*Peer* e *Tracker*), na linguagem de programação que seja adotada para a sua concretização.

Como exemplo, ilustra-se na Listagem 4.2 o protótipo de *script* `startseeder.sh` de arranque de um *Peer* de tipo SEED.

Listagem 4.2: Definição do “serviço” de um *Peer* de tipo SEED

```
1  #
2  # CORE
3  # Copyright (c)2010-2012 the Boeing Company.
4  # See the LICENSE file included in this distribution.
5  #
6  ''' PPSP PeerSeeder user-defined service.
7  '''
8
9  import os
10
11 from core.service import CoreService, addservice
12 from core.misc.ipaddr import IPv4Prefix, IPv6Prefix
13
14 class PeerSeeder(CoreService):
15     ''' This service starts up a PPSP Peer in Seeder Mode sharing contents in
```

```
        linked ~/cache/.
    '''
    # a unique name is required, without spaces
    _name = "PeerSeeder"
    # you can create your own group here
    _group = "PPSP"
    # list of other services this service depends on
    _depends = ()
    # per-node directories
    _dirs = ()
    # generated files (without a full path this file goes in the node's dir,
    #  e.g. /tmp/pycore.12345/n1.conf/)
    _configs = ('startseeder.sh', )
    # this controls the starting order vs other enabled services
    _startindex = 55
    # list of startup commands, also may be generated during startup
    _startup = ('sh startseeder.sh',)
    # list of shutdown commands
    _shutdown = ()

    @classmethod
    def generateconfig(cls, node, filename, services):
        ''' Return a string that will be written to filename, or sent to the
            GUI for user customization.
        '''
        cfg = "#!/bin/sh\n"
        cfg += "# auto-generated by PeerSeeder (peerseeder.py)\n"
        for ifc in node.netifs():
            cfg += 'echo "Node %s has interface %s"\n' % (node.name, ifc.name
                )
            # here we do something interesting
            cfg += "\n".join(map(cls.subnetentry, ifc.addrlist))
            break
        cfg += "\n"
        # Link to cache the media sources
        cfg += "ln -s /opt/media cache\n"
        # create link to peer apps
        cfg += "ln -s /usr/local/libppsp/bin/peer_agent peer_agent\n"
```

```
52          cfg += "ln -s /usr/local/libppsp/bin/peer_engine_client_retry
                peer_engine_client_retry\n"
53          # install peer-agent-plugins
54          cfg += "mkdir -p peer-agent-plugins\n"
55          cfg += "ln -s /usr/local/libppsp/lib/libppsp-0/libbandwidth.so ./peer
                -agent-plugins/libbandwidth.so \n"
56          cfg += "ln -s /usr/local/libppsp/lib/libppsp-0/libconsumer_interface.
                so ./peer-agent-plugins/libconsumer_interface.so \n"
57          cfg += "ln -s /usr/local/libppsp/lib/libppsp-0/
                libmanifest_v2_stream_descriptor.so ./peer-agent-plugins/
                libmanifest_v2_stream_descriptor.so \n"
58          cfg += "ln -s /usr/local/libppsp/lib/libppsp-0/
                libpart_chunkmap_manager.so ./peer-agent-plugins/
                libpart_chunkmap_manager.so \n"
59          cfg += "ln -s /usr/local/libppsp/lib/libppsp-0/libppsp_tp_client_inov
                .so ./peer-agent-plugins/libppsp_tp_client_inov.so \n"
60          # Start Peer after a connecyivity test with the main Tracker to
                collect link data
61          cfg += "ping -n -D -c 50 -W 70 198.18.11.10 >/opt/ppsp/ping%s.log\n"
                % (node.name)
62          cfg += "./peer_agent -f -p 8080 -d -P %s.pid %s >/opt/ppsp/%s.log
                2>&1 &\n" % (node.name, node.name, node.name)
63          # Wait a bit for peer startup
64          cfg += "sleep 5\n"
65          # Now the Client can invoke the JOIN to the Seeder SWARMS
66          cfg += "./peer_engine_client_retry load localhost 8080 3 5 ./cache/
                streams/manifests/*.xml >/opt/ppsp/client%s.log 2>&1 \n" % (node.
                name)
67          return cfg
68
69      @staticmethod
70      def subnetentry(x):
71          ''' Generate a subnet declaration block given an IPv4 prefix string
72              for inclusion in the config file.
73          '''
74          if x.find(":") >= 0:
75              # this is an IPv6 address
76              return ""
```

```
77          else:
78              net = IPv4Prefix(x)
79              return 'echo "  network %s"' % (net)
80
81  # this line is required to add the above class to the list of available
        services
82  addservice(PeerSeeder)
```

O processo de identificação do Peer e verificação entre o *Tracker* e o Portal é mostrado na linha 15 da Listagem 4.3, que corresponde ao "script" de arranque do *Tracker*/Portal. Se o identificador do processo cliente não for considerado como válido é removido, tal como apresentado na linha 18, o que pode significar ser um identificador suspeito, por apresentar um identificador de origem duvidosa.

O *Peer-ID* é um valor obtido a partir de processo descrito na linha número 28 desta mesma listagem, e é baseado em função do endereço IP atribuído ao *Peer* e do momento temporal de entrada no *Swarm*, sendo portanto um identificador praticamente único.

O identificador mantém-se válido até ao *Time Out* do *Peer*, ou seja, quando o mesmo sair do *swarm*. Isto vai permitir garantir também a integridade da fonte de *Streaming* e consequentemente garantir a integridade do respetivo conteúdo.

Listagem 4.3: Algoritmos de Aquisição do Peer-ID

```
1  echo "Node PORTAL has interface eth0"
2  echo "  network 198.18.11.0/24"
3  ln -s /usr/local/libppsp/bin/tracker2 tracker2
4  cp /opt/media/configs/asn.csv .
5  mkdir -p tracker2-plugins
6  ln -s /usr/local/libppsp/lib/libppsp-0/libbandwidth.so ./tracker2-plugins/
       libbandwidth.so
7  ln -s /usr/local/libppsp/lib/libppsp-0/libppsp_tp_server_inov.so ./tracker2-
       plugins/libppsp_tp_server_inov.so
8  ln -s /usr/local/libppsp/lib/libppsp-0/libwebmonitor_feeder.so ./tracker2-
       plugins/libwebmonitor_feeder.so
9  ln -s /usr/local/libppsp/lib/libppsp-0/liblocation_emulator.so ./tracker2-
       plugins/liblocation_emulator.so
10 ping -n -D -c 50 -W 70 198.18.11.1 >/opt/ppsp/pingPORTAL.log
11 ./tracker2 -d -p 8070 -f -P PORTAL.pid >/opt/ppsp/PORTAL.log 2>&1 &
12
13 cp /opt/media/streams/manifests/* /var/www/
```

```
14
15 if  ! test -d var.www/hosts; then
16   mkdir var.www/hosts
17 else
18   rm -rf var.www.hosts/*
19 fi
20
21 sleep 1m
22
23 hostname_list="$(cat var.log.apache2/access.log|cut -d' ' -f3| grep PEER|sort
       -u | xargs)"
24 export hostname_list
25
26 for host in $hostname_list;
27 do
28   h="$(echo `date +%s`$hst|sha256sum|base64|head -c32)"
29   echo $h > var.www/hosts/"$host".txt
30 done
```

Na Listagem 4.4 ilustra-se o a "script" de arranque de um *Peer* (cliente PEER03-22) que pretende efetuar *stream* de um conteúdo cujo *Media Presentation Description* (MPD) está descrito no ficheiro `32_sea_2s_d80s_svc_p2p_emu.xml`.

Listagem 4.4: Script de arranque de Peer (para "SWARM-ID"específico)

```
1
2 #!/bin/sh
3 # auto-generated by PeerPlay32 (peerplay32.py)
4 echo "Node PEER03-22 has interface eth0"
5 echo "  network 198.18.13.0/24"
6 mkdir -p cache
7 ln -s /usr/local/libppsp/bin/peer_agent peer_agent
8 ln -s /usr/local/libppsp/bin/peer_engine_client_retry
      peer_engine_client_retry
9 ln -s /usr/local/libppsp/bin/player_emulator player_emulator
10 cd .
11 curl -k -u PEER03-22:123456 https://198.18.11.10/32
```

```
    _sea_2s_d80s_svc_p2p_emu.xml
12
13 mkdir -p peer-agent-plugins
14 ln -s /usr/local/libppsp/lib/libppsp-0/libbandwidth.so ./peer-agent-
    plugins/libbandwidth.so
15 ln -s /usr/local/libppsp/lib/libppsp-0/libconsumer_interface.so ./
    peer-agent-plugins/libconsumer_interface.so
16 ln -s /usr/local/libppsp/lib/libppsp-0/
    libmanifest_v2_stream_descriptor.so ./peer-agent-plugins/
    libmanifest_v2_stream_descriptor.so
17 ln -s /usr/local/libppsp/lib/libppsp-0/libpart_chunkmap_manager.so ./
    peer-agent-plugins/libpart_chunkmap_manager.so
18 ln -s /usr/local/libppsp/lib/libppsp-0/libppsp_tp_client_inov.so ./
    peer-agent-plugins/libppsp_tp_client_inov.so
19
20 sleep 1m
21 curl -k -u PEER03-22:123456 https://198.18.11.10/hosts/PEER03-22.txt
22
23 PEERNAME=$(cat PEER03-22.txt)
24
25 ping -n -D -c 5 -W 70 198.18.11.10 >/opt/ppsp/pingtestPEER03-22.log
26 ./peer_agent -f -p 8080 -d -P PEER03-22.pid $PEERNAME >/opt/ppsp/
    PEER03-22.log 2>&1 &
27
28
29 ./peer_engine_client_retry load localhost 8080 3 5 32
    _sea_2s_d80s_svc_p2p_emu.xml >/opt/ppsp/clientPEER03-22.log 2>&1
30 ./player_emulator -d -p 0 -c 3 -P localhost 8080 32
    _sea_2s_d80s_svc_p2p_emu.xml 2>/opt/ppsp/play32PEER03-22.log 1>/
    opt/ppsp/play32PEER03-22.ts
```

Tal como é ilustrado na (Figura 4.3), que temos a arquitetura de *Streaming*, com vários grupos de *Streaming*, isto é os *Peer* juntam-se a um determinado *Swarm* para partilhar conteúdo da mesma natureza, diferente do que está sendo partilhado pelos outros grupos.

Na Listagem 4.5 ilustra-se o MPD de um conteúdo com `SWARM-ID` “UNDERTHESEASVC2SEC”.

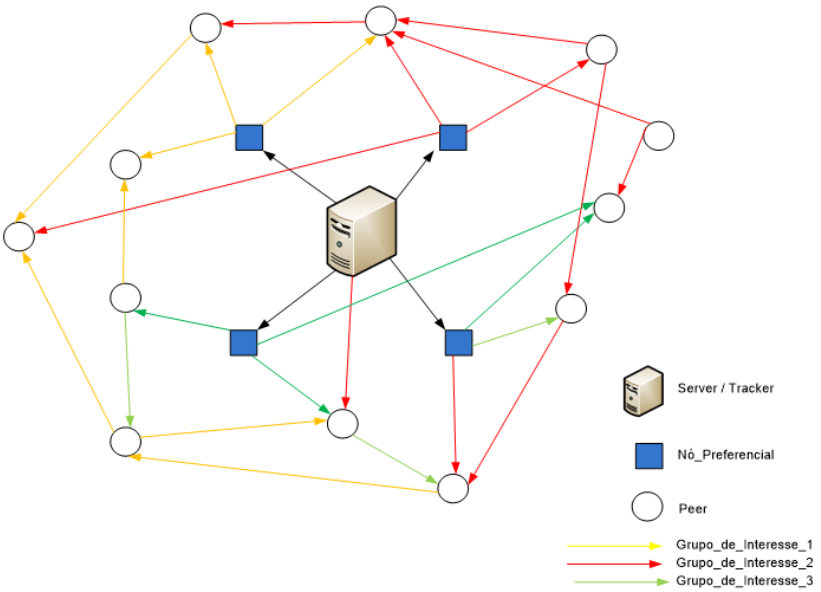

Figura 4.3: Arquitetura de *Streaming*

Listagem 4.5: Ficheiro de meta-datos do conteúdo com swarmID=UNDERTHESEASVC2SEC

```
<?xml version="1.0" encoding="UTF-8"?>
<StreamInfo version="2.0" xmlns="http://example.pt/schema/mpd-v2"
    PresentationType="OnDemand">
    <Trackers>
        <Tracker url="http://198.18.11.10:8070" tier="1"/>
        <Tracker url="http://198.18.14.10:8070" tier="1"/>
    </Trackers>
    <SwarmID>UNDERTHESEASVC2SEC</SwarmID>
    <Clip duration="PT03M21.05S">
        <BaseURL>streams/vod/svc/</BaseURL>
        <Description>Under The Sea 2009 (2sec) 2D SVC 250kbps to 3Mbps</Description>
        <Representation mimeType="video/SVC" codecs="svc" frameRate="23.976" width="640"
            height="360" bitRate="250000" bandwidth="250000" id="L0">
            <BaseURL>Under.The.Sea.720p_2D_SVC_2sec/</BaseURL>
            <SegmentInfo from="0" to="98" duration="PT2.00S">
                <BaseURL>Chunk2D-L0-</BaseURL>
            </SegmentInfo>
        </Representation>
        <Representation mimeType="video/SVC" codecs="svc" frameRate="23.976" width="640"
            height="360" bitRate="750000" bandwidth="750000" id="L1">
            <BaseURL>Under.The.Sea.720p_2D_SVC_2sec/</BaseURL>
            <SegmentInfo from="0" to="98" duration="PT2.00S">
                <BaseURL>Chunk2D-L1-</BaseURL>
            </SegmentInfo>
```

```
        </Representation>
        <Representation mimeType="video/SVC" codecs="svc" frameRate="23.976" width="1280"
            height="720" bitRate="1000000" bandwidth="1000000" id="L2">
            <BaseURL>Under.The.Sea.720p_2D_SVC_2sec/</BaseURL>
            <SegmentInfo from="0" to="98" duration="PT2.00S">
                <BaseURL>Chunk2D-L2-</BaseURL>
            </SegmentInfo>
        </Representation>
        <Representation mimeType="video/SVC" codecs="svc" frameRate="23.976" width="1280"
            height="720" bitRate="2000000" bandwidth="2000000" id="L3">
            <BaseURL>Under.The.Sea.720p_2D_SVC_2sec/</BaseURL>
            <SegmentInfo from="0" to="98" duration="PT2.00S">
                <BaseURL>Chunk2D-L3-</BaseURL>
            </SegmentInfo>
        </Representation>
    </Clip>
</StreamInfo>
```

4.3.2 Procedimento operacional

O procedimento operacional foi implementado como um conjunto de "serviços" no emulador, tal como indicado anteriormente, que se executam em cada nó da infraestrutura emulado de acordo com regras pre-definidas na configuração do cenário experimental, como por exemplo, a sequencia temporal de "boot" dos nós do sistema *PPSP*, onde os primeiros serão o Portal/*Tracker*, seguido pelo *Peer* SEED, e só depois os restantes *Peers*, sequencialmente ou em simultâneo (simulando uma "flash crowd").

Assim, após iniciar a emulação da rede, e verificado que o *Tracker* está contactável, as etapas para cada *peer* são as seguintes:

1. Um processo *Cliente*, autentica-se no sistema (para simplificação, assume-se que o utilizador correspondente a esse Cliente, efetuou previamente um Registo no sistema) e adquire um identificador de cliente que será o usado para associar o *Peer-ID* do processo *Peer* junto do *Tracker*. O *Peer-ID* é gerado associando o endereço IP e o registo temporal de acesso ao sistema;

2. O utilizador, através do processo *Cliente*, e já autenticado no sistema, seleciona um conteúdo para *Stream*, obtendo assim o ficheiro de meta-dados correspondente (*Swarm*), que identifica também o *Tracker* associado.

3. O processo *Peer* no cliente contacta então o *Tracker* apresentando-lhe o seu identificador de Cliente e o identificador do *Swarm* contido no ficheiro de meta-dados correspondente (como ilustrado na Listagem 4.5).

4. O *Tracker* solicita junto do Portal a validade do identificador do Cliente.

5. O Portal confirma a validade do identificador.

6. O *Tracker* fornece então ao *Peer* o seu *Peer-ID* e uma lista (de endereços IP) correspondente a um subconjunto de outros *Peers* ativos no *Swarm*, e que inclui o endereço do *Media Server*, que é um *Peer* de tipo *SEED*.

7. O *Peer* adere então ao *swarm*, para iniciar o *Stream* do conteúdo, partilhando os segmentos obtidos do mesmo com os outros *Peers*.

No início, pressupondo que os *Peers* arrancam em modo sequencial, o grupo de interesse (*swarm*) vai sendo formado com base na lista de *Peers* fornecido pelo *Tracker*.

Enquanto os *Peers* permanecerem no *Swarm*, eles tenderão a utilizar a melhor conexão possível para formarem grupos de interesse, conforme se ilustra na imagem Figura 4.3 da arquitetura de *Streaming*. Estes grupos de interesse podem ser baseados, por exemplo, nos nós de máxima interação ou (maior reputação). O protocolo atuará em duas etapas:

- Etapa 1: Enquanto não existe um número suficiente de *Peers* com (parte) do conteúdo, o *Peer SEED* disponibiliza diretamente e sem restrições os *Chunks* de conteúdo para os *Peers* tal como se fosse um *Media Server* tradicional numa arquitetura Cliente-Servidor.

- Etapa 2: Logo que os primeiros *Peers* já possuam *Chunks* de conteúdo, o *Media Server* manifesta-se ≪indisponível≫ para fornecer conteúdo a novos *Peers*, pelo que estes têm que estabelecer ligação com os outros *Peers* para aceder aos conteúdos de *Streaming*. Para benefício mútuo os nós que mais interagem no grupo de interesse, vão gerando indicadores de reputação que são comunicados ao *Tracker*.

4.4 Avaliação operacional dos mecanismos

Foi efetuada uma avaliação funcional aos mecanismos implementados, visto que o sistema PPSP, no que diz respeito à comunicação direta entre *Peers* cumpre com os objetivos definidos nas especificações do protocolo PPSPP, assegurando que o conteúdo da informação trocada entre *peers* não serão forjados em transito, e que a informação trocada entre os *peers* e os elementos lógicos do sistema (*Tracker* e Portal) serão agora (com a implementação efetuada neste trabalho)) transmitidos com segurança sob criptografia utilizando cifras.

Na base dos teste, começa-se por testar o funcionamento do protocolo HTTP seguro, em termos de autenticação dos *Peers*.

Listagem 4.6: Funcionamento do protocolo HTTPS em PPSP

```
1
2 root@PEER01:/tmp/pycore.55720/PEER01.conf# curl -v  -k -u PEER01
    -20:123456 https://198.18.11.10
3
4 * Rebuilt URL to: https://198.18.11.10/
5 * Hostname was NOT found in DNS cache
6 *   Trying 198.18.11.10...
7 * Connected to 198.18.11.10 (198.18.11.10) port 443 (#0)
8 * successfully set certificate verify locations:
9 *   CAfile: none
10   CApath: /etc/ssl/certs
11 * SSLv3, TLS handshake, Client hello (1):
12 * SSLv3, TLS handshake, Server hello (2):
13 * SSLv3, TLS handshake, CERT (11):
14 * SSLv3, TLS handshake, Server key exchange (12):
15 * SSLv3, TLS handshake, Server finished (14):
16 * SSLv3, TLS handshake, Client key exchange (16):
17 * SSLv3, TLS change cipher, Client hello (1):
18 * SSLv3, TLS handshake, Client Finished (20):
19 * SSLv3, TLS handshake, Finished (20):
20 * SSL connection using ECDHE-RSA-AES256-GCM-SHA384
21 * Server certificate:
22 *        subject: C=PT; ST=Lisboa; L=Lisboa; O=ISCTE; OU=ISCTE; CN
    =198.18.11.10
23 *        start date: 2017-07-07 00:17:43 GMT
24 *        expire date: 2020-10-19 00:17:43 GMT
25 *        issuer: C=PT; ST=Lisboa; L=Lisboa; O=ISCTE; OU=ISCTE; CN
    =198.18.11.10
26           SSL certificate verify result: self signed certificate
                (18), continuing anyway.
27 * Server auth using with user 'PEER01-20'
28 > GET / HTTPS/1.1
29 > Authorization: UEVFUjAxLTIwOjEyMzQ1Ng==
30 > User-Agent: curl/7.35.0
```

```
31 > Host: 198.18.11.10
32 > Accept: */*
33 >
34 < HTTPS/1.1 200 OK
35 < Date: Wed, 04 Oct 2017 18:55:12 GMT
36 * Server Apache/2.4.7 (Ubuntu) is not blacklisted
37 < Server: Apache/2.4.7 (Ubuntu)
38 < Last-Modified: Wed, 04 Oct 2017 17:26:05 GMT
39 < ETag: "109-55abbe71261c5"
40 < Accept-Ranges: bytes
41 < Content-Length: 265
42 < Vary: Accept-Encoding
43 < Content-Type: text/html
44 <
45 <html><body><!-- generated by utility.py:HttpService -->
46 <h1>PORTAL web server</h1>
47 <p>This is the default web page for this server.</p>
48 <p>The web server software is running but no content has been added,
      yet.</p>
49 <li>eth0 - ['198.18.11.10/24']</li>
50 * Connection #0 to host 198.18.11.10 left intact
51 </body></html>
```

Na linha número 1 da Listagem 4.6, observa-se um processo cliente fazendo o pedido de autenticação ao servidor, usando o seu *username* e a *password*, a que se segue o resultado do processo subsequente, onde se pode observar da linha 9 até à linha 17, a interação entre o processo cliente e o servidor. Esta interação é feita utilizando um *Handshake* seguro para certificar que ambos estão conectados e já podem manter contacto, utilizando protocolos TLS e Secure Socket Layer (SSL), que são os responsáveis por criptografar toda comunicação que circula a nível do sistema.

Uma vez validada as credenciais do processo cliente, é atribuída uma chave, representada na linha 20, que posteriormente é criptografada para se obter o *Hash*, que é a chave de sessão representada na linha 29.

4.4.1 Negação de acesso

O acesso não autorizado, é o resultado da utilização de credenciais incorretas, que acontece quando um processo cliente executa uma solicitação ao servidor, utilizando uma entidade não válida pelo sistema, logo que isso acontece o servidor recusa a autorização deste processo cliente, assim como demonstra na linha número 1 da Listagem 4.7.

Listagem 4.7: Exemplo de resposta de negação de acesso ao Peer

```
1
2  < HTTPS/1.1 401 Unauthorized
3  < Date: Thu, 05 Oct 2017 19:57:32 GMT
4  * Server Apache/2.4.7 (Ubuntu) is not blacklisted
5  < Server: Apache/2.4.7 (Ubuntu)
6  * Authentication problem. Ignoring this.
7  < WWW-Authenticate: Basic realm="Auth Area"
8  < Content-Length: 459
9  < Content-Type: text/html; charset=iso-8859-1
10 <
11 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
12 <html><head>
13 <title>401 Unauthorized</title>
14 </head><body>
15 <h1>Unauthorized</h1>
16 <p>This server could not verify that you
17 are authorized to access the document
18 requested.  Either you supplied the wrong
19 credentials (e.g., bad password), or your
20 browser doesn't understand how to supply
21 the credentials required.</p>
22 <hr>
23 <address>Apache/2.4.7 (Ubuntu) Server at 198.18.11.10 Port 443</
      address>
24 </body></html>
```

4.4.2 Identificador do *Peer*

Dado que os nós do emulador CORE não possuem uma interface gráfica de interação durante a execução do sistema, apenas uma "shell" de Terminal tipo BASH, os detalhes de execução que se apresentam foram obtidos diretamente nesses nós, em sessão de Terminal.

Quando o processo *Peer* no cliente, solicita acesso ao "swarm" por ligação ao *Tracker*, depois de se verificar a identidade do processo cliente, o *Tracker* já pode fornecer ao *Peer* um identificador para o "swarm", que é o *Peer-ID*, dando-lhe acesso ao *Streaming*, descrito na linha número 16 da Listagem 4.8.

Listagem 4.8: Aquisição de um identificador do *Peer03* numa sessão de um "swarm"

```
1
2 root@PEER03:/tmp/pycore.53223/PEER03.conf# cat prefs.xml
3
4 <pref version="1" name="/">
5 <pref name="core" >
6 <pref name="bind_port" type="string" value="8070" />
7 </pref>
8 <pref name="plugins" >
9 <pref name="core" />
10 </pref>
11 <pref name="peer" >
12 <pref name="sleep_interval" type="int" value="10" />
13 <pref name="session"/>
14 </pref>
15 <pref name="is_interested_timeout" type="int" value="10" />
16 <pref name="peer_id" type="string" value="NjkxYzcyNjM1OTA1MTExMTc0MzFiMDU1"
     />
17 <pref name="port" type="string" value="8070" />
18 <pref name="cache" type="string" value="cache" />
19 <pref name="storage" type="string" value="storage" />
20 <pref name="loaded_descriptors" type="string" value="loaded" />
21 </pref>
22 </pref>
```

Tal como já referido no procedimento operacional, depois do utilizador através do processo cliente obter o identificador ou a chave, existe um processo no lado do cliente que fala com o *Tracker*, apresentando o identificador obtido no momento em que o processo cliente se autenticou, e o *Tracker* deve

solicitar ao portal para saber se o identificador existe ou é válido.

Um dos pontos cruciais desta avaliação, foi de observar como está sendo feita a atribuição de identificadores aos *Peers*, de modo a garantir que dois *Peers* não tenha o mesmo identificador, ou então garantir que após o *Time Out* do *Peer*, na próxima comunicação lhe é atribuído um identificador diferente do identificador utilizado na sessão anterior.

Para demonstrar o correto funcionamento fez-se um teste com o mesmo *Peer03* em sessões diferentes, e verificamos que a atribuição de identificador aos *Peers* está sendo feita corretamente, tal como se pode confirmar comparando as linhas número 16 da Listagem 4.8 e da Listagem 4.9.

Listagem 4.9: Aquisição de um identificador do *Peer03* numa outra sessão de outro "swarm"

```
1
2 root@PEER03:/tmp/pycore.44510/PEER03.conf# cat prefs.xml
3
4 <pref version="1" name="/">
5   <pref name="core" >
6     <pref name="bind_port" type="string" value="8070" />
7   </pref>
8   <pref name="plugins" >
9     <pref name="core" />
10  </pref>
11  <pref name="peer" >
12    <pref name="sleep_interval" type="int" value="10" />
13    <pref name="session" />
14    </pref>
15    <pref name="is_interested_timeout" type="int" value="10" />
16    <pref name="peer_id" type="string" value="
          YzMyNmI1ODE2M2NkNjhhM2JiMTljNTY1" />
17    <pref name="port" type="string" value="8070" />
18    <pref name="cache" type="string" value="cache" />
19    <pref name="storage" type="string" value="storage" />
20    <pref name="loaded_descriptors" type="string" value="loaded" />
21  </pref>
22 </pref>
```

4.4.3 Download do conteúdo de vídeo

Após o processo de autenticação e troca de chaves entre o servidor e o processo*Peer* no cliente, os agentes do protocolo do *Peer*, auxiliam para realizar o processo final da comunicação. Na listagem Listagem 4.11 e na Listagem 4.10 apresentam-se os conteúdos das pastas "HOME" dos *Peers*, onde se observam o *load* os ficheiros de arranque, de configuração e de "log" gerados durante a execução. Nas linhas 2 e 3 da Listagem 4.10, temos a representação dos *Hash*, como identificação dos respetivos *Peers*, os "scripts" de configuração de ligação `conflinkadsl.sh` e `defaultroute.sh`, e o "script" de arranque `startpeer.sh`.

Listagem 4.10: Exemplo de um ficheiro de vídeo do tipo 32

```
1 root@PEER02:/tmp/pycore.53223/PEER03.conf# ls
2 32_sea_2s_d80s_svc_p2p_emu.xml  loaded
3 NjkxYzcyNjM1OTA1MTExMTc0MzFiMDU1.pid   peer-agent-plugins        prefs.xml
4 cache                                  peer_agent                startpeer.sh
5 conflinkadsl.sh                        peer_engine_client_retry  var.log
6 defaultroute.sh                        player_emulator           var.run
```

Listagem 4.11: Exemplo de um ficheiro de vídeo do tipo 30

```
1 root@PEER04:/tmp/pycore.53223/PEER04.conf# ls
2 30_sea_2s_svc_p2p_emu.xml  loaded
3 M2RjNWE3ZTE4MmMwNzg3MzZkNzE2YTc0.pid   peer-agent-plugins        prefs.xml
4 cache                                  peer_agent                startpeer.sh
5 conflinkadsl.sh                        peer_engine_client_retry  var.log
6 defaultroute.sh                        player_emulator           var.run
```

5

Conclusão

Conteúdo

5.1 Discussão 69
5.2 Recomendações 69

5.1 Discussão

Após as pesquisas feitas, verificou-se, que é tendência atual utilizar as redes *P2P* para a difusão de conteúdos multimédia na *Internet*, pois elas proporcionam aos seus utilizadores uma gama incrível de recursos, que podem ser divididos entre os vários *Peers* componentes da rede.

Apesar da sua grande utilidade e a capacidade de proporcionar recursos aos seus utilizadores este tipo de rede é sensível a faltas de segurança e a vulnerabilidades, que requer a implementação de mecanismos seguros de forma a garantir um sistema de *streaming* de conteúdos eficiente e com a melhor qualidade possível.

Na sequencia do estudo sobre os mecanismos adequados, implementou-se um demonstrador que evidencia o funcionamento de alguns dos mecanismos, recorrendo a um ambiente emulado, funcionando em tempo real. Como as redes *P2P* são amplamente utilizadas para difusão de conteúdos multimédia, elas precisam estar protegidas de modo a garantir que os dados trafeguem com segurança, e para que tal objetivo seja alcançado num ambiente funcional, é necessário que se atinja um nível de segurança, tal como o que foi ilustrado com a demonstração.

De forma geral, podemos considerar que o objetivo subjacente para a realização deste trabalho, foi alcançado de forma satisfatória, e já se pode considerar como um contributo válido para a transmissão de conteúdos multimédia em rede de natureza P2P, com o suporte do protocolo PPSP.

5.2 Recomendações

Além dos métodos de segurança identificados, para a construção de um sistema *streaming* seguro em aplicações P2P, que é garantido utilizando o mecanismo de autenticação, que foi ilustrado com o demonstrador, outras questões são consideradas cruciais, para garantir a segurança nas aplicação P2P, tais como o cálculo de reputação dos *Peers*.

Tendo em conta que a implementação de um sistema nunca chega ao fim, sendo sempre alvo de melhorias, por esta razão os sistemas devem ser escaláveis de forma a albergar as novas funcionalidades, sugerimos também a inclusão de um sistema de gestão de reputação, que permite calcular a reputação dos *Peers*, para estabelecer um grau de confiança entre as entidades no sistema.

Esta abordagem permite a atribuição de pesos aos nós que mais interagem no *Overlay* no sentido de incentivá-los a continuarem a colaborar com os outros *Peers* na rede, assim deve ser associado ao sistema de gestão de reputação, o cálculo de *Ranking* de reputação e contadores de *Streaming*.

Bibliografia

[1] F. J. Vilanova, L. Z. Granville, and M. J. Bosquiroli, “Uma Ferramenta Peer-to-Peer para Gerenciamento Cooperativo de Redes,” no. 1995, 2006.

[2] G. L. P. Barcellos. Marinho P and L. Gaspary, “Segurança em Redes P2P: Princípios, Tecnologias e Desafios,” Ph.D. dissertation, 2006.

[3] R. Cruz, M. Nunes, J. Xia, R. Huang, J. Taveira, and D. Lingli, “Peer-to-Peer Streaming Tracker Protocol (PPSTP),” RFC 7846, 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7846.txt

[4] Y. Zhang and N. Zong, “Problem Statement and Requirements of the Peer-to-Peer Streaming Protocol (PPSP),” Internet Requests for Comments, RFC Editor, Tech. Rep. 6972, 2013.

[5] V. Grishchenko, A. Bakker, and R. Petrocco, “Peer-to-Peer Streaming Peer Protocol (PPSPP),” RFC 7574, Tech. Rep. 7574, 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7574.txt

[6] R. Santos Cruz, M. S. Nunes, C. Z. Patrikakis, and N. C. Papaoulakis, “SARACEN: A platform for adaptive, socially aware multimedia distribution over P2P networks,” *2010 IEEE Globecom Workshops, GC’10*, no. February, pp. 1356–1360, 2010.

[7] J. Schäfer, K. Malinka, and P. Hanáček, “Peer-to-peer networks security,” *Proceedings - The 3rd International Conference on Internet Monitoring and Protection, ICIMP 2008*, pp. 74–79, 2008.

[8] G. Camarillo, “Peer-to-Peer (P2P) Architecture,” pp. 1–26, 2009.

[9] S. Androutsellis Theotokis and D. Spinellis, “A survey of peer-to-peer content distribution technologies,” *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[10] D. Tsoumakos and N. Roussopoulos, “A Comparison of Peer to Peer Search Methods,” *Proceedings of the 6th WebDB Workshop*, pp. 61–66, 2003.

[11] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with CFS,” *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, p. 202, 2001.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-Addressable Network,” *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, vol. 31, no. 4, pp. 161–172, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?doid=964723.383072

[13] P. Maymounkov and D. Mazières, “Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers,” P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, ch. Kademlia:, pp. 53–65. [Online]. Available: http://dx.doi.org/10.1007/3-540-45748-8_5

[14] B. Cohen, “Incentives Build Robustness in BitTorrent,” *Workshop on Economics of PeertoPeer systems*, vol. 6, p. 68–72, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.1911&rep=rep1&type=pdf

[15] O. Awodele, E. E. Onuiri, and S. O. Okolie, “Vulnerabilities in Network Infrastructures and Prevention / Containment Measures,” 2012.

[16] H. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking,*, 1st ed., 2009, vol. 53, no. 9.

[17] M. Haridasan and R. van Renesse, “SecureStream: An intrusion-tolerant protocol for live-streaming dissemination,” *Computer Communications*, vol. 31, no. 3, pp. 563–575, 2008.

[18] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, “The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses,” *P2P-TV’07 Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pp. 323–328, 2007.

[19] E. Sit and R. Morris, “Security Considerations for Peer-to-Peer Distributed Hash Tables,” *Peer-to-Peer Systems*, vol. 2429, pp. 261–269, 2002. [Online]. Available: http://www.springerlink.com/content/h414afjf3a3avnlm

[20] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach, “Eclipse attacks on overlay networks: Threats and defenses,” *Proceedings - IEEE INFOCOM*, 2006.

[21] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, “Multicast security: a taxonomy and some efficient constructions,” *IEEE INFOCOM ’99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 2, pp. 1–9, 1999.

[22] R. V. Coelho, M. P. Barcellos, I. Jansch-pôrto, and L. Gaspary, “P2P Streaming de Alta Definição : Análise Quantitativa de Esquemas de Assinatura Digital para Autenticação de Conteúdo,” pp. 3–16.

[23] L. Reyzin and N. Reyzin, “Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying,” *Information Security and Privacy*, vol. 2384, pp. 1–47, 2002. [Online]. Available: http://dx.doi.org/10.1007/3-540-45450-0_11

[24] R. Vieira Coelho, M. P. Barcellos, I. Jansch-Pôrto, and L. Gaspary, “P2P Streaming de Alta Definição: Análise Quantitativa de Esquemas de Assinatura Digital para Autenticação de Conteúdo.”

[25] Y.-J. Lin, S. Shieh, and W. W. Lin, “Lightweight, pollution-attack resistant multicast authentication scheme,” *ASIACCS’06 Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, p. 148, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1128817.1128840

[26] W. W. Lin, S. Shieh, and J. C. Lin, “A pollution attack resistant multicast authentication scheme tolerant to packet loss,” *Proceedings - The 2nd IEEE International Conference on Secure System Integration and Reliability Improvement, SSIRI 2008*, pp. 8–15, 2008.

[27] R. Meier and R. Wattenhofer, “ALPS: Authenticating live peer-to-peer streams,” *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, pp. 45–52, 2008.

[28] S. Dhumal and P. S. Raksha, “DIGITAL RIGHTS MANAGEMENT SYSTEM FOR,” vol. 2, no. 3, pp. 18–22, 2015.

[29] T. B. Zahariadis, “SEA : SEAmless content delivery What is SEA,” no. November, pp. 1–9, 2007.

[30] E. Leonardi, P. Torino, and R. A. L. Cigno, “Architecture of a Network-Aware P2P-TV Application : The NAPA-WINE Approach,” no. January, 2011.

[31] R. Jimenez, L. Eriksson, and B. Knutsson, “P2P-Next: technical and legal challenges,” *Research funded by Seventh Framework Programme*, 2009. [Online]. Available: http://www.tslab.ssvl.kth.se/files/sncnw_main.pdf

[32] F. Sales, D. L. Filho, F. Sales, and D. L. Filho, “Mecanismos de segurança para um sistema cooperativo de armazenamento de arquivos baseado Mecanismos de segurança para um sistema cooperativo de armazenamento de arquivos baseado em P2P,” 2010.

[33] R. V. Coelho, “Comparação Analítica dos Esquemas de Autenticação em Sistemas P2P de Live Streaming,” p. 76, 2011.

[34] D. Hardt, “The OAuth 2.0 Authorization Framework,” pp. 1–76, 2012.

[35] A. Detsch and L. Gaspary, “Uma Arquitetura para Incorporação Modular de Aspectos de Segurança em Aplicações Peer-to-Peer,” Ph.D. dissertation, 2005. [Online]. Available: http://bdtd.unisinos.br/tde_arquivos/1/TDE-2006-08-14T110133Z-70/Publico/umaarquitetura.pdf

[36] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach, “Secure routing for structured peer-to-peer overlay networks,” *In OSDI ’*, vol. 02, no. December, p. 16, 2002.