Aula 3 - Internet: Filosofias de Projeto e Métricas de Desempenho

Diego Passos

Universidade Federal Fluminense

Redes de Computadores

Material adaptado a partir dos slides originais de J.F Kurose and K.W. Ross.

Filosofias e Princípios de Projeto da Internet

Princípios e Filosofias: Motivação

- **Administração descentralizada:**
  - A Internet é, em princípio, um esforço coletivo.
  - Múltiplas entidades **colaboram** para um objetivo maior.
    - Interconexão entre elementos computacionais.
- Mas estas entidades possuem metodologias, objetivos individuais diferentes.
  - Algumas vezes conflitantes.
- Necessidade de **uniformização**.
  - Estabelecimento de princípios básicos seguidos por todos.

Heterogeneidade

- Por **projeto**, a Internet é heterogênea.
  - *e.g.*, enlaces de tecnologias, capacidades diferentes.
  - *e.g.*, ISPs com políticas diferentes.
  - *e.g.*, dispositivos computacionais com arquiteturas diferentes.
- Consequências:
  - São necessários protocolos **comuns a todos os dispositivos**.
    - Uma suíte de protocolos da Internet.
  - Estes protocolos devem ser **flexíveis**.
    - *i.e.*, não devem assumir muitas características da rede.

O Argumento Fim-a-Fim (I)

- Argumento apresentado por David Clark em 1981.
  - Chefe do desenvolvimento da arquitetura de protocolos na Internet entre 1981 e 1989.
- Guiou o desenvolvimento arquitetural da Internet.
  - Embora haja exceções.

Argumento Fim-a-Fim

Uma funcionalidade só pode ser implementada de forma **correta e completa** se isto for feito com o auxílio das aplicações executadas nas pontas do sistema de comunicação.

O Argumento Fim-a-Fim (II)

- Exemplo: transmissão de um arquivo entre dois computadores.
- Hipóteses:
  - Computadores interconectados por uma rede de comunicação.
  - Arquivo transmitido quebrado em uma série de pacotes.
  - Pacotes podem ser entregues pela rede com erros.
  - Pacotes podem ser completamente perdidos e descartados.
- Objetivo:
  - Garantir que arquivo chegue **íntegro** ao destinatário.

O Argumento Fim-a-Fim (III)

- Primeira abordagem: rede “garantirá” a integridade dos dados.
  - Todo roteador/*host*, ao receber um pacote por um enlace, verificará sua integridade.
    - Como?
  - Caso pacote não seja íntegro, roteador/*host* requisitará uma **retransmissão**.
  - A rede também garantirá que pacotes **não cheguem fora de ordem**.
- Pergunta: isto é suficiente?

O Argumento Fim-a-Fim (IV)

- Não, por uma série de motivos.
  - Quem garante que arquivo que saiu do *host* estava originalmente íntegro?
  - Quem garante que verificação de integridade dos pacotes não falhou?
  - Quem garante que as implementações dos roteadores/*hosts* estão corretas?
  - Em resumo: em última instância, receptor **ainda precisa verificar integridade**.
- Além disso, **é desejável que a rede faça este tipo de função?**
  - Adiciona complexidade.
  - Assume certas funcionalidades de **todos os roteadores intermediários**.
    - Mas a rede é heterogênea!
  - Tem **efeitos colaterais indesejados** para certas aplicações.

O Argumento Fim-a-Fim (V)

- **Alternativa:** deixar que sistemas finais lidem com o problema.
  - Façam as verificações.
  - Requisitem retransmissões, se necessário.
    - A partir do *host* de origem!
- Vantagem:
  - Já que as pontas precisam da funcionalidade, esta é mantida **apenas lá**.
  - Não há redundância de implementações.

O Argumento Fim-a-Fim: Desempenho

- Nunca há vantagens em se implementar funcionalidades nas redes?
  - Pode haver, por conta de desempenho.
  - Exemplo: pode ser mais rápido retransmitir pacotes por um único enlace que em um caminho fim-a-fim.
  - Há exemplos destas **otimizações** na Internet.
- Mas como princípio geral, estas são evitadas.

O Argumento Fim-a-Fim: Analogias

- RISC *vs.* CISC.
  - Processador (rede) deve implementar um grande número de funcionalidades?
  - Ou apenas um subconjunto básico, suficiente?
    - Emulando instruções (funcionalidades) mais complexexas em *software* (nos *hosts*).
- Navalha de Occam.
  - Princípio lógico que guia a escolha de explicações para fenômenos.
  - Se há múltiplas explicações plausíveis, opta-se pela mais simples.

O Argumento Fim-a-Fim: Inteligência nas Bordas

- Consequência direta: na Internet, procura-se manter a complexidade nas bordas.
  - “Inteligência nas bordas”.
  - “Núcleo simples”.
- Núcleo processa volume enorme de pacotes.
  - Mantê-lo simples, especializado, **rápido**.
  - Objetivo maior: **escalabilidade**.
- Volume de tráfego nas bordas é menor.
  - Processamentos mais complexos são mais viáveis.
  - Escalabilidade não é tão problemática.
- Oposição às redes telefônicas.
  - Núcleo complexo, bordas simples.

Princípio KISS

- KISS: *Keep It Simple, Stupid!*
- Princípio originado na Marinha Americana, na década de 1960.
  - Incorporado nas filosofias que guiam a Internet.

Princípio KISS

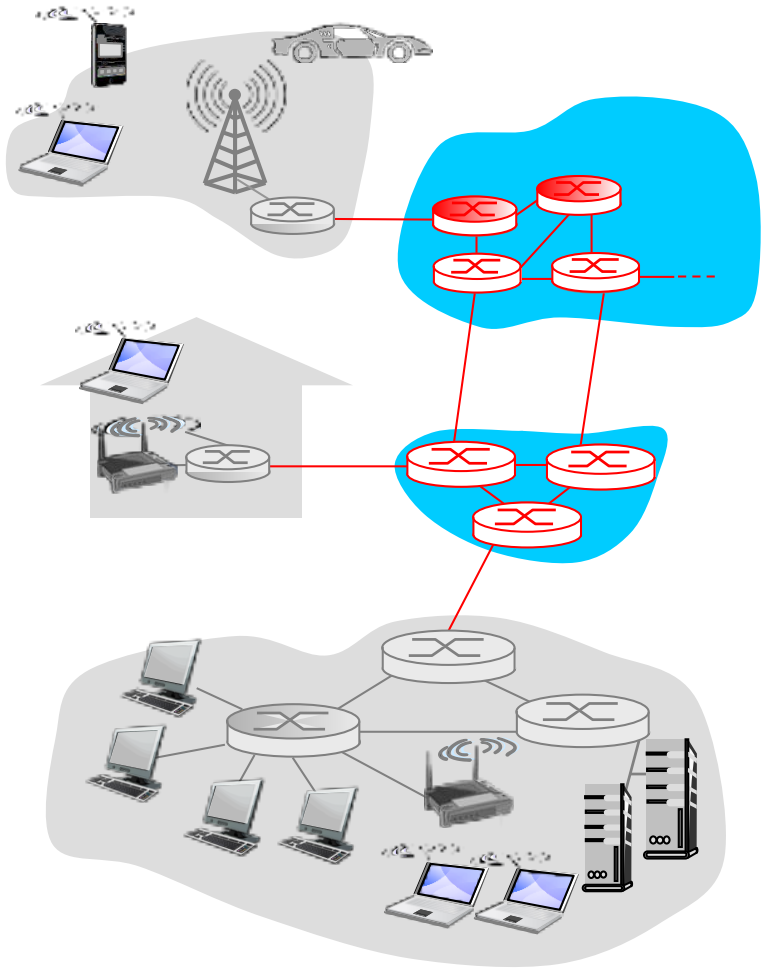
Sistemas funcionam melhor quando são mantidos simples. Logo, simplicidade deve ser um objetivo de projeto, evitando complexidades desnecessárias.

- Aplicado à Internet:
  - Protocolos "simples".
    - Relativamente, falando.
  - Fáceis de implementar, depurar.
    - Relativamente, falando.

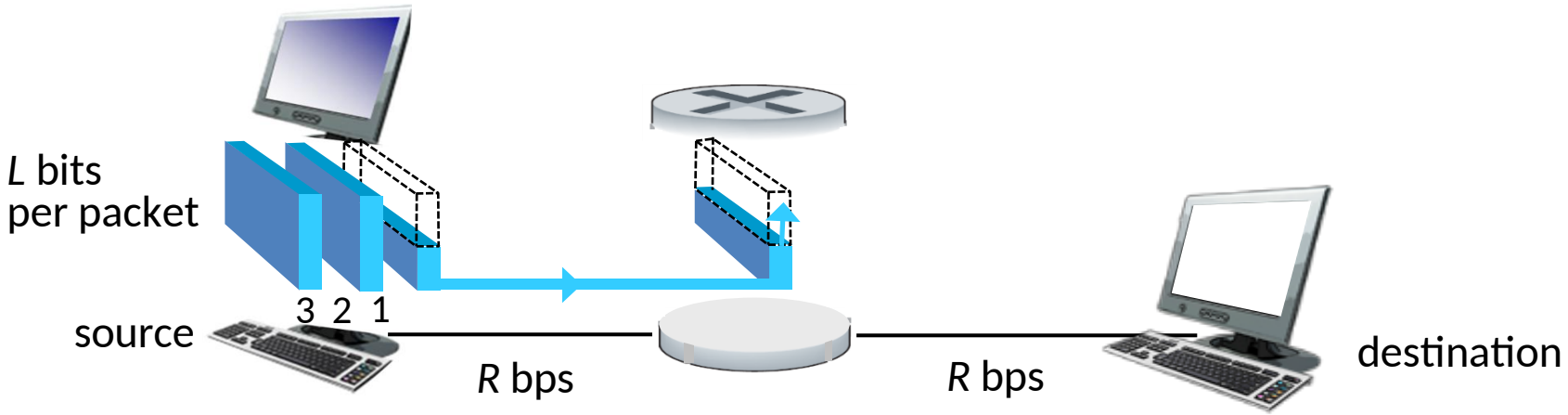
Comutação de Pacotes

Comutação de Pacotes: Conceitos Básicos

- *Hosts* quebram dados em **pacotes**.
  - Pacotes são a unidade fundamental de transferência.
- Pacotes são **encaminhados** por comutadores.
  - Passam por sequência de enlaces, formando um caminho ou **rota**.
- Sempre que pacote é transmitido por enlace, **transmissão utiliza toda a capacidade do enlace**.



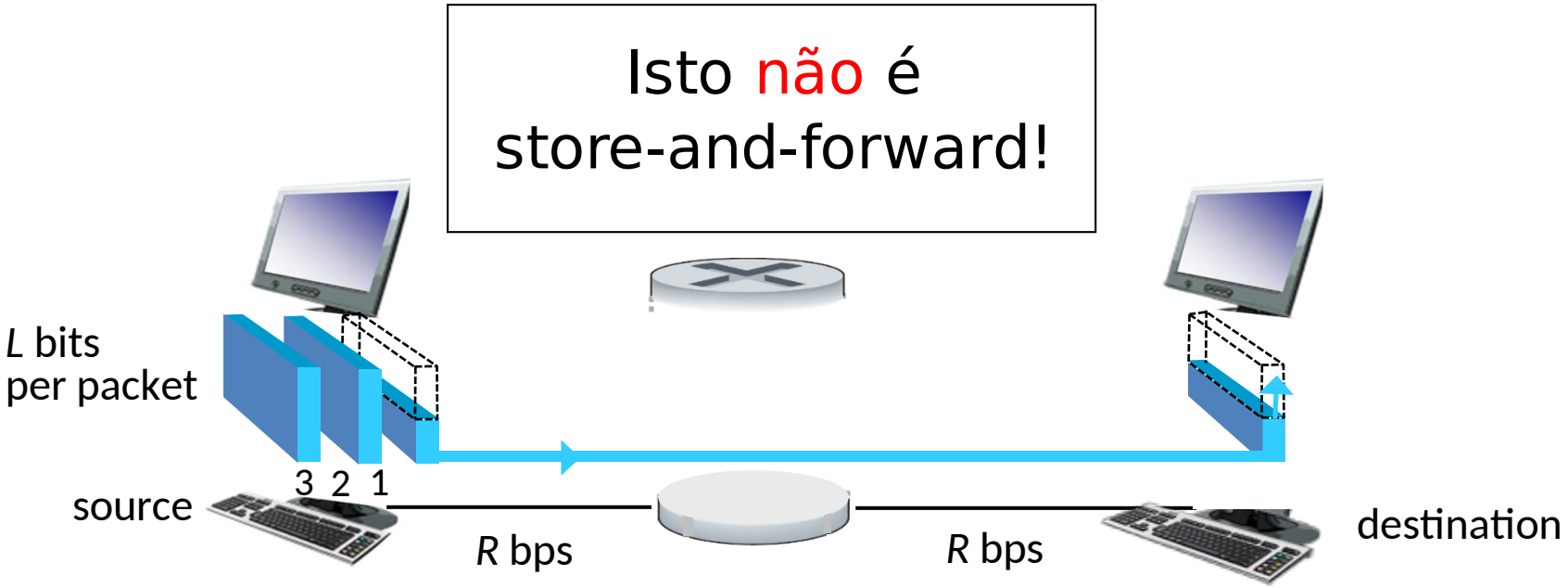
Comutação de Pacotes: *Store-and-Forward* (I)



- São necessários $\frac{L}{R}$ segundos para transmitir ("colocar no enlace") pacote de L bits em enlace de capacidade R.
- **Store-and-Forward:** pacote precisa ser recebido por **completo** antes transmissão começar no próximo enlace.
  - Comutador recebe pacote **inteiro**, armazena em memória, processa, ...
- **Consequência:** tempo total até o destinatário é $\frac{2L}{R}$.
  - Ignorando tempo de propagação.
- Exemplo numérico:
  - $L = 4\ MB$.
  - $R = 2\ Mb/s$.
  - Tempo de transmissão por um enlace: 16 s.
  - Para transmissão **fim-a-fim**: 32 s.

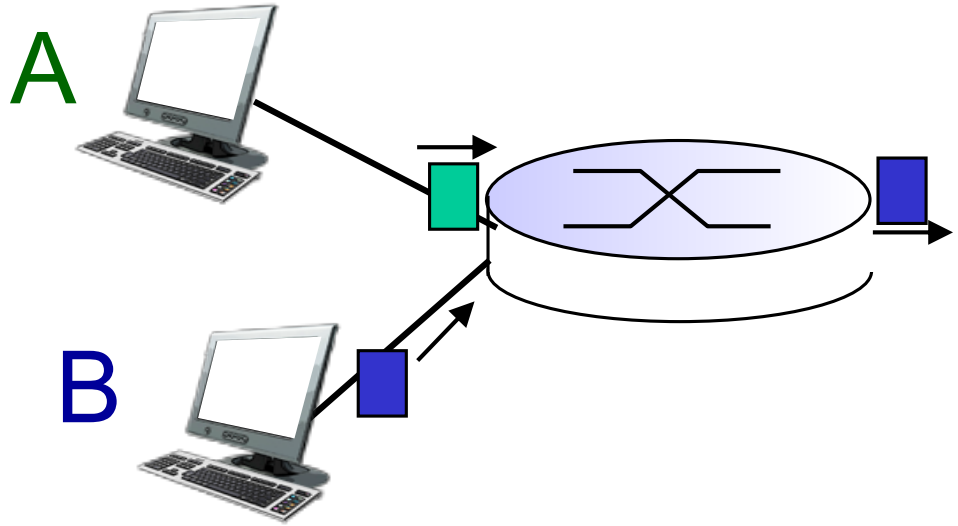
Comutação de Pacotes: *Store-and-Forward* (II)

- Contra-exemplo:

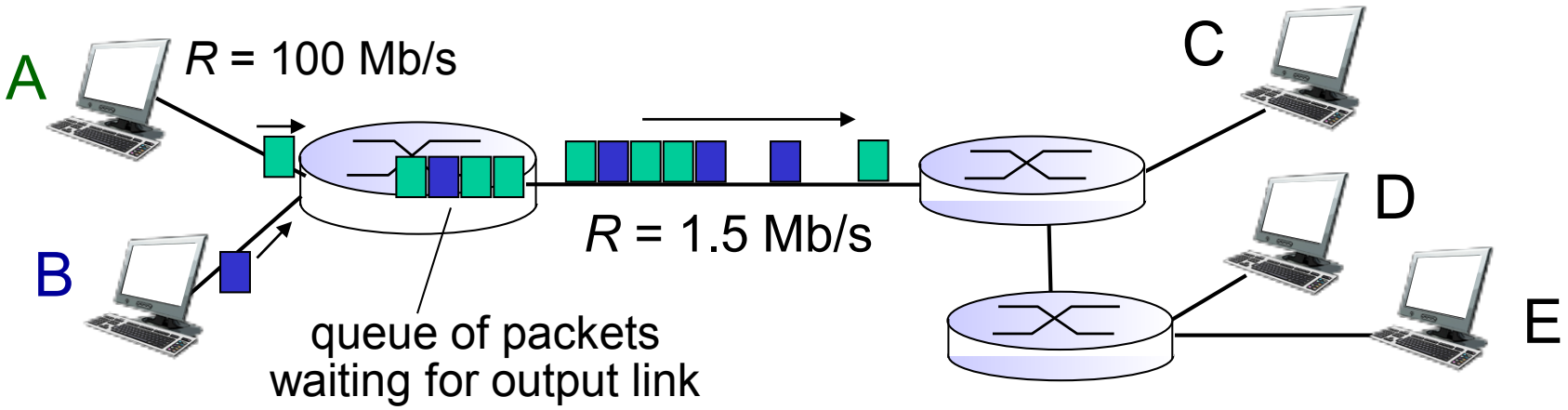


Comutação de Pacotes: *Store-and-Forward* (III)

- Por que comutador não pode transmitir logo que recebe primeiros bits?
- Por que esperar que o pacote chegue completamente?
- Alguns motivos:
  - Processamentos são necessários.
    - Encaminhamento, verificação de integridade, ...
  - Enlace de saída pode estar ocupado com outro pacote.
    - Pacote que chega pode ter que **esperar**.



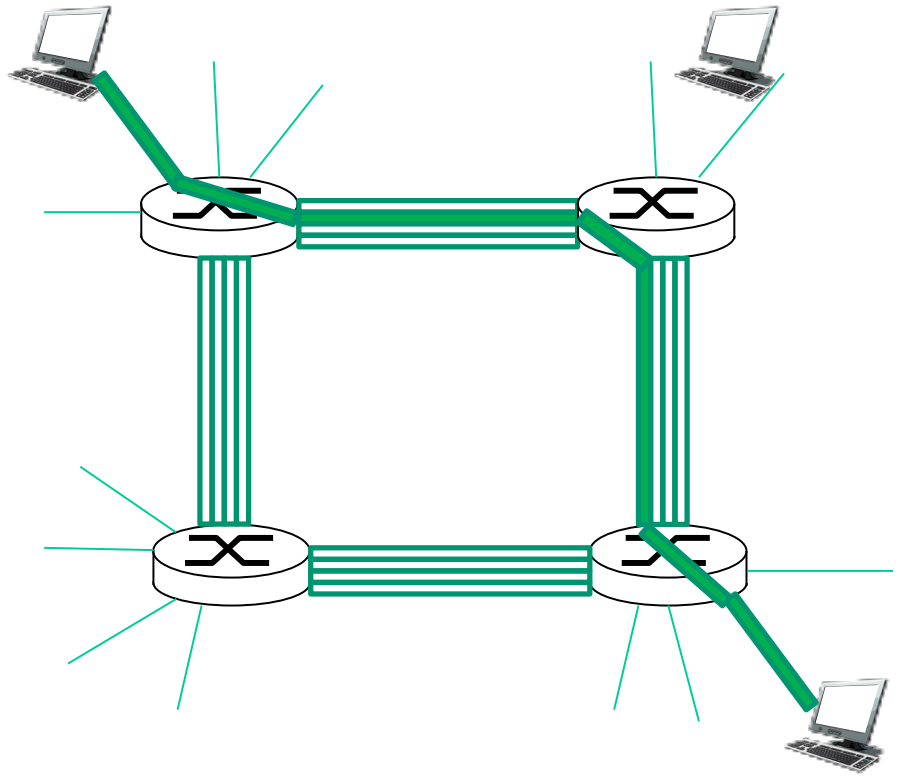
Comutação de Pacotes: Enfileiramento e Descartes



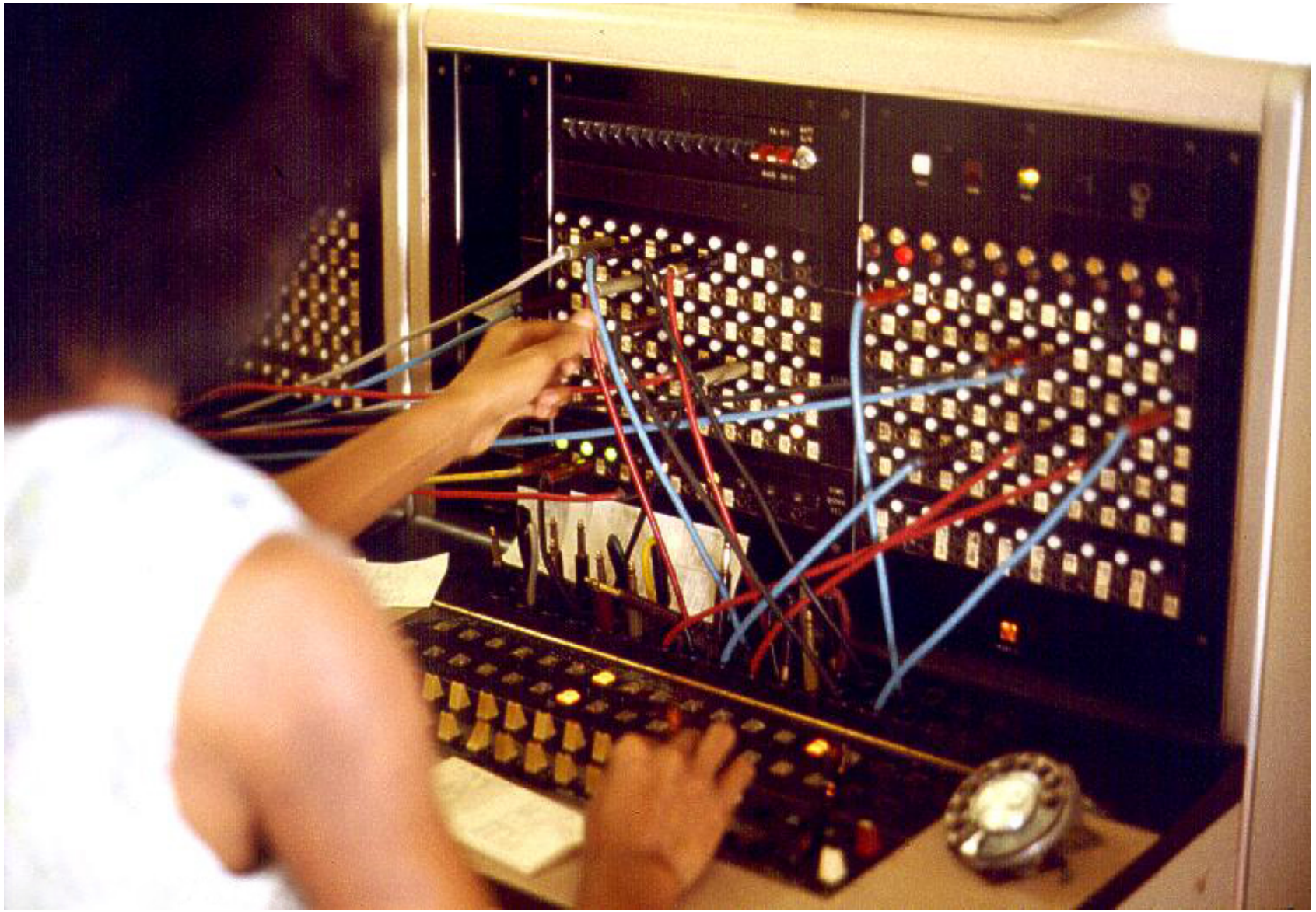
- **Enfileiramento:** pacotes são armazenados em uma **fila**.
- Aguardam oportunidade de transmissão.
- Se a **taxa de chegada** é maior que a capacidade de escoamento, fila tende a crescer.
- Mas o tamanho da fila é limitado (**Por que?**).
  - Quando a capacidade é excedida, pacotes são **descartados**.

Alternativa: Comutação de Circuitos (I)

- Recursos fim-a-fim são **alocados, reservados** para uma "chamada" entre origem e destino.
  - No exemplo, cada "enlace" contém 4 circuitos independentes.
    - Chamada usa segundo circuito do enlace de cima, primeiro circuito do enlace da direita.
  - Recursos são dedicados: uma vez alocados, ninguém mais os utiliza.
    - Desempenho de circuito, garantido.
  - Recursos ficam **ociosos** se não usados pela chamada.
  - Comumente utilizado em redes de telefonia.

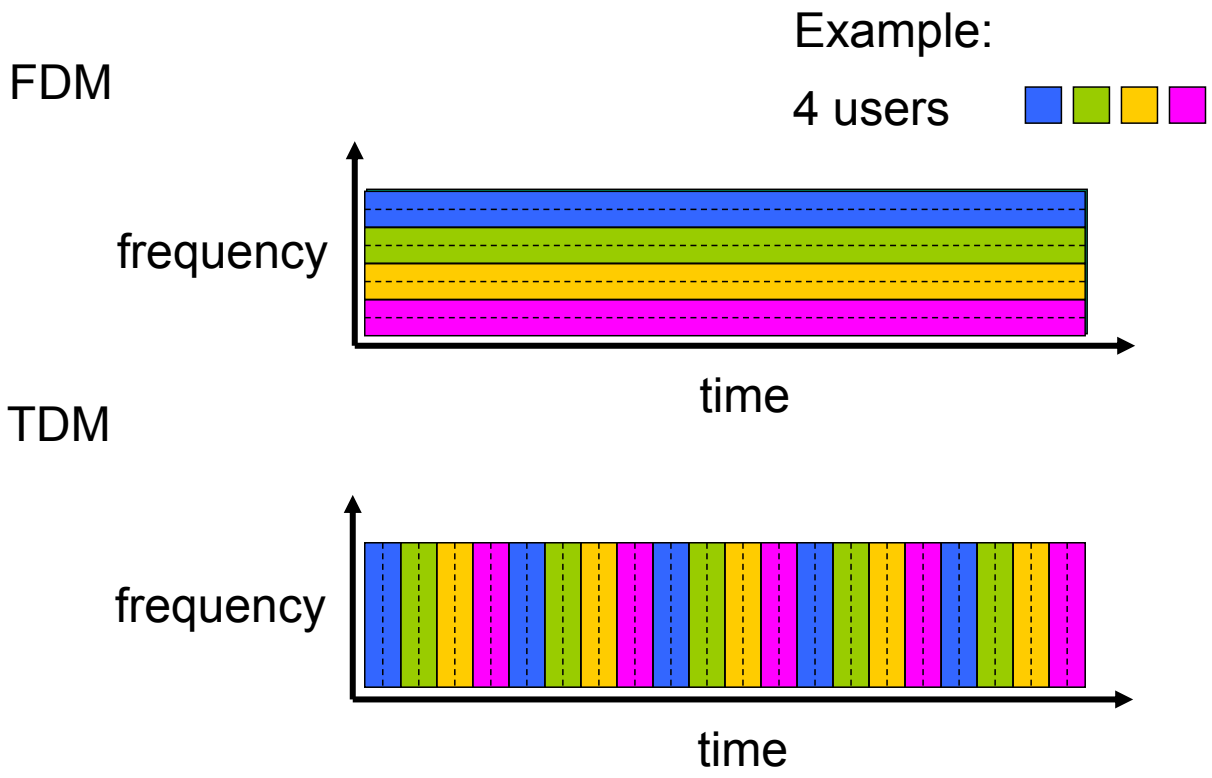


Alternativa: Comutação de Circuitos (II)



Comutação de Circuitos: Multiplexação

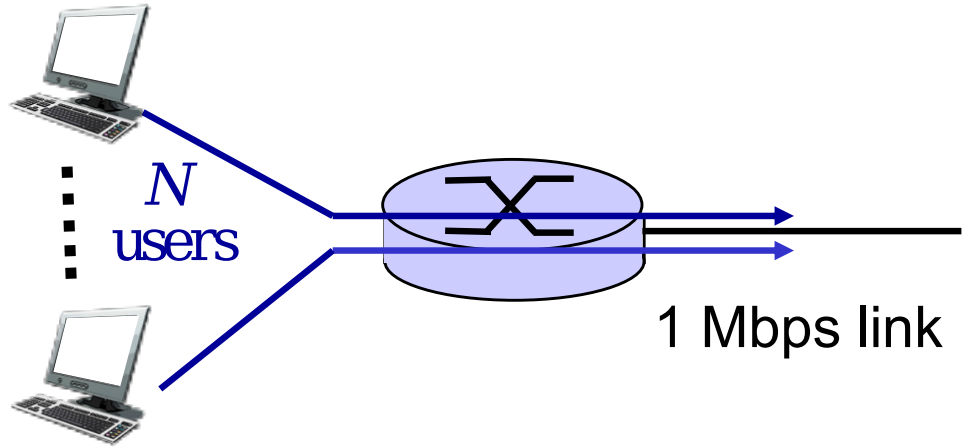
- Não necessariamente precisamos de circuitos fisicamente separados entre cada comutador.
- Chamadas podem ser **multiplexadas** em um mesmo meio físico.
- Dois exemplos comuns:
  - Multiplexação por Divisão no Tempo (TDM).
  - Multiplexação por Divisão na Frequência (FDM).



Comutação de Pacotes: Multiplexação Estatística (I)

- Na comutação de pacotes, também há multiplexação.
  - Multiplexação ocorre "naturalmente".
  - Pacotes de usuários distintos são transmitidos de forma intercalada.
  - Ordem depende da chegada (geralmente aleatória) dos pacotes no comutador.

- **Pergunta:** quantos usuários podem compartilhar um único enlace de saída?
- Exemplo: enlace de saída de 1 Mb/s.
- "Prometemos" 100 kb/s a cada usuário **quando ativos**.
  - Hipótese: cada usuário ativo apenas 10% do tempo.
- **Comutação de circuitos:**
  - Alocações fixas, recursos reservados.
  - No máximo, 10 usuários.



- **Comutação de pacotes:**
  - Recursos ociosos podem ser utilizados por outros usuários.
  - Logo: espera-se poder colocar **mais de 10** usuários!
- Mas qual é o valor exato de N?

Comutação de Pacotes: Multiplexação Estatística (II)

- Enlace de 1 Mb/s, 100 kb/s para cada usuário = no máximo 10 usuários **simultâneos**.
- Digamos que haja $N = 35$ usuários.
  - Qual a **probabilidade** de que mais de 10 estejam ativos simultaneamente?
  - Considerando certas simplificações implícitas.
- Um usuário **qualquer** fica ativo com probabilidade $p = 10\%$.

1 dado usuário ativo (demais inativos)	**2 dados usuários ativos (demais inativos)**	**k dados usuários ativos (demais inativos)**
$p \cdot (1-p)^{34}$	$p^2 \cdot (1-p)^{33}$	$p^k \cdot (1-p)^{(N-k)}$

Comutação de Pacotes: Multiplexação Estatística (III)

- Note que com 35 usuários, há múltiplas combinações de k usuários que podem estar ativos simultaneamente.
  - P/ $k = 1$, 35 combinações.
  - P/ $k = 2$, $\frac{35 \cdot 34}{2} = 595$ combinações.
  - ...
  - P/ k em geral, $\binom{35}{k}$ combinações.

1 usuário ativo (demais inativos)	**2 usuários ativos (demais inativos)**	**k usuários ativos (demais inativos)**
$35 \cdot p \cdot (1-p)^{34}$	$595 \cdot p^2 \cdot (1-p)^{33}$	$\binom{35}{k} \cdot p^k \cdot (1-p)^{(35-k)}$

Comutação de Pacotes: Multiplexação Estatística (IV)

- Voltando ao problema original: probabilidade de mais de 10 ativos.

$$Pr(k > 10) = \sum_{i=11}^{35} \binom{35}{i} \cdot p^i \cdot (1-p)^{(35-i)} \approx 0{,}00042$$

- Resumindo:
  - Mesmo com 35 usuários, entregamos a banda prometida 99,958% do tempo.
  - **Número muito maior de usuários** que na comutação de circuitos.
  - Percentual de falha "aceitável".
- Mas o que acontece se mudarmos os parâmetros?
  - *e.g.*, aumentarmos N, aumentarmos p, ou reduzirmos a capacidade do enlace?

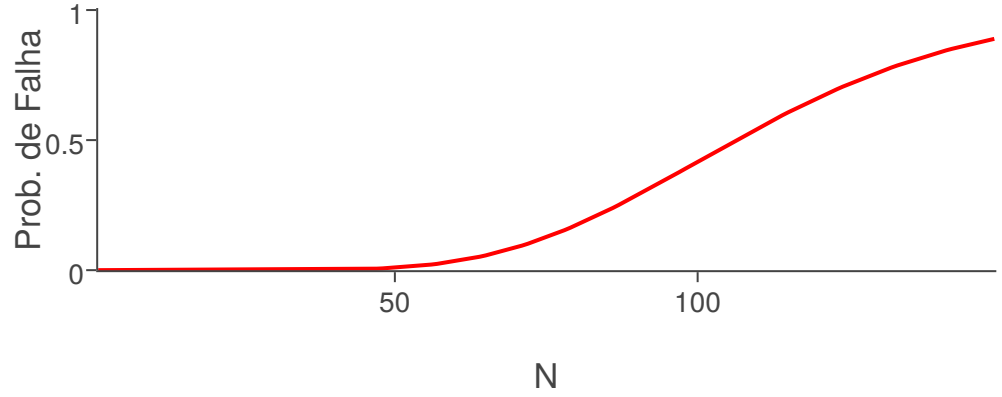
Comutação de Pacotes: Multiplexação Estatística (V)

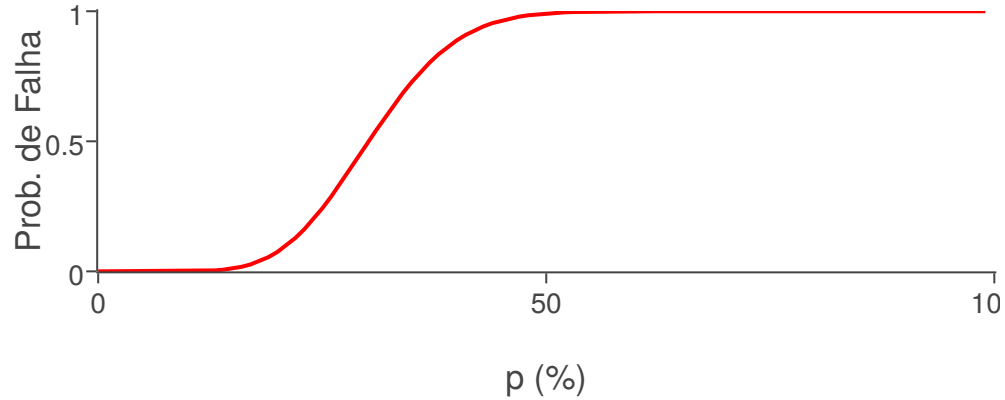
$$Pr(k > x) = \sum_{i=x}^{N} \binom{N}{i} \cdot p^i \cdot (1-p)^{(N-i)}$$

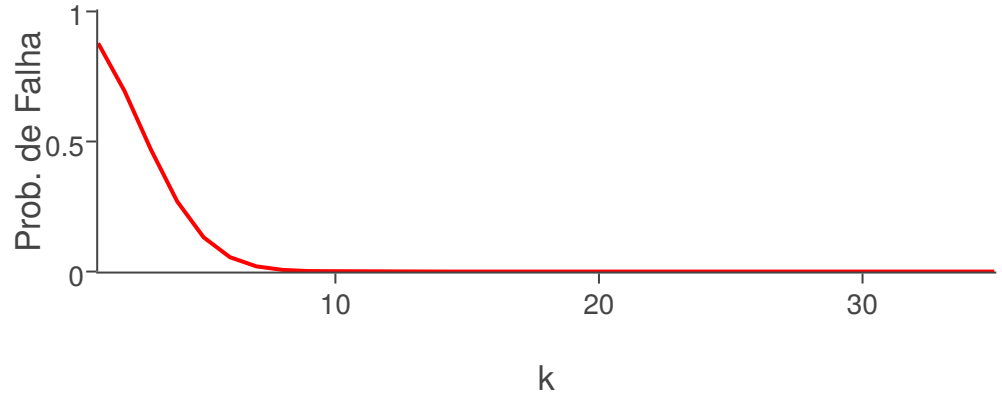
Parâmetros

k: 10
N: 35
p: 0.1
Recalcular

- Onde:
  - k: máximo de usuários simultâneos.
    - Depende da capacidade do enlace/rede e da banda para cada usuário.
  - N: número total de usuários.
  - p: probabilidade de um dado usuário estar ativo.

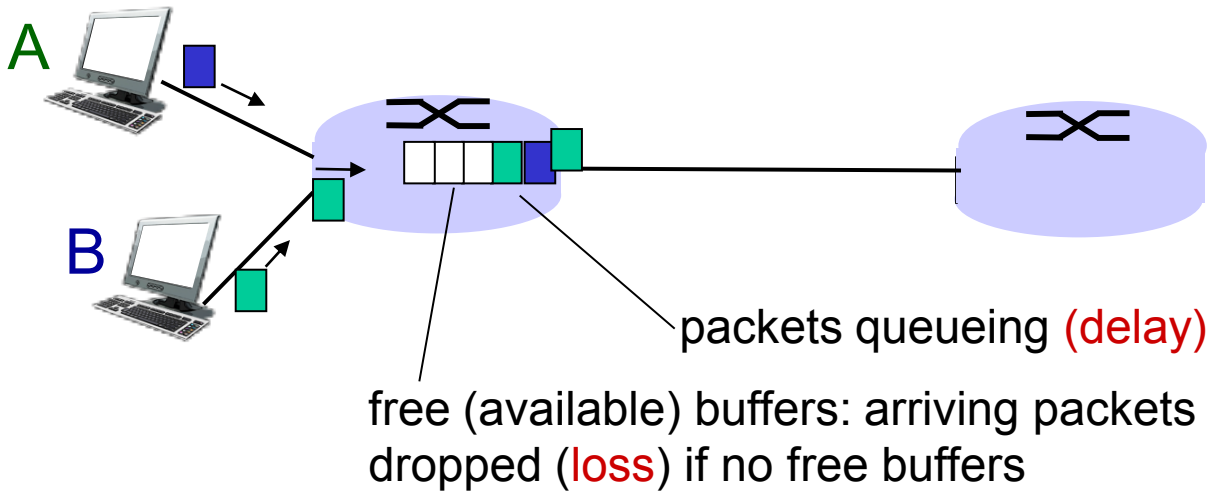






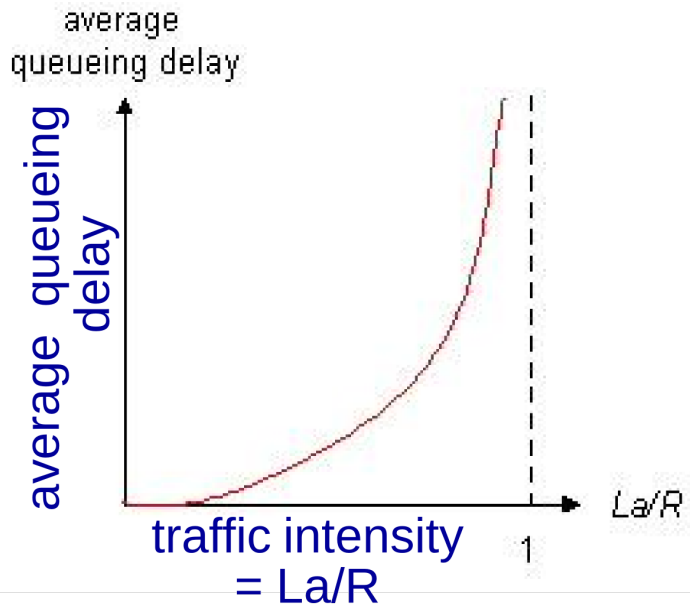
Comutação de Pacotes: Enfileiramento e Atraso (I)

- Já sabemos que enfileiramento pode levar a descarte de pacotes.
  - Pacote chega ao comutador, não há mais espaço disponível no *buffer*.
- Mas o enfileiramento tem outra consequência: **o aumento no atraso**.
- Pacote enfileirado precisa aguardar que todos os pacotes a sua frente sejam transmitidos.
  - Maiores filas → maior tempo de espera.
- *e.g.*, assumindo que cada pacote leve 10 ms para ser transmitido.
  - Com 5 pacotes na fila, 50 ms.
  - Com 100 pacotes na fila, 1 s.



Comutação de Pacotes: Enfileiramento e Atraso (II)

- Sejam:
  - R: capacidade do enlace (b/s).
  - L: tamanho dos pacotes (b).
  - a: taxa de chegada de pacotes (p/s).
- Logo, $L \cdot a$ é a taxa de chegada em b/s.
- Se:
  - $\frac{L \cdot a}{R} \approx 0$, pacotes esperam pouco na fila.
  - $\frac{L \cdot a}{R} \rightarrow 1$, pacotes esperam muito na fila.
  - $\frac{L \cdot a}{R} > 1$, mais trabalho que o comutador consegue suportar → tempo de espera "infinito".
    - *i.e.*, descarte de pacotes inevitável!

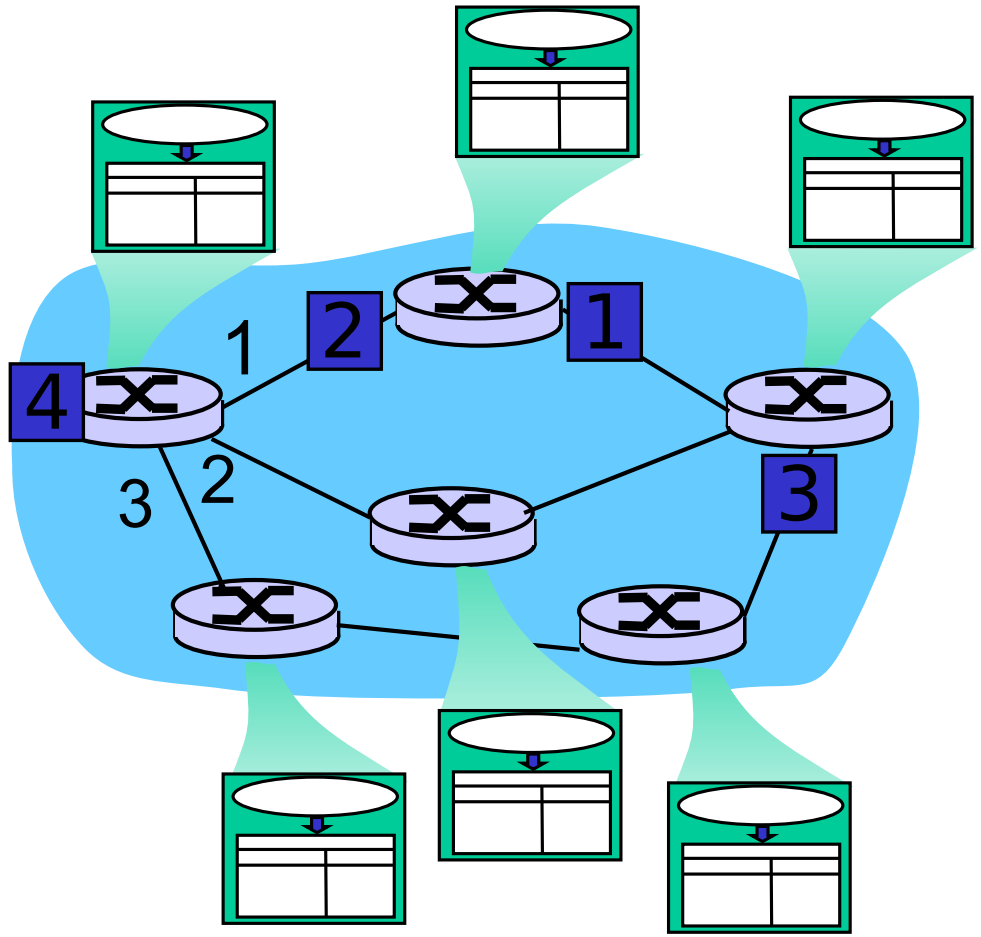


La/R ~ 0

La/R -> 1

Comutação de Pacotes: Reordenação

- Outra característica da comutação de pacotes.
- Pacotes podem “mudar de ordem”.
  - *i.e.*, pacote transmitido antes, pode chegar depois.
- Razão mais comum: caminhos distintos.
  - Decisão de como encaminhar pacote é (normalmente) do comutador.
  - Guiada por um processo de **roteamento**.
  - Roteamento pode mudar escolhas **dinamicamente**.



Comutação de Pacotes *vs.* Comutação de Circuitos

- **Pergunta:** Qual abordagem é melhor?
- **Resposta:** depende do cenário, objetivos.
- Comutação de pacotes funciona bem para tráfego em **rajada**.
  - Permite compartilhamento melhor dos recursos.
  - Não necessita de estabelecimento de chamada.
- Mas pode resultar em **congestionamento excessivo**.
  - Filas longas, descartes de pacotes, atrasos.
  - Não podemos **prometer** muita coisa.
- Certas aplicações precisam de garantias.
  - Como prover garantias em uma rede de comutação de pacotes?
  - Problema ainda em aberto.

Comutação de Pacotes: Quem Lida com os Problemas?

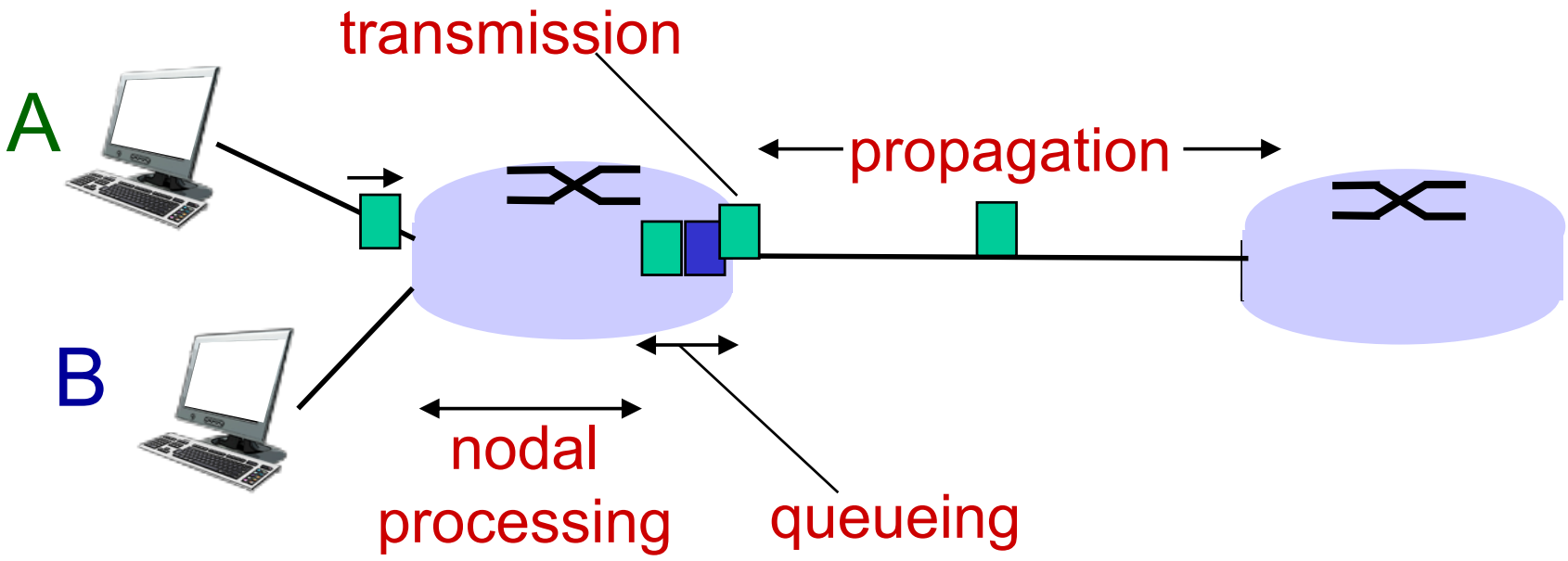
- Usamos a Internet todos os dias para transmitir arquivos.
- Obviamente, queremos que estes arquivos cheguem perfeitos, **íntegros** no destinatário.
- Como isso é possível, se a rede pode descartar, reordenar pacotes?
- **Resposta:** os terminais precisam lidar com isso.
  - *i.e.*, implantar mecanismos de recuperação.
  - **Assumindo que isto seja desejável**.
  - Argumento fim-a-fim, inteligência nas bordas.

Métricas de Desempenho

Métricas de Desempenho em Redes

- Há **quatro** métricas clássicas de desempenho em redes:
  - Atraso: tempo que pacote leva para sair de um ponto do sistema até outro.
  - Perda de pacotes: fração dos pacotes transmitidos que são descartados.
  - Vazão: o quão rapidamente dados são transmitidos entre uma origem e um destino.
  - *Jitter*: variação do atraso.
- Fim-a-fim *vs.* por salto:
  - Métricas podem ser definidas a cada salto/enlace.
  - Mas também podem ser calculadas considerando a comunicação fim-a-fim.
    - Desde o nó de origem até o nó de destino.

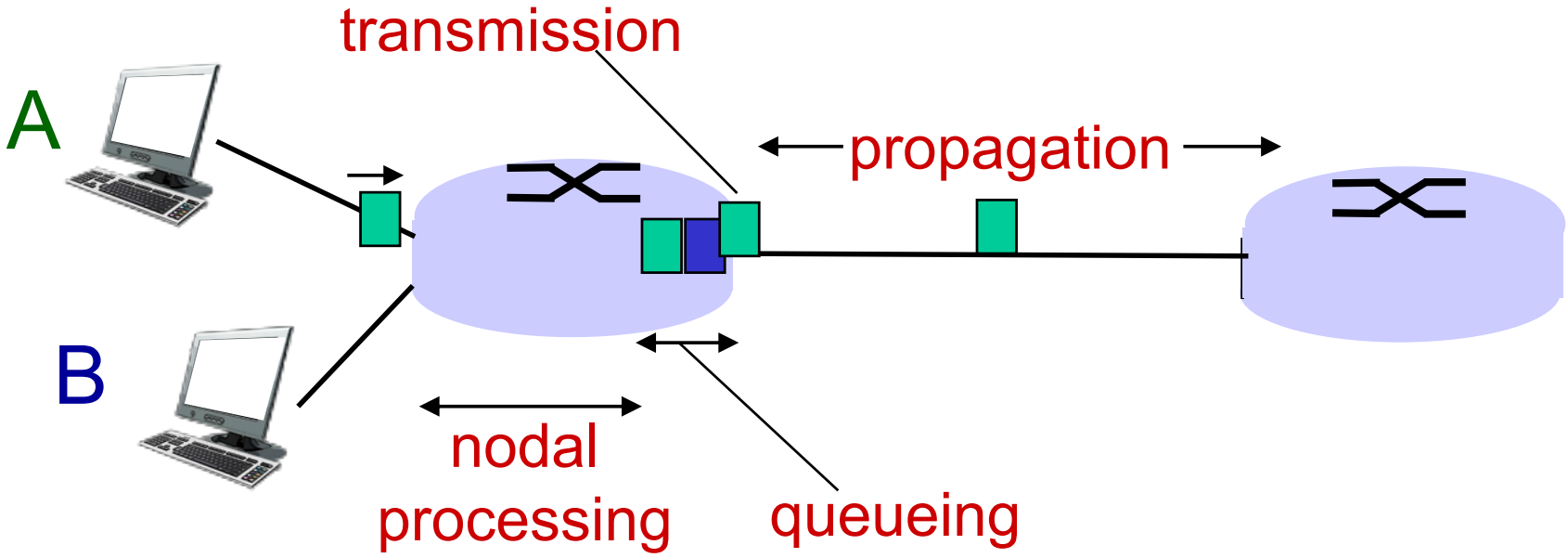
Quatro Fontes de Atraso (I)



$$d_{nodal} = d_{proc} + d_{fila} + d_{trans} + d_{prop}$$

- **d_{proc}: atraso de processamento.**
  - Verificar integridade.
  - Determinar enlace de saída.
  - ...
  - Geralmente, < ms
- **d_{fila}: atraso de enfileiramento.**
  - Tempo que pacote aguarda a fila para ser transmitido.
  - Depende do nível de congestionamento do roteador.

Quatro Fontes de Atraso (II)



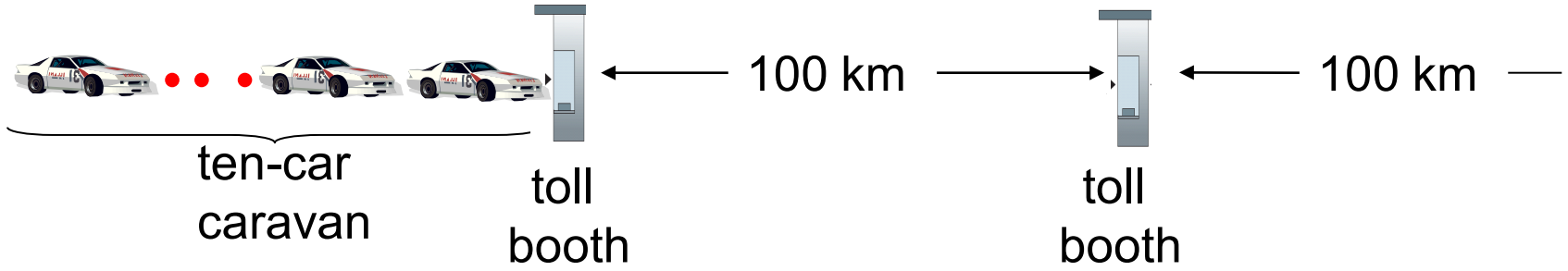
$$d_{nodal} = d_{proc} + d_{fila} + d_{trans} + d_{prop}$$

- **d_{trans}: atraso de transmissão.**
  - L: comprimento do pacote (bits).
  - R: capacidade do enlace (b/s).
  - $d_{trans} = \frac{L}{R}$.
- **d_{prop}: atraso de propagação.**
  - d: comprimento do meio físico.
  - s: velocidade de propagação do sinal no meio.
  - $d_{prop} = \frac{d}{s}$.

d_{trans} e d_{prop} são **muito diferentes!**

Atraso de Transmissão *vs.* Atraso de Propagação (I)

- Analogia de caravana:

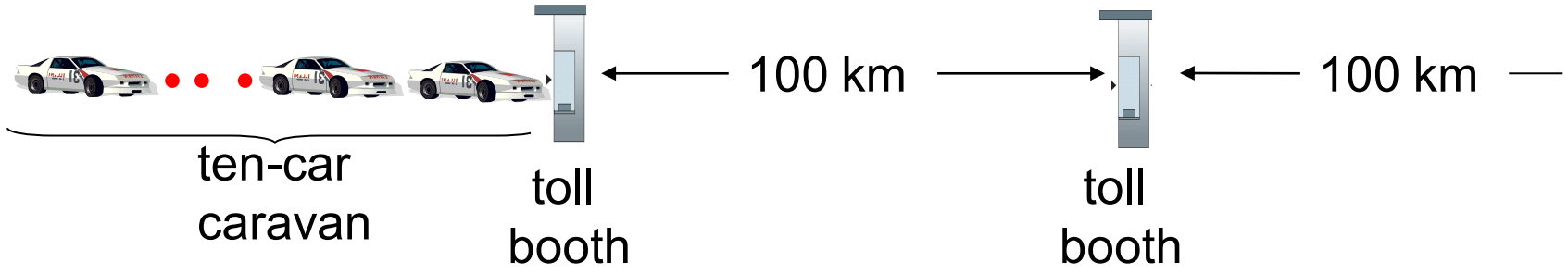


- Carros "se propagam" a 100 Km/h.
- Cabine de pedágio leva 12 segundos para servir carro.
  - Tempo de "transmissão de um bit".
  - "Taxa de transmissão": $\frac{1}{12}$ carros/s.
- Carro ~ bit, caravana ~ pacote.
- **Pergunta: quanto tempo até a caravana estar toda enfileirada na segunda cabine?**

- Tempo para "colocar" toda a caravana na estrada = $12 \cdot 10 = 120$ segundos.
- Tempo para que o último carro se propague da primeira para a segunda cabine = $\frac{100\ km}{100\ km/h} = 1$ hora.
- **Ou 62 minutos.**

Atraso de Transmissão *vs.* Atraso de Propagação (II)

- Analogia de caravana:



- Suponha que agora carros propaguem a 1000 km/h.
- E cabines agora levam um minuto para servir cada carro.
- **Pergunta: um ou mais carros chegarão à segunda cabine antes que todos sejam servidos pela primeira?**
  - **Sim!** Depois de 7 minutos, primeiro carro chega à segunda cabine.
    - Mas três ainda serão servidos pela primeira.

Atraso de Transmissão *vs.* Atraso de Propagação (III)

Atraso de Propagação (ms)	200
Taxa de Transmissão (kb/s)	120
Tamanho do Pacote (B)	1000

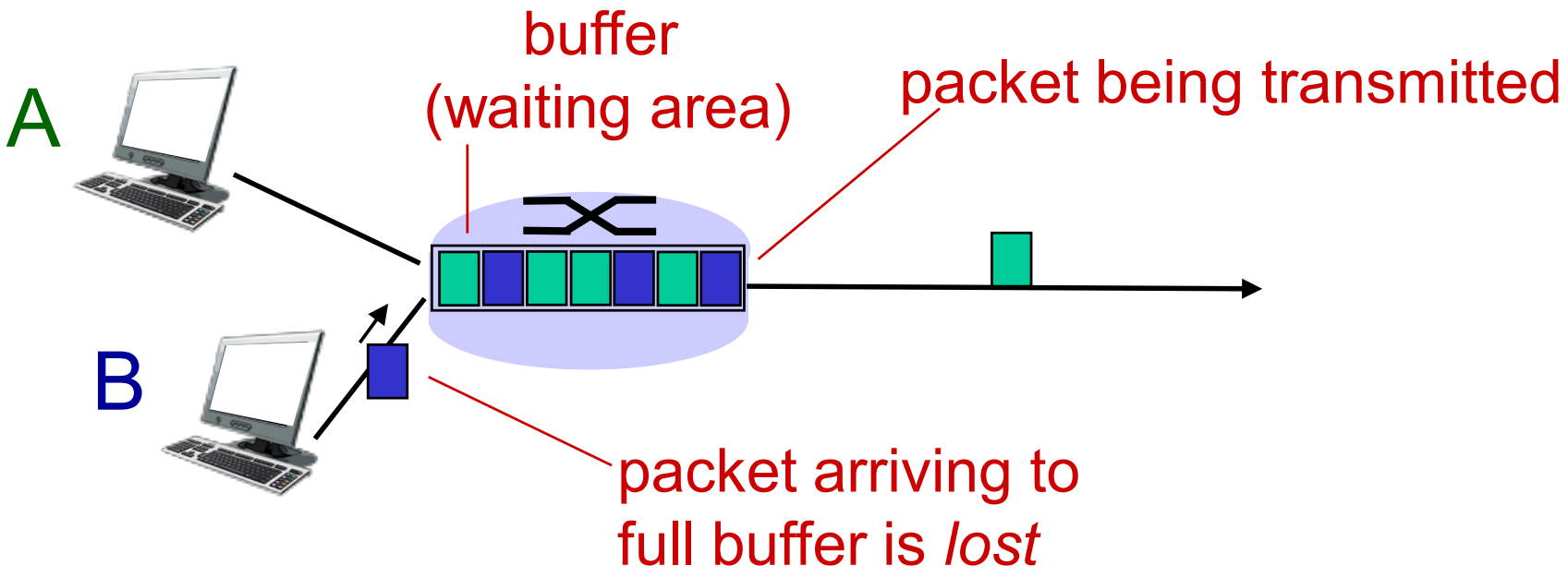
Transmitir

t = 0 ms

Último bit é colocado no canal em t =
Primeiro bit chega ao receptor em t =
Último bit chega ao receptor em t =

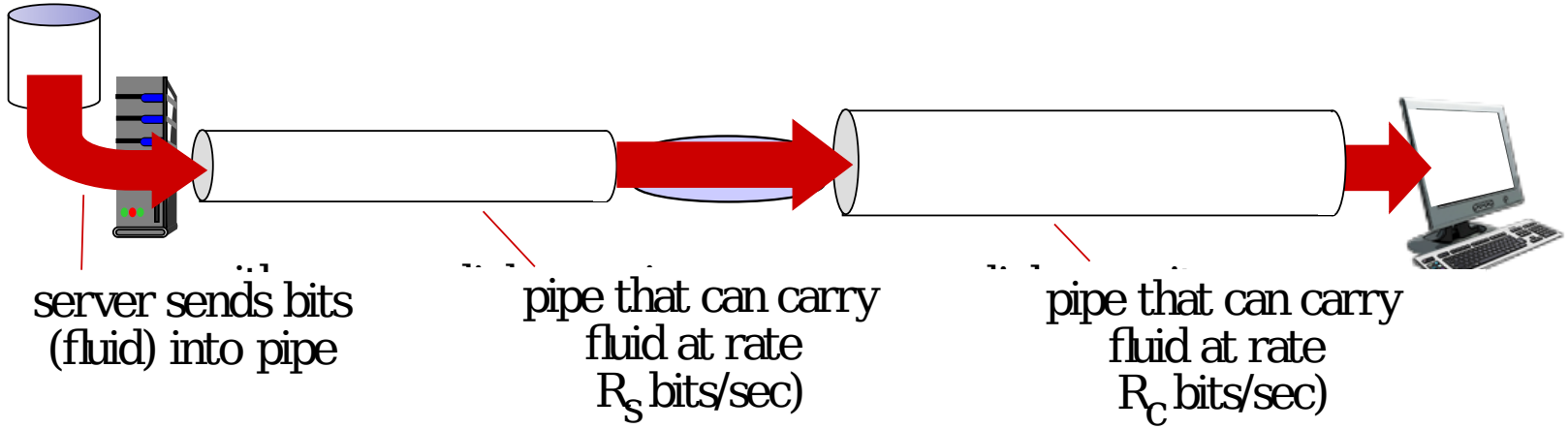
Perdas de Pacotes

- Filas (ou *buffers*) têm capacidade finita.
- Pacote que chega a um *buffer* causa descarte.
  - Do pacote recém chegado ou de outro.
  - Também conhecido como **perda**.
- Pacotes perdidos **podem** ser retransmitidos.
  - Pelo nó anterior (menos comum).
  - Pela origem (mais comum).



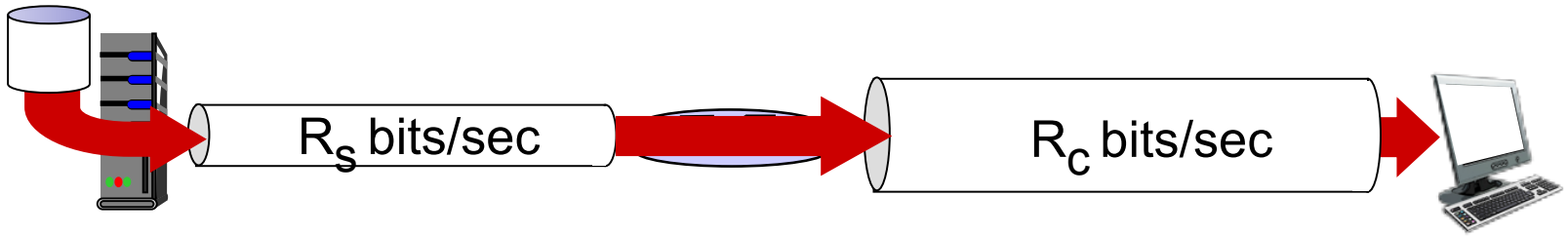
Vazão (I)

- **Vazão:** taxa (bits/unidade de tempo) na qual bits são transferidos entre origem e destino.
  - **Instantânea:** vazão em um ponto no tempo.
  - **Média:** taxa considerando um período mais longo.
- Vazão do fluxo *vs.* taxa de transmissão do fluxo.
  - Aplicação pode gerar pacotes a uma taxa alta e vazão ser limitada pela rede.
  - Rede pode ter capacidade alta, mas aplicação pode gerar fluxo com baixa taxa.

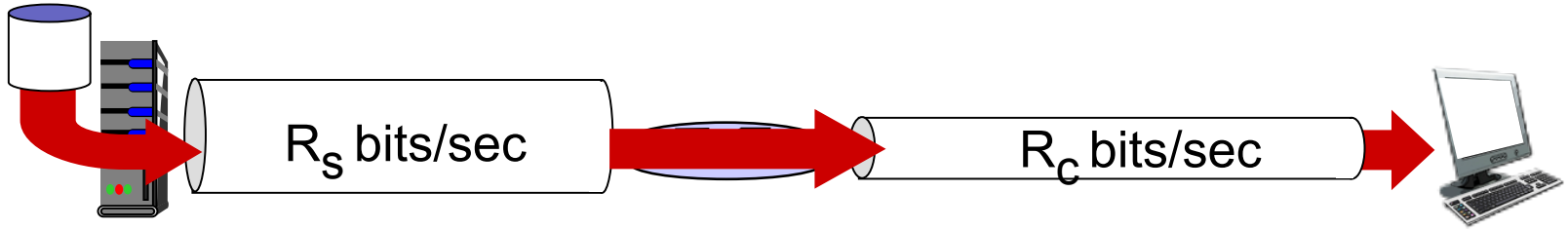


Vazão (II)

$R_s < R_c$ Qual é a vazão média fim-a-fim?



$R_s > R_c$ Qual é a vazão média fim-a-fim?



Enlace de Gargalo

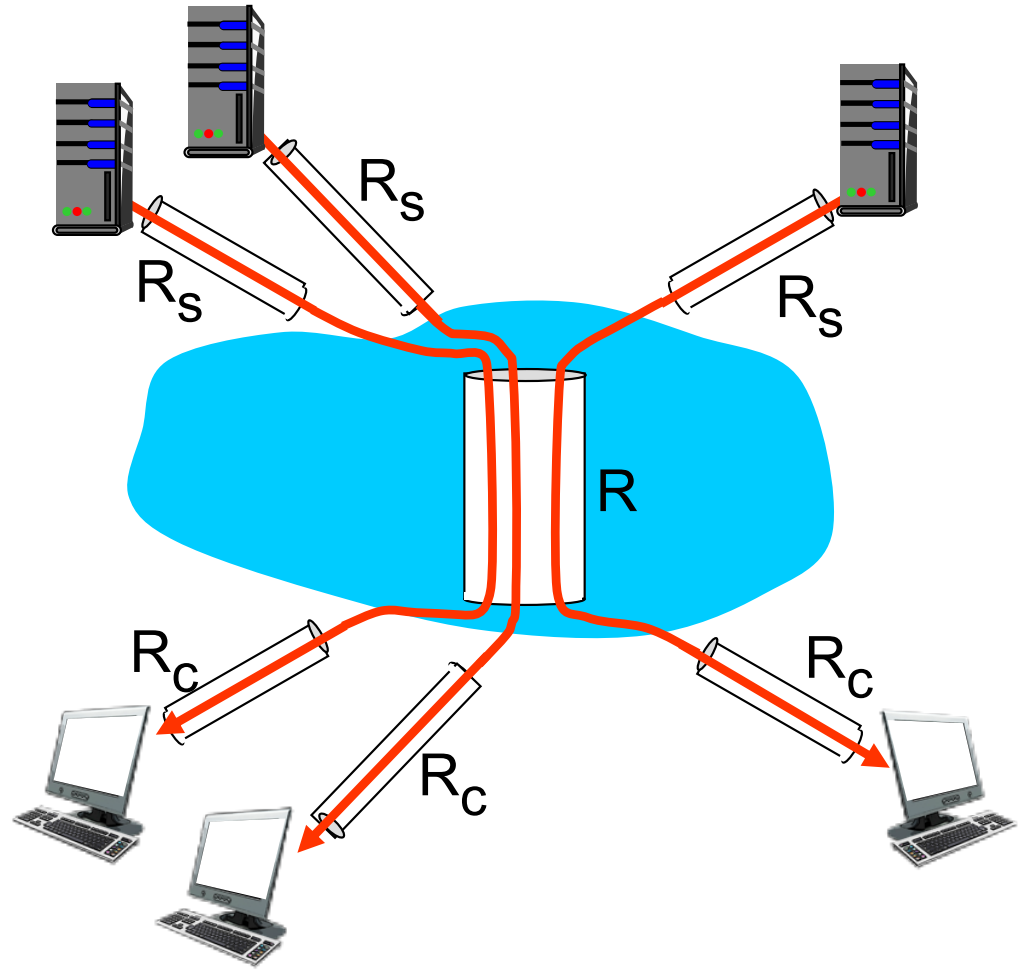
Enlace no caminho fim-a-fim que **restringe** vazão fim-a-fim.

Cenário de Vazão na Internet

- Vazão fim-a-fim por conexão:

$$min\left(R_c, R_s, \frac{R}{10}\right)$$

- Na prática: R_c ou R_s são normalmente os gargalos.



10 conexões compartilham (de forma justa) o enlace de gargalo no backbone