Redes de Computadores II
Aulas 10-11
O Protocolo IP

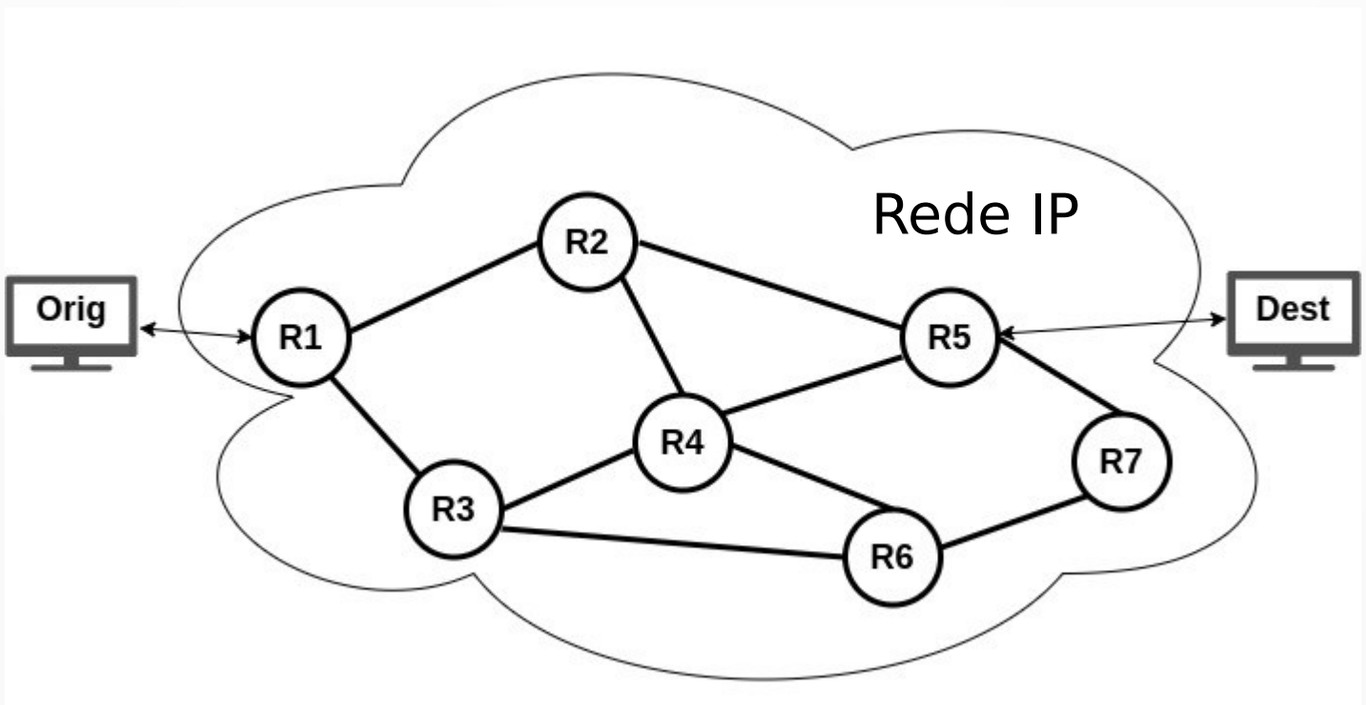
Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/redes

Sumário

- Nestas duas aulas vamos estudar o protocolo IPv4
- Vamos examinar os campos de controle do header IPv4: assim compreendemos o que faz!
- Antes: vamos recapitular o tipo de serviço da “rede IP”: melhor esforço ou *best-effort*
- E também vamos recapitular a arquitetura da Internet: confiabilidade nas pontas

A Internet: Arquitetura

- Infraestrutura da Internet: Rede IP
- Rede IP: coleção de roteadores conectados por enlaces das mais diversas tecnologias



Arquitetura da Internet

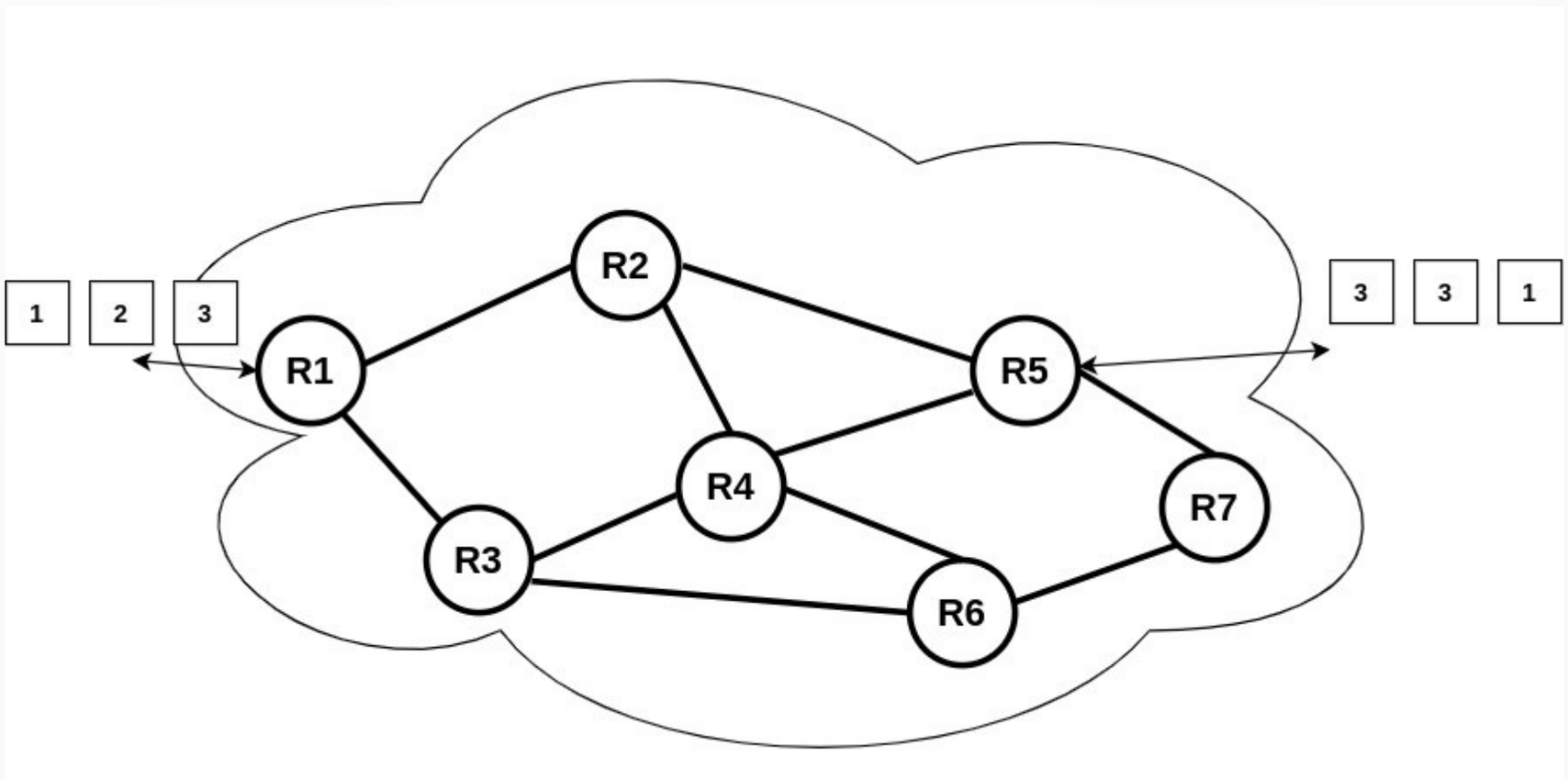
- A rede IP é não-confiável, não orientada à conexão
- O que isso significa?

Arquitetura da Internet

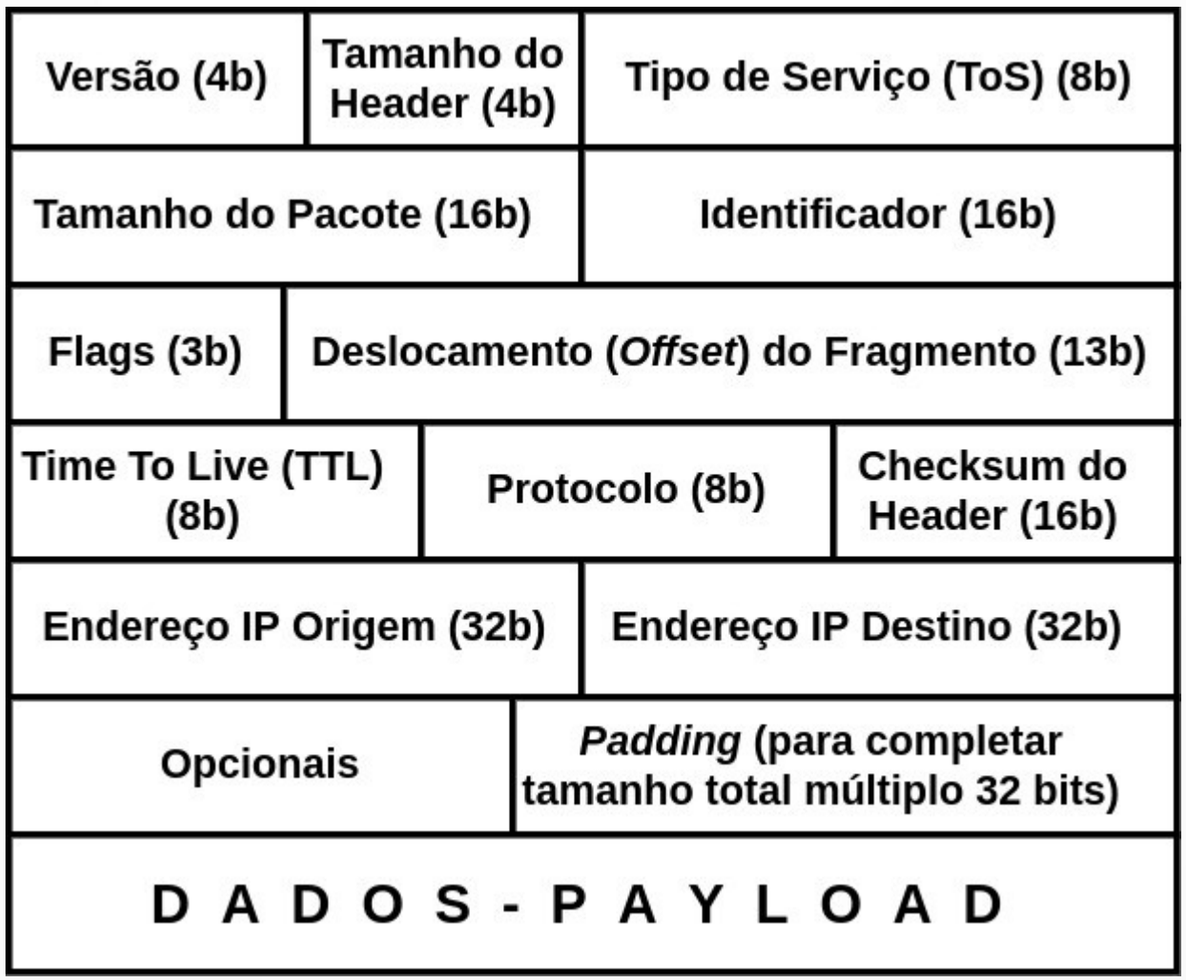
- A rede IP é não-confiável, não orientada à conexão
  - Serviço do MELHOR ESFORÇO (*BEST EFFORT SERVICE*)
- O que isso significa?
- Pacotes podem se perder na rede, ou chegar fora de ordem ou duplicados
- Não há circuitos virtuais, cada roteador trata cada pacote de forma independente
  - dos demais pacotes e dos demais roteadores

Confiabilidade nas Pontas

- Ou o protocolo de transporte (p.ex. TCP) ou o protocolo da camada de aplicação devem garantir que os pacotes chegam e são entregues em ordem



O Datagrama IPv4 - RFC 791



O Campo Versão

- O primeiro campo do header: versão do protocolo
- Ótima ideia! Por que?

O Campo Versão

- O primeiro campo do header: versão do protocolo
- Ótima ideia! Por que?
- Permite a co-existência de versões distintas do protocolo – bem como está acontecendo hoje
- O processamento do pacote IP começa pelo campo versão
- Os demais campos são distintos nas duas versões, o 1º campo idêntico

Campo: #Bytes Header

- Se precisamos informar quantos bytes tem o header do protocolo IP, isso é por que...

Campo: #Bytes Header

- Se precisamos informar quantos bytes tem o header do protocolo IP, isso é por que...
- O tamanho do header é variável
- Motivo: opcionais IP (vamos ver exemplos a frente)
- Entretanto: hoje (na verdade já há um bom tempo) os roteadores comerciais não consideram os opcionais
  - Pior! pode esperar o descarte de pacote com opcinais

Campo: #Bytes Header

- O campo conta o número de palavras de 32 do header (grupos de 4 bytes)
- Quase sempre têm 20 bytes, caso sem opcionais
- Neste caso o campo carrega o valor: 5

Campo: Tipo de Serviço

- ToS: Type of Service
- Originalmente este campo tinha o seguinte formato:

Prioridade (3bits)	D (1 bit)	T (1 bit)	R (1 bit)	Reservados Futuro (2 bits)

O Campo ToS Original

- A utopia da comunidade global perfeita
- Permite definir o tipo de serviço esperado para o pacote
- Por exemplo: 3 bits de prioridade → 8 classes
- Qual classe você vai usar para seus pacotes?

O Campo ToS Original

- A utopia da comunidade global perfeita
- Permite definir o tipo de serviço esperado para o pacote
- Por exemplo: 3 bits de prioridade → 8 classes
- Qual classe você vai usar para seus pacotes?
- Virtualmente todos: a máxima prioridade 
- Hoje roteadores simplesmente ignoram este campo

Os Flags DTR

- Os flags indicam necessidade do payload em termos de
  - D: Delay (Atraso)
  - T: Throughput (Vazão – taxa efetiva de transmissão em bits por segundo)
  - R: Reliability (Confiabilidade)

Nos Anos 1990: ToS Revisitado

- Anos 1990: saltos de ordens de magnitude de capacidade dos equipamentos de redes e telecomunicações
- Na época tentou-se emplacar o termo: “Internet2”
- Mais importante: vinha por aí a comunicação multimídia, inclusive em tempo real :-)
- Na comunidade de pesquisa internacional na área: foco em QoS - *Quality of Service*, Qualidade de Serviço

Qualidade de Serviço

- Algumas aplicações têm demandas específicas, por exemplo voz sobre IP (VoIP)
- Exemplo: se um pacote demora mais de 50ms para chegar ao destinatário, melhor descartar
- Como fazer para garantir a mesma qualidade da rede telefônica na Internet
- Problema: a rede telefônica garante a qualidade por ser confiável e orientada à conexão

QoS: Qualidade de Serviço

- Duas arquiteturas concorrentes foram propostas no IETF (com emoção!)
  - IntServ
  - DiffServ
- A arquitetura DiffServ era baseada em uma re-interpretação do campo ToS:

DCSP (6 bits) *indica do código do tipo de serviço esperado*	**Reservado Futuro (2bits)**

QoS Vs. *Overprovisioning*

- No final das contas: nenhuma das arquiteturas emplacou
- Na prática começou a ser usado o chamado *overprovisioning*
- Os provedores instalam uma capacidade nas suas redes maior que a que demanda dos consumidores
- Outro fator: os consumidores passaram a aceitar “pequenos problemas” com um modelo tarifário mais livre

Próximo Campo IPv4

- Tamanho do Datagrama: número de bytes total do pacote IP
- Temos 16 bits para indicar o tamanho, portanto podemos ter pacotes IP de até __ bytes

Próximo Campo IPv4

- Tamanho do Datagrama: número de bytes total do pacote IP
- Temos 16 bits para indicar o tamanho, portanto podemos ter pacotes IP de até 64K bytes
- Mas... e o limite da rede física?
- Quem lembra o máximo de dados que conseguimos transmitir em 1 quadro Ethernet?

Próximo Campo IPv4

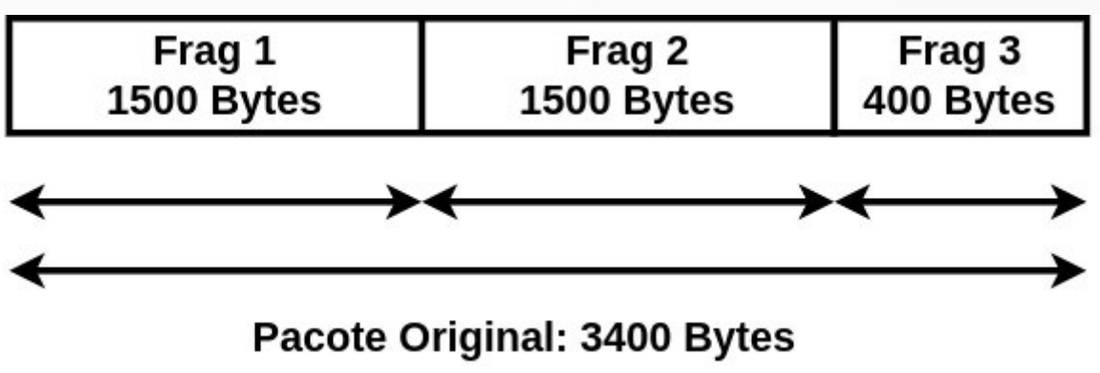
- Tamanho do Datagrama: número de bytes total do pacote IP
- Temos 16 bits para indicar o tamanho, portanto podemos ter pacotes IP de até 64K bytes
- Mas... e o limite da rede física?
- Quem lembra o máximo de dados que conseguimos transmitir em 1 quadro Ethernet?
- 1500 bytes → é o MTU da Ethernet

MTU: Maximum Transfer Unit

- Toda rede física especifica em seus padrões o número máximo de bytes de dados que consegue transmitir em 1 quadro
- Sigla muito importante, usada popularmente
- MTU, “o MTU da Ethernet é 1500 bytes”
- Como fazer quando um pacote IP não cabe no quadro?
- Em outras palavras: quando o tamanho do pacote IP é maior que o MTU do enlace onde será transmitido?

Fragmentação IP

- Um único pacote IP é “quebrado” em múltiplos fragmentos que são transmitidos independentemente
- Cada fragmento é um pacote IP!
- Vamos entender a estratégia do IP através de um exemplo
- Considere um pacote IP de 3400 bytes e uma rede física com MTU = 1500 bytes

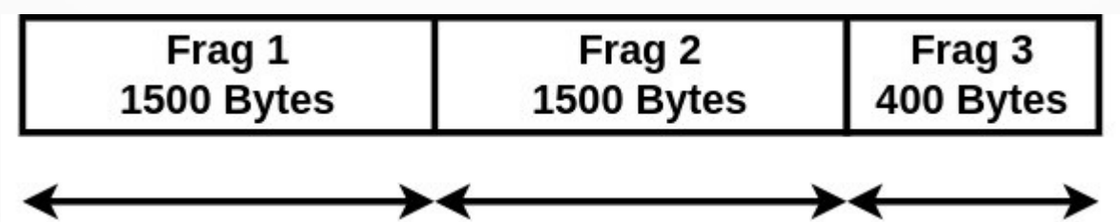


Campos para Fragmentação

- Identificador IP: 16 bits
- Este é um campo do header IP que se mantém em todos os fragmentos
- Dos 3 *flags* dois são usados no contexto de fragmentos:
  - More: um (1) em todos os fragmentos menos no último, no qual é zero (0)
  - *Don't Fragment*: em certos casos é proibido fragmentar, por exemplo implementações mínimas do IP

Campos para a Fragmentação

- Deslocamento ou *offset*: indica a distância dos *dados do fragmento* com relação ao início do *pacote original*
  - *em palavras de 8 bytes*
- Voltando ao exemplo: como fragmentar um pacote IP de 3400 bytes em um enlace com MTU 1500?



Campos Fragmentos: Exemplo

- Pacote original 3400 bytes, MTU = 1500 bytes
  - Fragmento 1: Offset = 0, More = 1
  - Fragmento 2: Offset = 1500, More = 1
  - Fragmento 3: Offset = 3000, More = 0

Campos Fragmentos: Exemplo

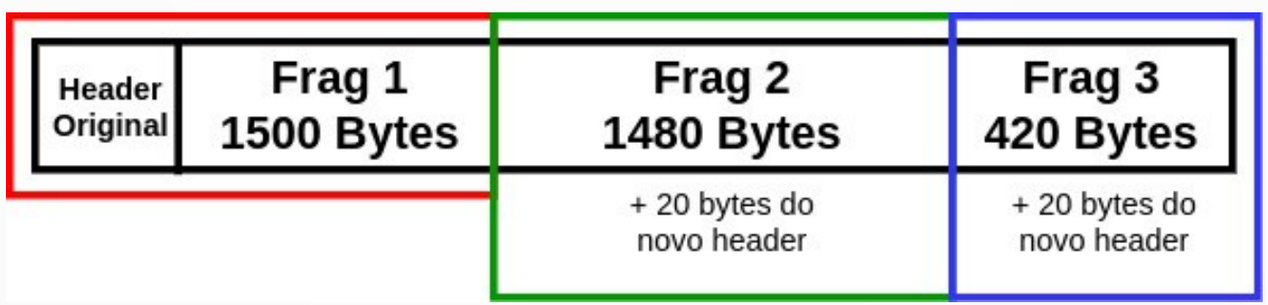
- Pacote original 3400 bytes, MTU = 1500 bytes
  - Fragmento 1: Offset = 0, More = 1
  - Fragmento 2: Offset = 1500, More = 1
  - Fragmento 3: Offset = 3000, More = 0
- É basicamente isso, só que tem que levar em conta os novos headers dos fragmentos
  - Fragmento 1: os 1500 bytes iniciais do pacote original
  - Fragmento 2: os 1480 bytes seguintes + header (20 bytes)
  - Fragmento 3: os 420 bytes finais + header (20 bytes)

Campos Fragemento Exemplo

- Desta forma, no 2º fragmento vão apenas 1480 bytes de dados do pacote original!
- Assim os valores certinhos são:
  - Fragmento 1: Offset = 0, More = 1
  - Fragmento 2: Offset = 1500, More = 1
  - Fragmento 3: Offset = 2980, More = 0

Entendendo os Headers

- O primeiro fragmento já inclui o header como o pacote original
- O segundo fragmento precisa de um header a mais, todos a partir do 2º fragmento



Vamos Fazer um Exercício

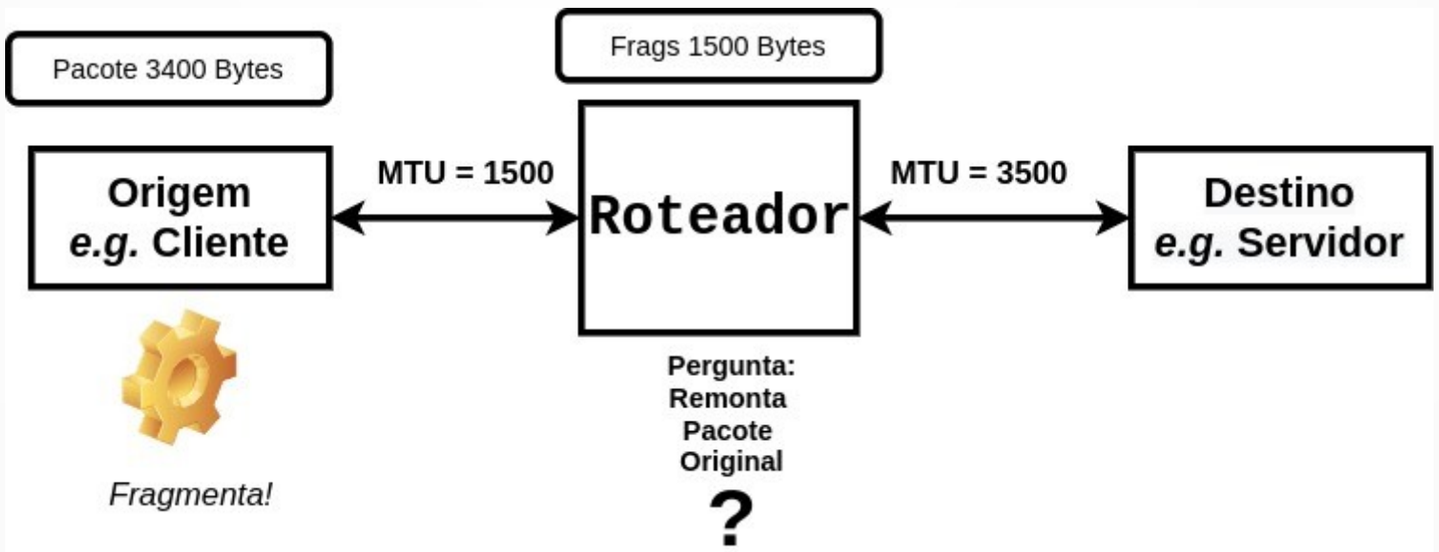
- Mostre como fragmentar um pacote IP de 4500 bytes para transmissão sobre um enlace com MTU = 1500 bytes
  - Fragmento 1: Offset = 0, More = 1 (1500 bytes)
  - Fragmento 2: Offset = 1500, More = 1 (1480 bytes)
  - Fragmento 3: Offset = 2980, More = 1 (1480 bytes)
  - Fragmento 4: Offset = 4460, More = 0 (40 bytes)

Exercício Fragmentação

- Mostre como um pacote de tamanho 2800 bytes é fragmentado para transmissão em um enlace com MTU = 600 bytes

Onde o Pacote Original é Remontado?

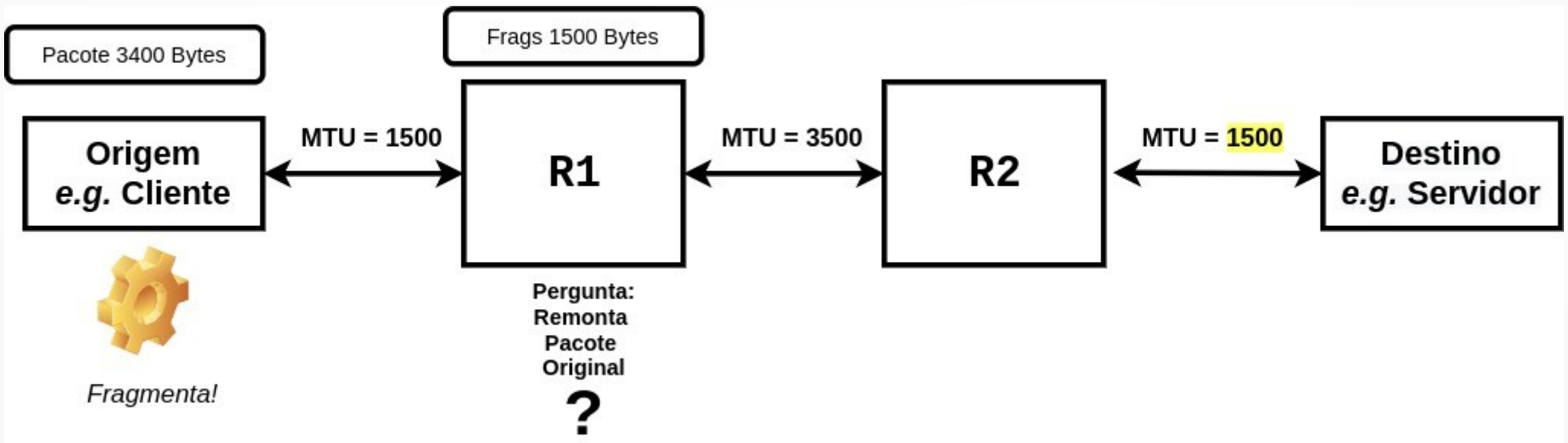
- Considere o seguinte exemplo:



- Quem remonta o pacote original?
- No caso acima há vantagens do roteador intermediário já remontar!

Só o Destino Final Remonta o Pacote

- Mas há muitas desvantagens, considere por exemplo:



- Se o roteador R1 fizesse a remontagem, R2 teria que fragmentar de novo! 😱

Se Precisar: Pode Fragmentar de Novo!

- A estratégia de fragmentação do IP permite fragmentar fragmentos, quantas vezes for necessário!
- Vamos usar o 1º exemplo: um pacote de 3400 bytes é quebrado em fragmentos de 1500 bytes:
  - Fragmento 1: Offset = 0, More = 1 (1500 bytes do pacote orig)
  - Fragmento 2: Offset = 1500, More = 1 (1480 bytes + 20 hder)
  - Fragmento 3: Offset = 2980, More = 0 (420 bytes + 20 hder)
- Agora eles vão ter que passar por um enlace com MTU = 1000 bytes

Exemplo: Fragmentando Fragmentos

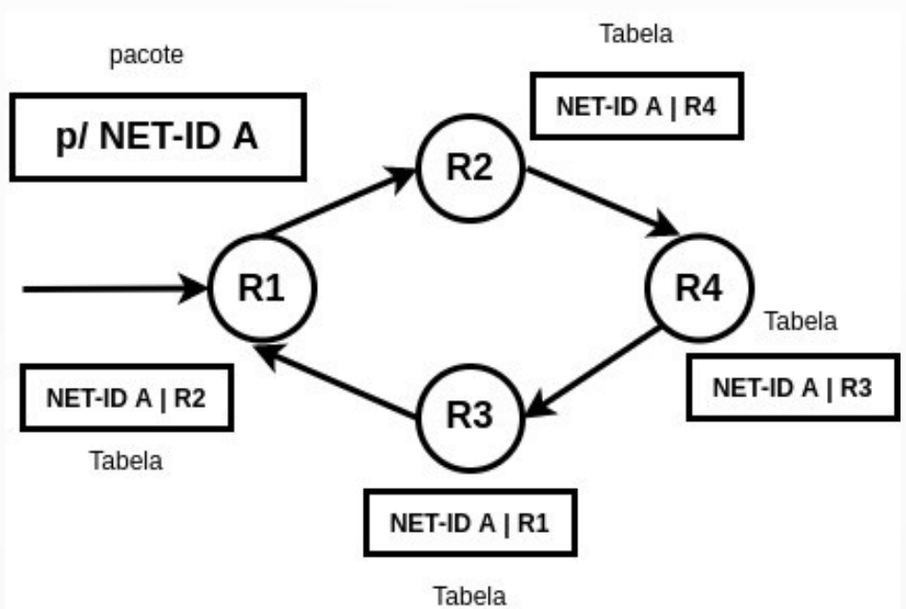
- Cada fragmento vai ser refragmentado:
  - Fragmento 1: Offset = 0, More = 1 (1000 bytes)
  - Fragmento 2: Offset = 1000, More = 1 (500 + 20 bytes)
  - Fragmento 3: Offset = 1500, More = 1 (1000 bytes)
  - Fragmento 4: Offset = 2440, More = 1 (500 + 20 bytes)
  - Fragmento 5: Offset = 2980, More = 0 (420 bytes)

Frag: Para Deixar Bem Claro

- Quem faz a fragmentação (e depois a remontagem do pacote original) é próprio protocolo IP
- A toda transmissão no enlace o IP verifica se o tamanho do pacote > MTU do enlace
- A fragmentação pode ser feita tanto no host origem como em um roteador intermediário
- A fragmentação deve ser evitada tanto quanto possível: degrada o desempenho do IP e introduz vulnerabilidade a falhas
  - Perdeu 1 fragmento, perdeu tudo

Time To Live: TTL

- Já falamos que os roteadores tomam decisões independentes: para cada pacote, com respeito às decisões de outros roteadores
- Assim, loops podem acontecer



O Campo TTL

- O campo TTL especifica quanto tempo um pacote IP pode permanecer na rede
- Se não fizer nada com os pacotes que estão em loop eles permanecem para...

O Campo TTL

- O campo TTL especifica quanto tempo um pacote IP pode permanecer na rede
- Se não fizer nada com os pacotes que estão em loop eles permanecem para... sempre!
- O nome deste campo não é bom... tempo em redes e sistemas distribuídos, um conceito que demanda cuidado → dificuldade de sincronização
- TTL não conta tempo

O Campo TTL

- O campo TTL conta o número de *hops* (saltos) dados pelo pacote na rede
- Em outras palavras: o número de roteadores pelos quais passou:

```
Inicialmente TTL ← max_hops (definido pela implementação do IP (S.O.) valores comuns 64 ou 128;

Cada roteador que processa o pacote faz TTL ← TTL –1;

SE TTL = 0 ENTÃO descarta o pacote;
```

O Campo Protocolo

- Este campo indica qual protocolo está usando o IP para comunicar
- O protocolo IP não gera pacotes “espontaneamente”
- Em geral: TCP, UDP, ICMP, outros...
- Às vezes IP encapsulando IP
  - backbones virtuais

- Os identificadores de protocolo são definidos pela IANA: *Internet Authority for Number Assignment*

O Checksum do Protocolo IP

- Muito mais simples e frágil que o checksum da Ethernet
- Por que?

O Checksum do Protocolo IP

- Muito mais simples e frágil que o checksum da Ethernet
- Por que?
- Ethernet: implementada em hardware &
  IP implementado em software
- Facinho de enganar 🤭
- Além disso: calculado apenas sobre o header, não sobre o *payload*

O Checksum IP

- Um código de detecção de erros
- A origem: calcula o checksum e armazena no campo checksum
- O destino: recalcula o checksum, se deu problema é porque tem erro no header
- Erro neste contexto: um bit transmitido como 1 é recebido como 0 e vice-versa

Checksum IP: O Algoritmo

- O algoritmo do checksum IP trata o header como uma sequência de palavras de 16 bits
- Soma as palavras usando complemento de 1
- A soma complemento de 1: igual a soma binária, se no final der “vai 1”, então soma +1
- Depois da soma: tira o complemento do resultado
- Destinatário: faz o mesmo cálculo incluindo o checksum
  - se deu 0 (zero) então tudo OK senão erro/descarta!

Exemplo Cálculo Checksum IP

- Vamos calcular o checksum IP para o caracter ASCII ‘a’: 0110 0001
- Aqui no exemplo vamos considerar palavras de 4 bits ao invés de 16 bits
- A origem soma 0110 + 0001 = 0111
- Tira o complemento: 0111 → 1000 (é o checksum)
- Transmite o dado com o checksum:

  0110 0001 1000

O Destinatário

- No destino o mesmo cálculo é feito somando inclusive o checksum
- Soma 0110 + 0001 + 1000 = 1111
- Tira o complemento: 0000 → tudo certo! sem erro!
- Vamos fazer outro exemplo com “vai 1”: transmita 1101 0101
- Fazer também um exemplo em que troca um bit
- Pense como enganar o algoritmo ;-)

Checksum da Internet

- O mesmo algoritmo que estudamos hoje para o IP é usado por diversos outros protocolos
- ICMP, UDP, TCP, ...

Opcionais IP

- Antes de vermos, lembre-se: quase sempre desabilitados nos roteadores da Internet
  - Antes opcionais ignorados → hoje pacotes descartados!
  - Segurança é problema #1 hoje
- Apenas em redes privadas e contextos específicos: por exemplo em um laboratório
- Para discussão: TCP/IP é mais que Internet!
  - Construção de redes internas de organizações, muitas nem conectadas à Internet

Alguns Opcionais Importantes

- *Source Route* – termo muito importante em redes de computadores! “Source routing” é roteamento na origem
  - O pacote sai da origem com a rota completa que deve percorrer
  - Pode ser *strict* (ordem exata) ou não (*loose*)
- *Record Route* – determina que os roteadores devem ir gravando no pacote a rota que está percorrendo

Mais Opcionais

- Timestamp: os roteadores que processam o pacote gravam não apenas seus endereços, mas também o instante de tempo do processamento
  - O tempo é especificado com precisão de milissegundos, usando hora UTC
  - UTC: *Universal Coordinated Time*
- Além disso: opcionais de segurança

Padding

- A melhor tradução de *padding* para português é “enchimento”
- O tamanho total do header tem que ser múltiplo de 32
- Como os opcionais podem ter tamanho arbitrário...
- ... o *padding* é usado para garantir que o tamanho do header múltiplo de 32
- Basta acrescentar bits com valor zero (0)

Conclusão

- Nestas aulas estudamos o protocolo IP
- Iniciamos definindo “rede IP” e recapitulando o tipo de serviço que oferece: *best-effort*
- Arquitetura da Internet
- Campos: Versão, #bytes do header, ToS, Tamanho do Pacote
- Fragmementação: MTU e limites da rede física
- TTL, Protocolo, Checksum, Opcionais

Obrigado!

Lembrando: a página da disciplina é:
https://www.inf.ufpr.br/elias/redes