ESR

Cap1 - Camada de aplicação

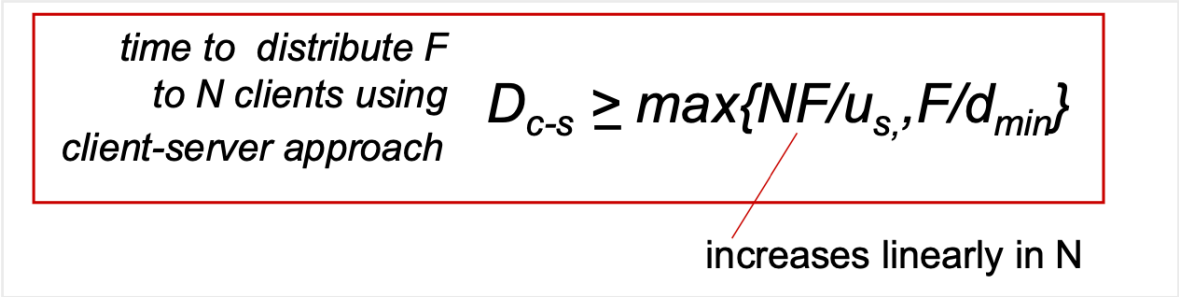
Arquitetura cliente-server

`Servidor – aguarda para ser contactado`

- Host sempre ativo
- Endereço ip permanente
- Data centers para escalabilidade

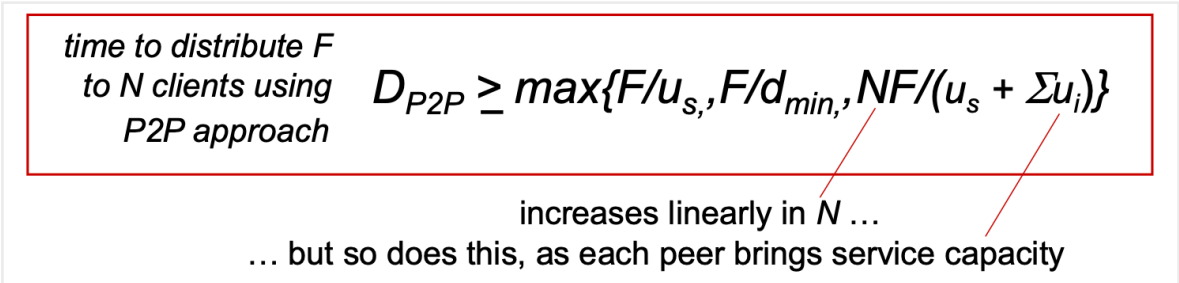
`Cliente – inicia a comunicação`

- Comunicam com o servidor(pode ser intermitente)
- Podem ter IP dinâmicos
- Não comunicam diretamente entre si



P2P

- Não há servidor sempre ativo
- Sistemas finais arbitrários comunicam diretamente
- Peers solicitam serviço a outros peers e também fornecem
  Autoescalabilidade - novos peers trazem mais capacidade ao servidor
- Peers estão conectados intermitentemente e mudam o endereço IP frequentemente
  Gestão complexa



Processo de comunicação - se os programas estiverem a correr no mesmo host comunicam através de um processo interno (definido pelo OS), se estiverem em hosts diferentes comunicação através de mensagens(socket)

Endereço de processos - Para receber mensagens, um processo precisa de um identificador(inclui IP e o nº da porta)

Uma aplicação precisa dos seguintes serviços de transporte:

Integridade de dados (transferencias de arquivos)
Tempo (baixa latência)
Taxa de transferência
Segurança

Serviços de protocolo de transporte
TCP
- Confiável
- Controlo de fluxo
- Controlo de congestionamento
- Orientado à conexão: necessita de um setup entre processos cliente-servidor

UDP
- Não é confiável
- **Não fornece:**
  - confidencialidade
  - controlo de fluxo/congestionamento
  - tempo
  - garantia de taxa de transferência
  - segurança

Segurança TCP
- Sockets tcp e dup padrão - sem criptografia
- Segurança da camada de transporte (TLS)
  - Fornece conexões tcp criptografadas
  - Integridade dos dados
  - Autenticação end-point

Aplicações usam bibliotecas TLS que usam TC por sua vez

Exemplo de uma aplicação P2P(Bitorrent)
Os peers **contribuem** para o envio de chunks enquanto **baixam**, criando uma rede colaborativo. Os peers podem **entrar e sair** sem comprometer o funcionamento.
Os peers são registados com um **tracker** que lhe fornece a lista de outros peers
Há duas estreias principais
- **Requisição de chunks** - Peer solicita os chunks que ainda não tem
- **Tik-for-tat** - Peer envia chunks para o seu top4, esse top4 pode ser alterado(através de **desbloqueio otimista - começa a enviar para outro peer aleatoriamente**) se um novo peer enviar mais rápido

Transmissão de video : DASH
Permite ao cliente ajustar dinamicamente a qualidade do video de acordo com as condições da rede. A transmissão é feita em chunks e o cliente invés de

pedir o video todo, solicita chunks com a maior qualidade possível suportada pela conexão, garantindo uma experiência continua mesmo que a largura de banda varie.
É o cliente que :

- **Pede o chunks** (starvation (quando pede muito tarde), overflow (quando os pede muito cedo) ou idealmente)
- Escolhe a **qualidade** dos chunks (+ ou - bits)
- Escolhe **de onde vem** os chunks (geograficamente)

CDN - Content Delivery Network
É uma rede de **servidores distribuídos geograficamente** que trabalham juntos para entregar conteúdo de forma rápida e eficiente. Os servidores CDN **armazenam cópias** do conteúdo, em vários locais (**PoP's**). Quando um utilizador solicita o conteúdo, este é **servido pelo PoP mais próximo**, reduzindo a latência, aumentando a velocidade do carregamento e minimizando o congestionamento da rede.

OTT -> serviços que não possuem redes físicas para transmitir os dados, dependem de terceiros para garantir que o conteúdo chega aos utilizadores

Programação de sockets com UDP
Servidor

- Cria um socket e fica à escuta numa porta especifica

Cliente

- Cria um socket UDP e envia pacotes para o servidor especificando o endereço IP e a porta que o servidor está à escuta

Comunicação

- Não há nenhum handshake, o cliente envia, o servidor recebe e envia de volta

Encerramento

- Não é formal, apenas deixam de enviar pacotes

Programação de sockets com TCP
Servidor

- Cria um socket e aguarda conexões na porta especifica

Cliente

- Cria um socket TCP e tenta se conectar ao servidor especificando o endereço IP e a porta que o mesmo está à escuta

Estabelecimento da conexão
O TCP realiza um processo de handshake de 3 vias para garantir a conexão de forma confiável
Comunicação
O servidor cria um novo socket para comunicar individualmente com o cliente, permitindo múltiplas conexões simultaneamente
Os dados são transmitidos de forma confiável e em ordem
Encerramento

- Após a troca de dados, a conexão é encerrada libertando os recursos

(FIN / ACK e FIN / ACK)

Cap2 - Multicast

Aplicações e Serviços na Internet geralmente têm como objetivo:

- Atender **utilizadores individuais** ou **grupos de utilizadores**.
- Dependendo do cenário, isso pode exigir diferentes modelos de comunicação:
  - **Ponto-a-ponto (point-to-point)**: Comunicação direta entre dois dispositivos.
  - **Ponto-a-multiponto (point-to-multipoint)**: Um emissor envia informações para vários recetores simultaneamente.
  - **Multiponto-a-multiponto (multipoint-to-multipoint)**: Vários emissores e recetores interagem uns com os outros.

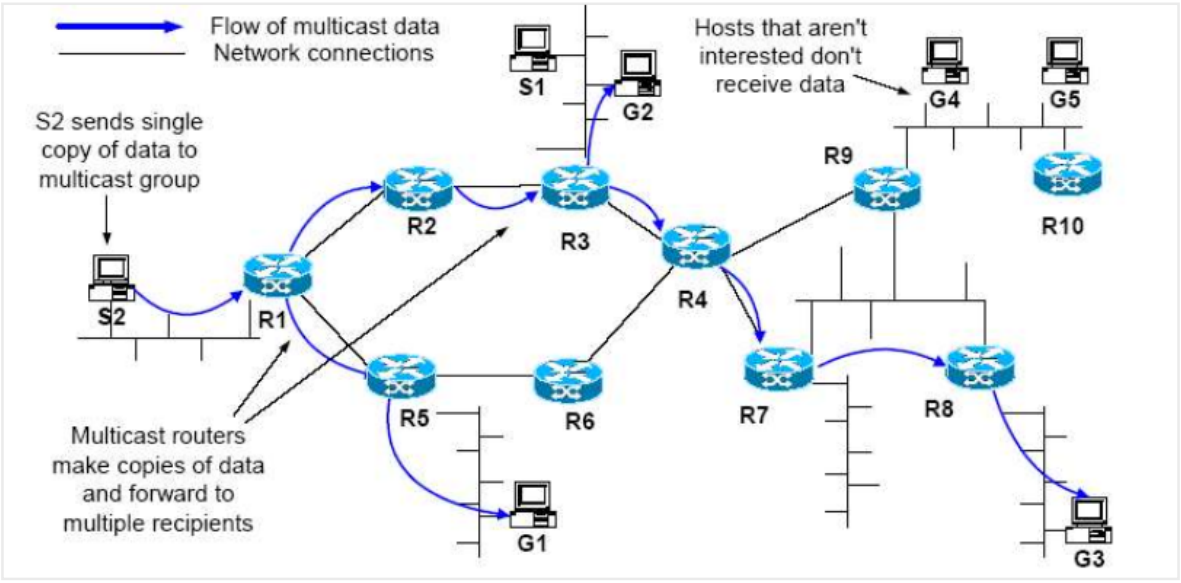
Problema dos Modelos Baseados em Unicast:

- Utilizar **vários canais unicast** (comunicação direta para cada recetor) para entregar conteúdos a múltiplos utilizadores é **ineficiente**:
  - **Consome muitos recursos**, tanto em termos de **processamento** quanto de **transmissão**.
  - Pode levar a **desempenho reduzido** da rede.

Solução:

- **Modelo de Entrega Multicast**:
  - Propõe uma abordagem mais eficiente para a entrega de dados ou conteúdos multimédia.
  - Reduz o consumo de recursos ao enviar uma única transmissão para vários recetores simultaneamente.
  - Embora seja instanciado **ao nível da rede** neste caso, os conceitos podem ser aplicados em **camadas superiores** (como na camada de aplicação).

O multicast simplifica a transmissão de dados para grupos ao usar um único endereço multicast e delegar aos nós da rede a responsabilidade de encaminhar e duplicar pacotes conforme necessário(através de um **árvore de distribuição**). Isso reduz a complexidade e melhora a eficiência da comunicação



Protocolos como **IGMP (IPv4) e MLD (IPv6)** permitem que hosts se inscrevam nos grupos multicast, comunicando as suas intenções aos roters locais. Os roters gerem essas associações e verificam regularmente se os hosts ainda estão ativos nos grupos(enviando mensagens query). O IGMP evoluiu para reduzir a latência e melhorar a eficiência, introduzindo temporizadores para evitar respostas redundantes

IGMPv1:

- Depende do **protocolo de roteamento multicast** para decidir qual roteador atuará como **querier** (o roteador responsável por enviar as mensagens Query).

IGMPv2:

- Introduz um **processo de eleição de querier**, no qual os roteadores determinam automaticamente qual será o responsável por enviar as mensagens Query
- mensagens Leave Group para saída rápida de grupos

ASM (Any Source Multicast) - recebe dados do grupo vindo de qualquer fonte
SSM (Source-specific multicast) - apenas recebe dados de uma fonte especifica (requer o use de source based porque obriga aos dados a irem por um roter)

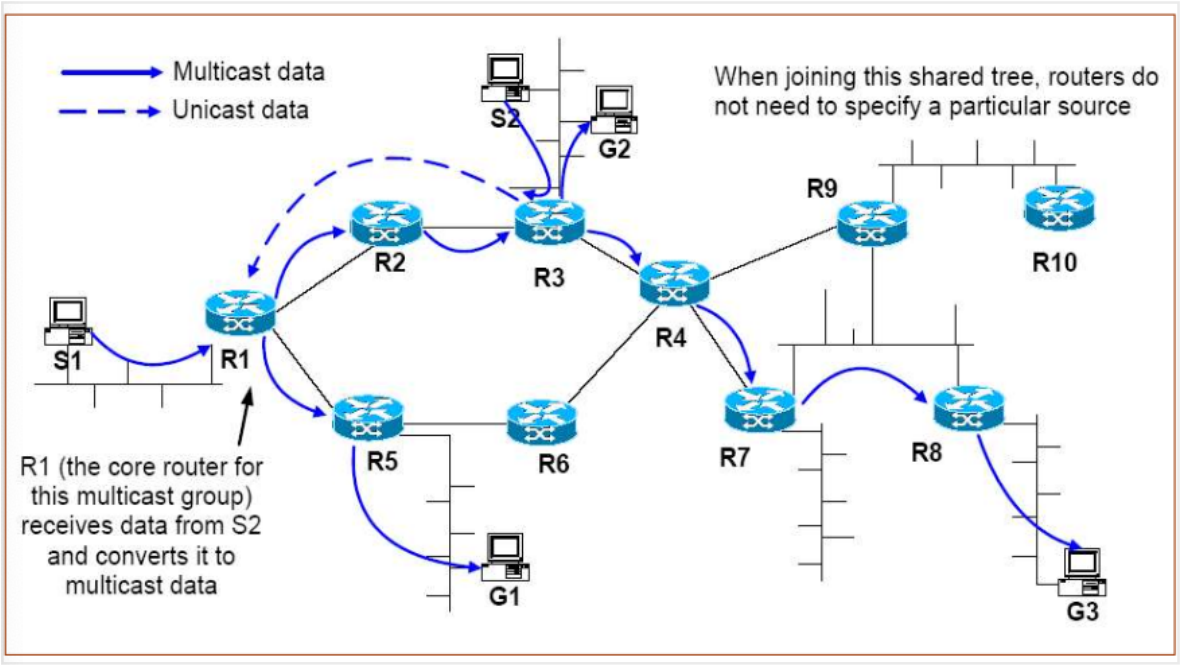
A escolha do design multicast depende do protocolo

- **opt-in** para otimização no momento da associação(só é iniciado quando se junto ao grupo multicast)
- **opt-out** para simplificação inicial porque todos são adicionados e são removidos se não tiverem interesse)
-

e do tipo de árvore

- **source-based** para maior eficiência no tráfego (cada fonte tem um arvore)

- **shared trees** para simplicidade (enraizada num **único ponto RP - Rendezvous Point**)



O algoritmo **Reverse Path Forwarding (RPF)** é utilizado para evitar loops e duplicações de pacotes, garantindo que os pacotes de controle e dados sigam a trajetória correta. Para isso, **protocolos como PIM** utilizam uma tabela chamada **MRIB**, que funciona de maneira similar à tabela de routing unicast, mas focada no tráfego multicast.

Há dois principais protocolos PIM:

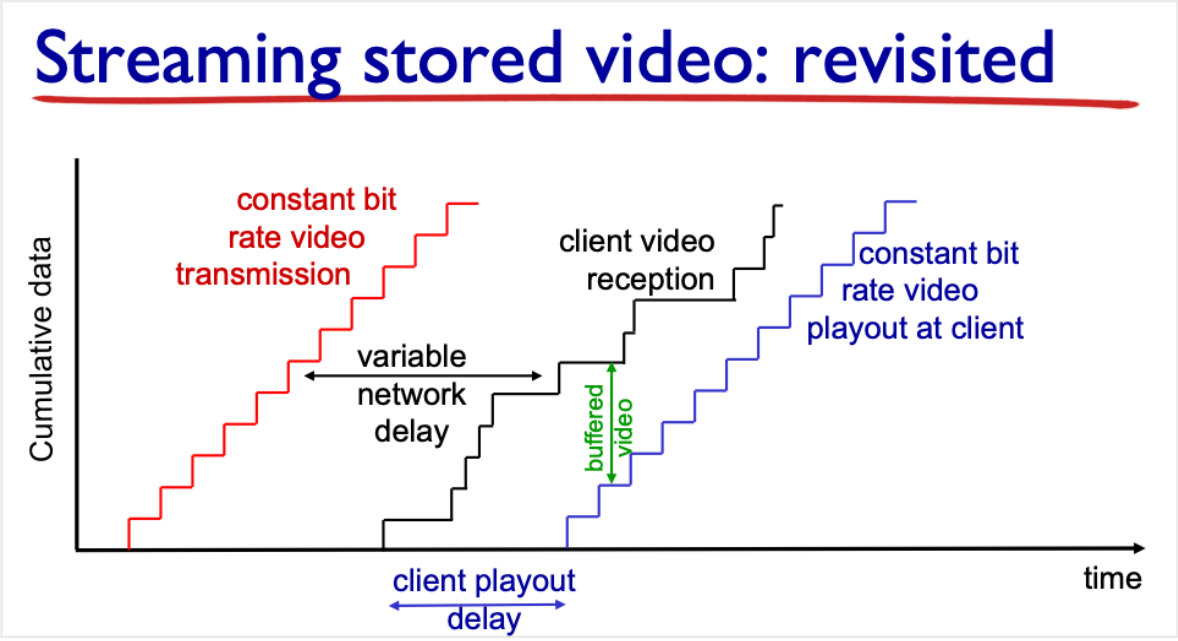
- **PIM Sparse Mode**, os routers apenas encaminham o tráfego multicast após uma solicitação, sendo ideal para ambientes com poucos receptores de multicast. Por default usa shared trees mas também suporta source based

- **PIM Dense Mode** envia o tráfego multicast para todas as interfaces inicialmente, e só interrompe o envio quando solicitado para parar(PIM(**S**ource, **G**roup) prune message), sendo mais adequado para redes pequenas ou domínios individuais (broadcast)

Cap3 - Rede Multimédia

Os dois tipos principais de redundância em vídeo são **redundância espacial** e **redundância temporal**. Ambas podem ser exploradas para compactar vídeos de forma eficiente, reduzindo a quantidade de dados a serem transmitidos ou armazenados sem perder significativamente a qualidade.

Taxa de bits para PCM
R=fs(frequência da amostra) ×N(numero de bits da amostra)



1. **Atraso variável na rede (variable network delay):**
   - Existe uma **separação** entre a curva de transmissão (vermelha) e a curva de recepção (preta), indicando atrasos e irregularidades na entrega dos dados devido à rede.
2. **Buffering de vídeo (buffered video):**
   - O vídeo é armazenado temporariamente no buffer do cliente (indicado pela linha verde vertical) antes de ser reproduzido.
   - Isso ajuda a compensar os atrasos variáveis da rede.
3. **Atraso de reprodução no cliente (client playout delay):**
   - Antes de iniciar a reprodução (linha azul), o cliente aguarda para preencher o buffer com uma quantidade mínima de dados.
   - Esse atraso ajuda a manter uma reprodução **suave e contínua**, mesmo que ocorram atrasos na rede.

Streaming multimedia: UDP

- O **UDP** é usado em streaming multimídia por sua **baixa latência**.
- O servidor transmite em uma taxa constante, com um pequeno atraso para compensar jitter.
- A recuperação de erros é limitada ao nível da aplicação para manter a reprodução fluida.
- Protocolo **RTP** gere o transporte e monitorização, enquanto **RTSP** controla a sessão de reprodução

Streaming multimedia: HTTP

- **HTTP Streaming** utiliza **TCP** para transmitir arquivos multimídia.
- A taxa de envio depende do controle de congestionamento do TCP e pode variar.
- Se a **taxa de consumo(r)** for menor que a **taxa de entrega(x(t))**, a reprodução será contínua, com um pequeno atraso inicial.

- Se a **taxa de consumo(r)** for maior que a **taxa de entrega(x(t))**, o **buffer esvazia** e o vídeo pode **travar** (stall)
- A **taxa de envio do servidor** é limitada pela **taxa de consumo** do vídeo no cliente (a taxa de reprodução).
- Isso garante que o servidor não sobrecarregue a rede ou o cliente, mantendo o streaming equilibrado

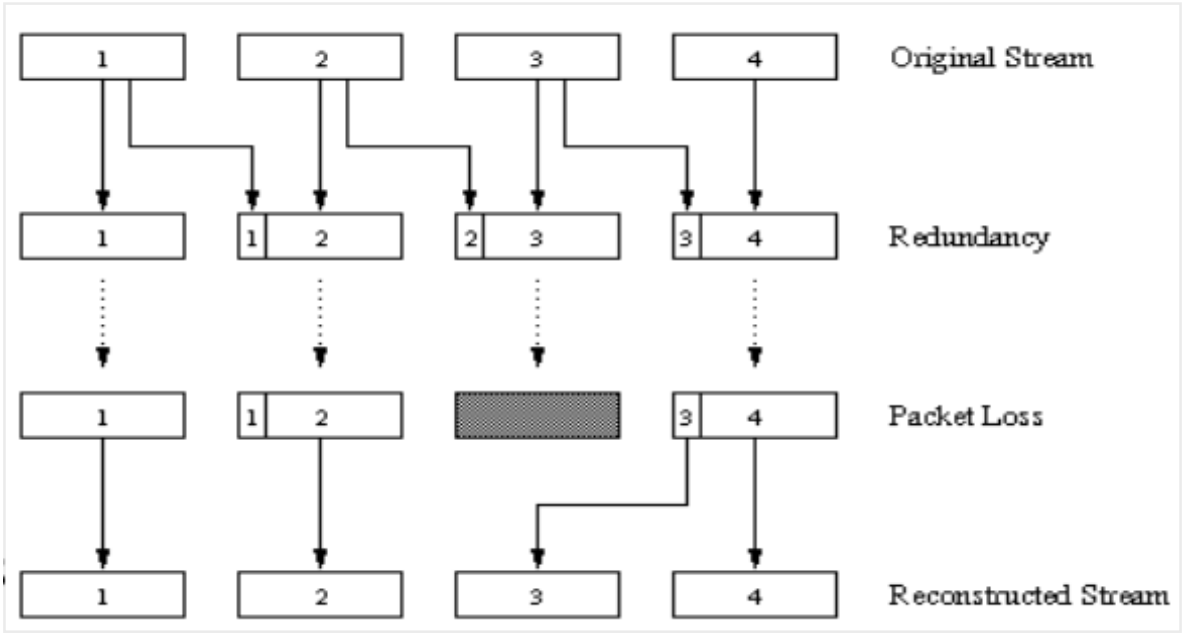
Voice-over-IP (VoIP)

- **Atraso**: Deve ser mantido abaixo de **150 ms** para garantir uma experiência de conversa natural.
- **Inicialização da Sessão**: Utiliza protocolos como **SIP** e **SDP** para troca de informações sobre IP, portas e codificação.
  - **SIP (Session Initiation Protocol)**: Estabelece, mantém e termina sessões de VoIP.
  - **SDP (Session Description Protocol)**: Usado em conjunto com SIP para anunciar endereços, portas e formatos de codificação.
- **Serviços adicionais**: Incluem funcionalidades como **encaminhamento**, **redirecionamento** e **gravação** de chamadas, tornando o VoIP uma solução flexível e rica em recursos.

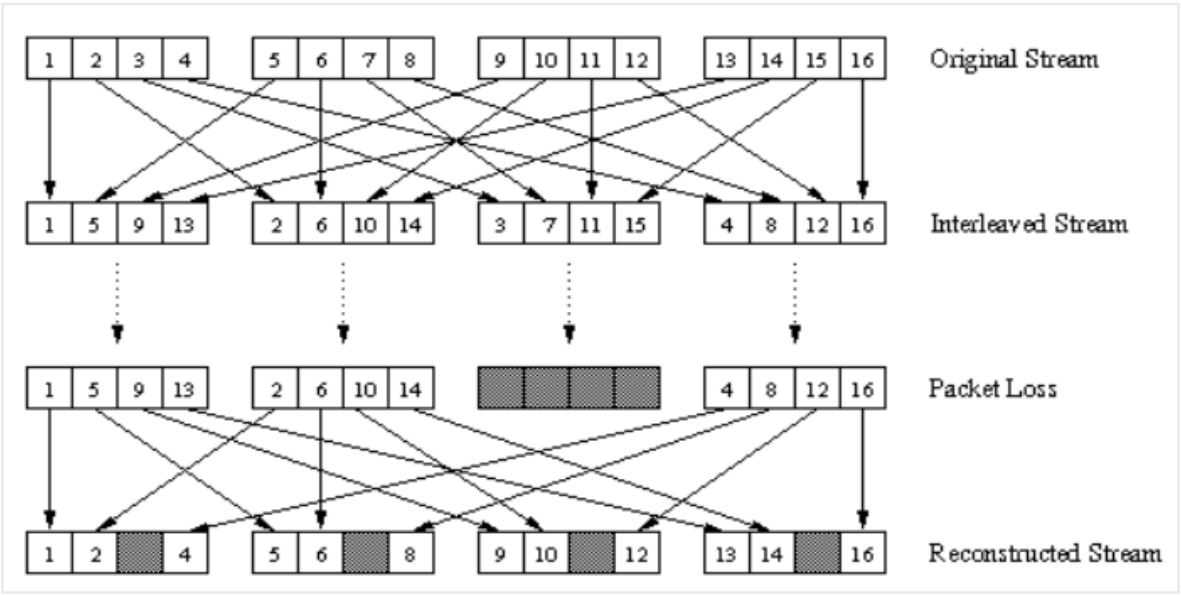
O **fixed playout delay** é uma técnica usada para melhorar a qualidade das chamadas VoIP ao compensar variações no tempo de chegada dos pacotes de dados, garantindo uma reprodução de áudio mais estável e sem perdas. No entanto, como qualquer ajuste,
precisa de ser balanceado para garantir que o impacto na latência não prejudique a experiência de comunicação.

O **Adaptive Playout Delay** permite que o sistema se ajuste às condições dinâmicas da rede em tempo real. Embora ofereça uma melhor experiência de chamada, especialmente em redes instáveis, também exige um maior processamento e cuidados no ajuste do atraso

O **Forward Error Correction (FEC)** permite **recuperar pacotes perdidos** em VoIP sem a necessidade de retransmissões e evitando o impacto de **latência adicional**. Usando a operação **XOR** para gerar redundâncias, é possível recuperar até um pacote perdido de um grupo de pacotes. É vantajoso em redes instáveis, onde perdas de pacotes são comuns, mas pode haver um trade-off no aumento da largura de banda e na limitação de recuperação quando múltiplos pacotes são perdidos



Neste gráfico, é dividido em múltiplos chunks e no final, apesar da perda de um pacote, os pacotes contem a maior parte do conteúdo ainda.



RTP header

payload type	sequence number	time stamp	Synchronization Source ID	Miscellaneous fields

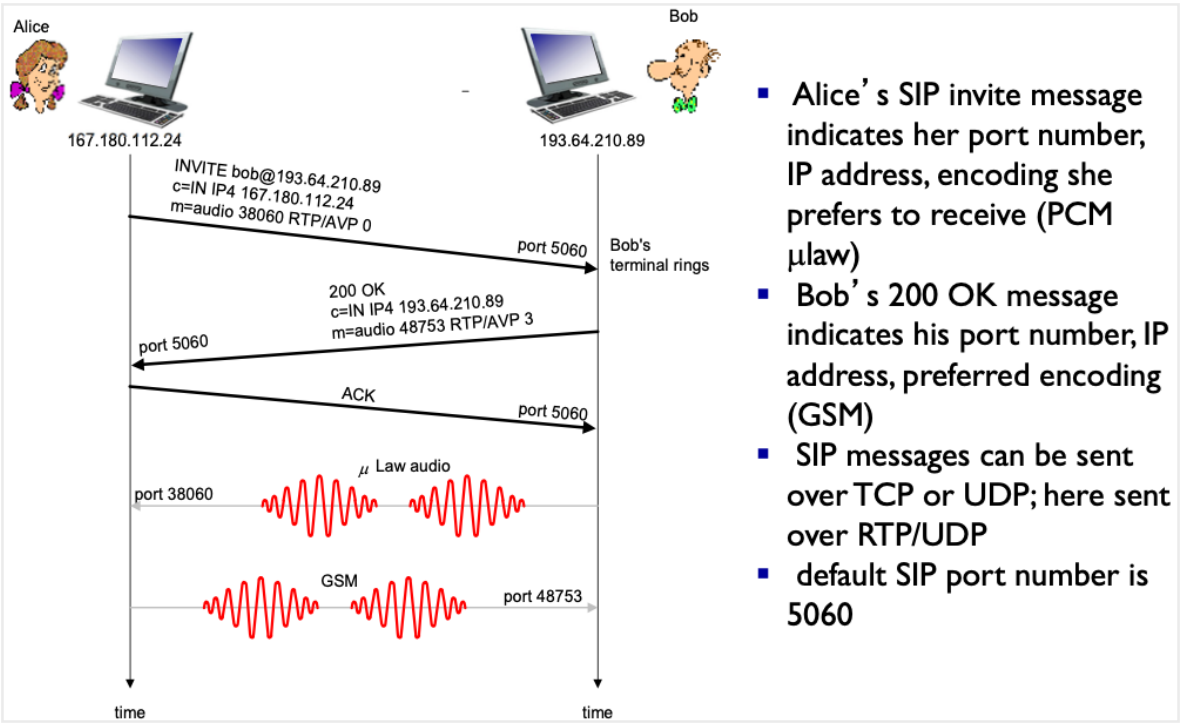
payload type (7 bits) - indica o atual tipo de codificação utilizada
sequence # (16 bits) - numero do pacote enviado
timestamp field (32 bits long) - se a aplicação gera chunks de 160 o timestamp aumenta 160 por cada pacote
SSRC field (32 bits long) - indica a origem da stream RTP

SIP devices

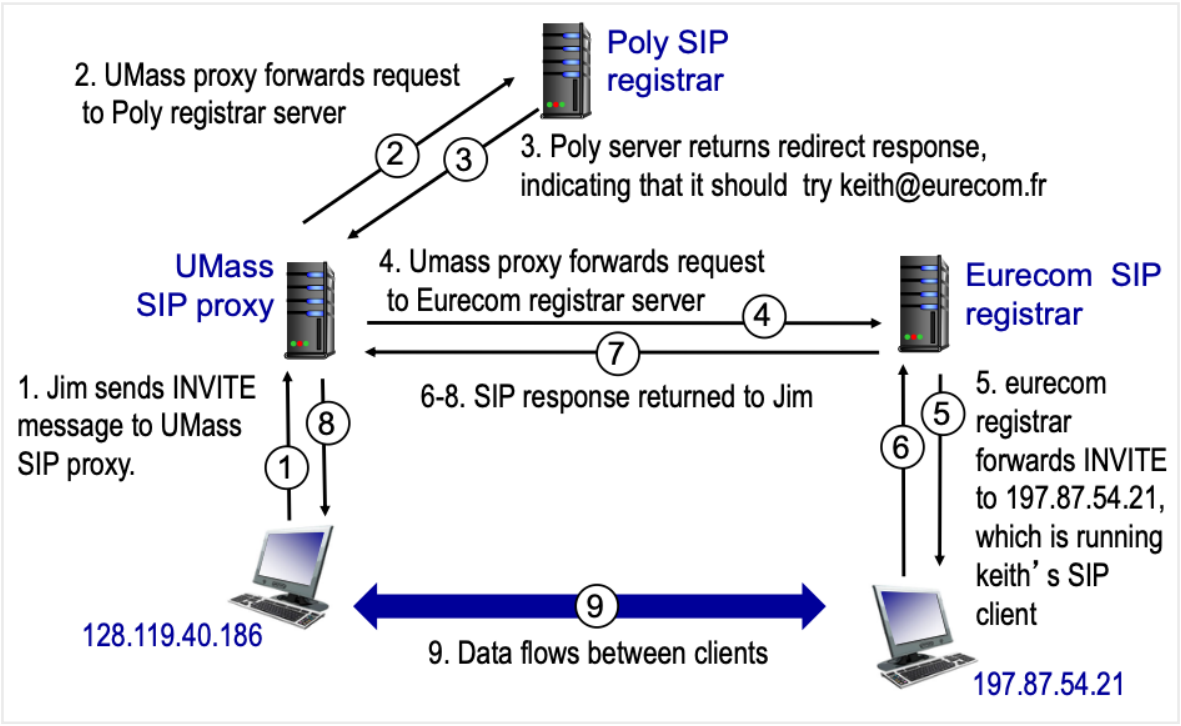
- **Call setup -** Iniciar, negociar e finalizar chamadas entre os

participantes

- **Determinar o IP atual do destinatário (callee) -** Mapeamento de identificadores mnemonicos (ex: user@dominio.com) para o endereço IP atual do callee.
- **Call management -** Adicionar fluxos de mídia, alterar codificação e negociar parâmetros durante a chamada, convidar outros utilizadores, transferir chamadas e colocar em espera



O **SIP Proxy** é essencial para o processo de **encaminhamento e controle** das mensagens SIP, ajudando a estabelecer chamadas, resolver endereços e garantir que a comunicação seja encaminhada corretamente entre os participantes



Cap4 - SIP(Session Initiation Protocol)

Resumo das Funcionalidades do SIP

Funcionalidade	Descrição
Localização do Usuário	Determina o endereço IP atual do usuário para garantir que a comunicação seja roteada corretamente.
Disponibilidade do Usuário	Informa se o usuário está disponível para comunicação e se aceita ou rejeita uma chamada.
Capacidades do Usuário	Negocia quais tipos de mídia e parâmetros (códecs, formatos) serão usados na comunicação.
Configuração da Sessão	Estabelece os parâmetros da sessão, como tipo de mídia, códecs, e configura o processo de toque (ringing).
Gerenciamento de Sessão	Permite modificar parâmetros da chamada, transferir a chamada, colocar em espera e encerrar a sessão.

Integra o **DNS** para localização de utilizadores e usa o **SDP** para negociar os detalhes da comunicação. Além disso, pode ser combinado com vários outros protocolos(RTP, RTSP...) para criar uma arquitetura multimodal e completa para comunicação em tempo real.

Entidades SIP:
Client (Cliente)

- **Função**: O cliente é quem **envia requisições** e **recebe respostas** dentro de uma comunicação SIP.
- Tipos de clientes:
  - User Agent Clients (UAC) - emitem requisições SIP
  - Proxies - reencaminham requisições

Server (Servidor)

- **Função**: O servidor **recebe requisições** e **envia respostas** para os clientes SIP.
- Tipos de servidores:
  - User Agent Servers (UAS) - recebem requisições SIP e respondem
  - Proxy Servers - reencaminham requisições

User Agent (Agente de utilizador)

- **Função**: O **User Agent** está presente em **todos os terminais finais ou estações SIP** (como telefones SIP, softphones, etc.).
- Tipos de User Agents:
  - User Agent Client (UAC)
  - User Agent Server (UAS)

Redirect Server (Servidor de Redirecionamento)

- **Função**: O **Redirect Server** redireciona o cliente para um conjunto alternativo de **URIs**. Ele não roteia as mensagens diretamente, mas **informa ao cliente** onde ele pode encontrar o destino (como uma busca iterativa no DNS).

Proxy Server (Servidor Proxy)

- **Função**: O **Proxy Server** é uma **entidade que atua como servidor e cliente** ao mesmo tempo.

Registrar (Servidor de Registro)

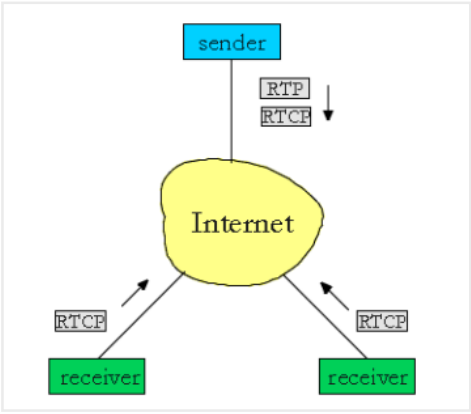
- **Função**: O **Registrar** é um servidor que aceita requisições do tipo **REGISTER** e mantém informações sobre a **localização** dos utilizadores SIP.

Location Service (Serviço de Localização)

- **Função**: O **Location Service** é uma **entidade utilizada por servidores como o Proxy e Redirect Server** para obter informações sobre a **localização do callee** (destinatário da chamada).

Cap5 - Real-time Support RTP/RTCP

O Real-Time Protocol (RTP) visa apoiar o transporte de dados, áudio e vídeo em tempo real.
Suporta sessões unicast e multicast



Cap6 - Stream Control Transmission Protocol (SCTP)

O que é Head-of-Line Blocking: No **TCP**, todos os pacotes são processados na ordem em que foram recebidos. Se um pacote em uma determinada sequência de transmissão for perdido ou corrompido, **todos os pacotes subsequentes** precisam esperar até que o pacote perdido seja retransmitido e recebido corretamente. Esse fenómeno é conhecido como **head-of-line blocking**.

O que é Multihoming: O conceito de **multihoming** se refere à capacidade de uma máquina ou dispositivo estar conectado a múltiplas redes ou interfaces ao mesmo tempo, permitindo que a comunicação continue mesmo se uma das conexões falhar. Por exemplo, se uma máquina tiver duas conexões de rede, ela pode usar uma delas enquanto a outra estiver inativa ou falhando.

Conclusão: Por que o SCTP foi Necessário?
Embora o **TCP** e o **UDP** fossem amplamente utilizados na época, eles apresentavam limitações significativas em determinadas aplicações de **controle de chamadas** e **sinalização**. Essas limitações, especificamente o **Head-of-Line Blocking** e o **Multihoming**, criavam **atrasos inaceitáveis** e **falhas na continuidade do serviço**, o que era problemático em sistemas críticos, como **videoconferências**

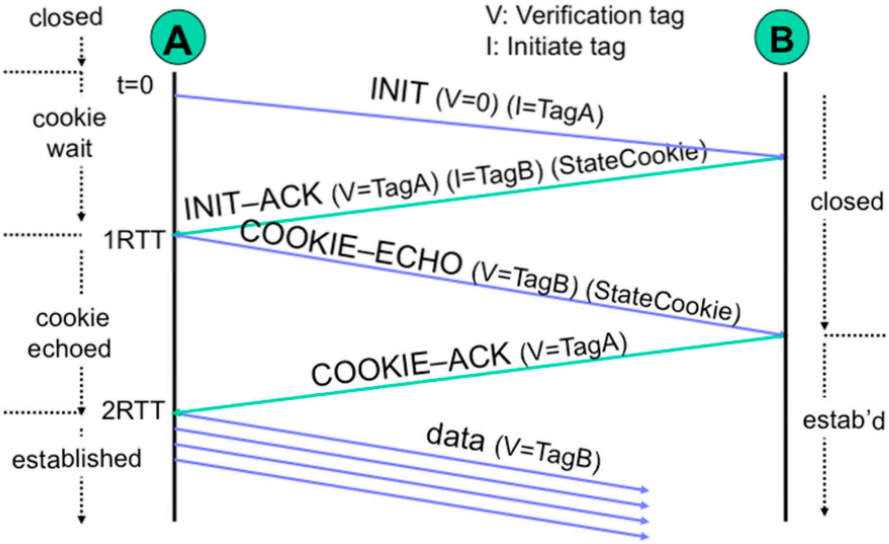
O **SCTP** foi desenvolvido para resolver esses problemas, oferecendo:

1. **Solução para o Head-of-Line Blocking**: Permite múltiplos fluxos de dados independentes dentro de uma única conexão.
2. **Suporte a Multihoming**: Permite que uma única conexão utilize várias interfaces de rede, garantindo continuidade do serviço mesmo se uma delas falhar. (Para redundância e não pelo balanceamento !!!)

Quando comparado ao TCP, o SCTP visa melhorar desempenho e robustez do transporte

endpoint = [10.1.4.2, 10.1.5.3 : 80] - pode ter mais que um endereço IP mas terá sempre uma só porta
association = { [10.1.61.11: 2223], [10.1.4.2,10.1.5.3 : 80]} - relação entre mais que um endpoint

SCTP – 4-way handshake



- Host A sends INIT chunk to Host B, Host B returns INIT-ACK containing a cookie w/ state information that only Host B can verify; 32 bits random verification tags; no memory allocated
- Host A replies with COOKIE-ECHO chunk; may contain A's first data.
- Host B checks validity of cookie; connection is established

Services/Features	SCTP	TCP	UDP
Full-duplex data transmission	yes	yes	yes
Connection-oriented	yes	yes	no
Reliable data transfer	yes	yes	no
Partially reliable data transfer	optional	yes	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	yes
Flow and Congestion Control	yes	yes	no
ECN support	yes	yes	no
Selective acks	yes	optional	no
Preservation of message boundaries	yes	no	yes
Path MTU Discovery	yes	yes	no
Application data fragmentation	yes	yes	no
Multistreaming	yes	no	no
Multihoming	yes	no	no
Protection against SYN flooding attack	yes	no	n/a
Half-closed connections	no	yes	n/a

O **QUIC** (Quick UDP Internet Connections) é um protocolo de transporte de rede desenvolvido pelo **Google**, projetado para melhorar o desempenho da internet, especialmente para aplicações que exigem baixa latência, como

navegação na web, streaming de vídeo e comunicação em tempo real.