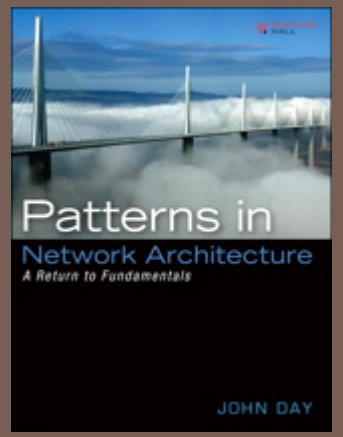


ADIVINHANDO AS CAMADAS

Capítulo 6

Patterns in Network Architecture

- *Em tudo a perfeição é atingida não quando não houver mais nada a adicionar e sim quando não houver mais nada a ser retirado...*

  - Antoine de Saint Exupery

- *Interligação em rede é uma comunicação entre processos.*

  - Robert Metcalfe, 1972

Introdução

Preâmbulo

- Nos capítulos anteriores revimos a nossa experiência com sistemas em rede e descobrimos algumas das estruturas arquiteturais dos protocolos.
- Identificamos propriedades invariantes e componentes comuns.
- Agora estamos prontos para considerar o que estes padrões nos dizem sobre montá-los em estruturas maiores.
- Tradicionalmente usamos camadas, mas estas são problemáticas.
- Será que são mesmo as camadas o princípio organizacional correto para redes?

Juntando os Protocolos

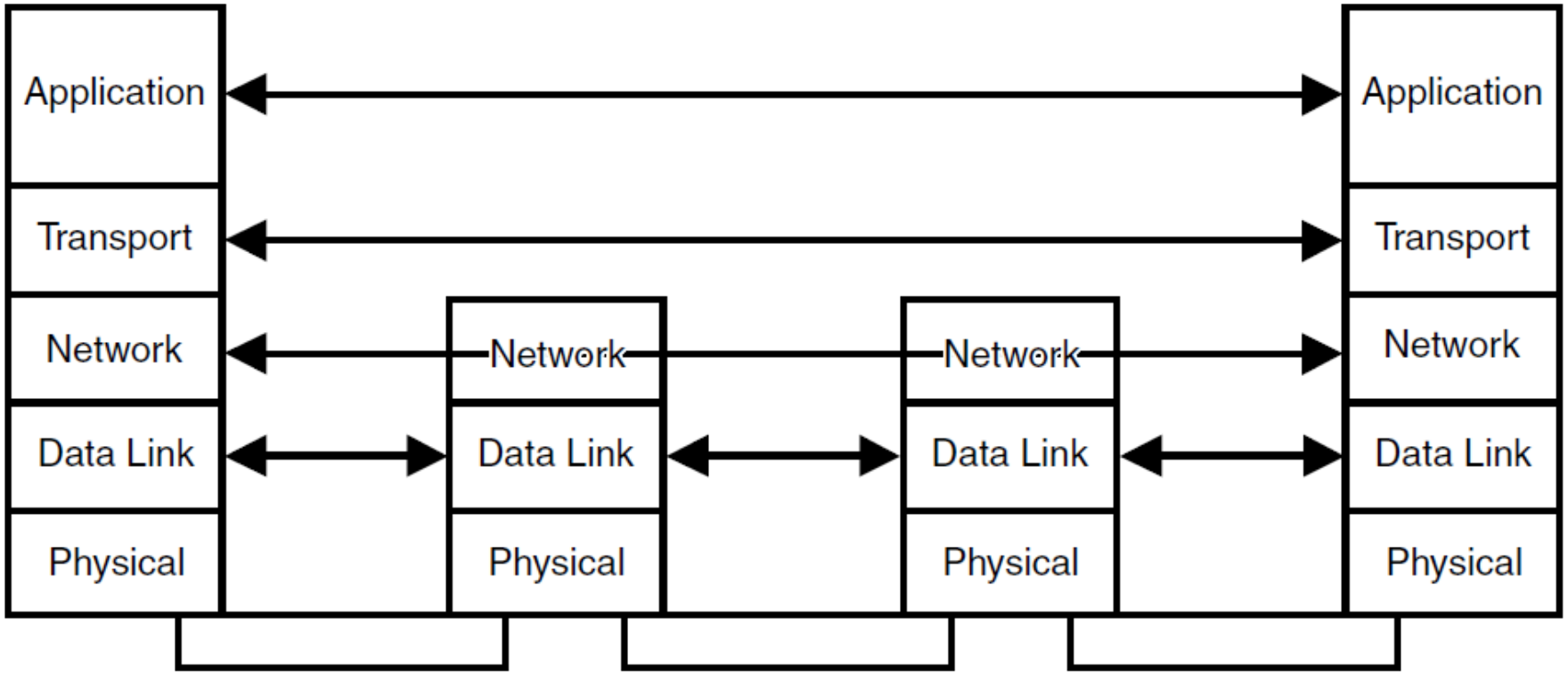
O que nós vimos

- O modelo de referência OSI foi o primeiro a tentar definir de modo mais rigoroso os elementos de uma arquitetura:
  - "Uma camada é a coleção de subsistemas de mesmo grau (*rank*)".
  - "Subsistemas são a interseção de um sistema com uma camada".
    - (ISO 7498-1, 1984, 1994)
  - Implicação: uma camada era o conjunto de todas as máquinas de protocolos de mesmo nível em um conjunto de sistemas: um protocolo, uma camada.

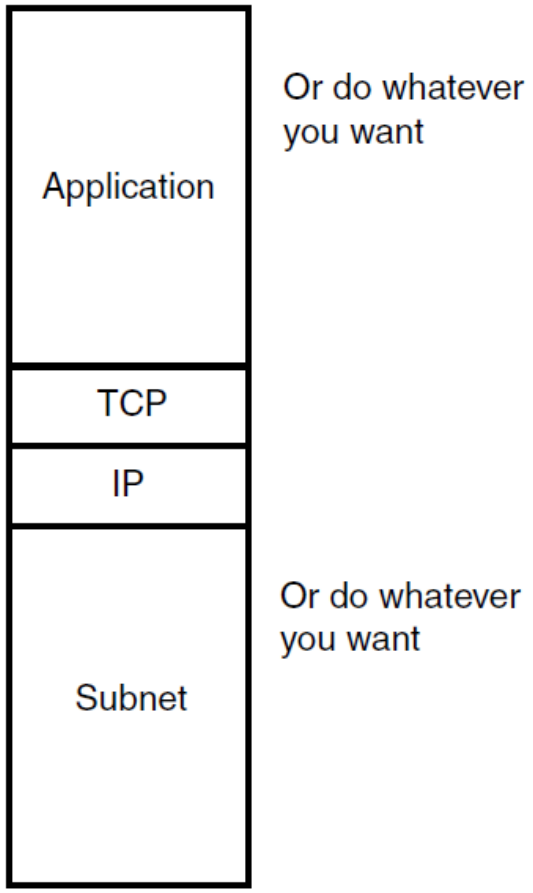
O que nós vimos

- Os limites de uma camada era denominado de seu *escopo*.
- Há pouco ou quase nada no modelo OSI que indica o que está ou não está em uma camada.
- O uso do termo *subsistema* indica que deveria haver mais do que o protocolo; caso contrário teria sido usado o termo *entidade*.

Arquitetura Inicial de Redes



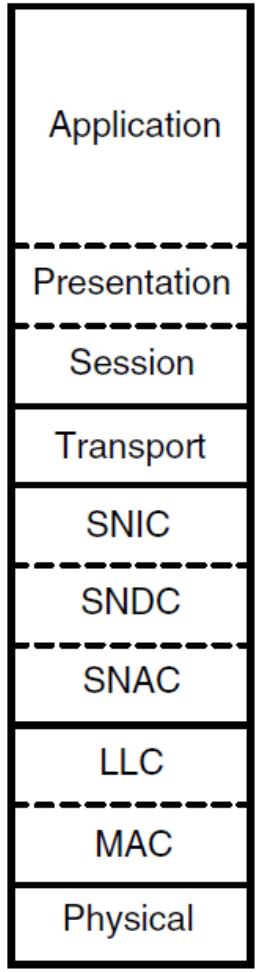
Os Modelos Internet e OSI



Or do whatever you want

Or do whatever you want

Supposedly, seven layers.
But . . .



The Application Layer was more complex and some layers didn't exist.

And there were more of others.

Problemas com a Implementação

- A teoria de camadas requer que cada camada seja uma máquina de estados distinta
- O cruzamento de uma fronteira de camada geralmente envolve uma API
- Há muito *overhead* nas mudanças de contexto.
- A junção de máquinas de protocolos em camadas adjacentes poderia ser feito, mas na prática há fatores que tornam esta abordagem não sábia.

Divisão em Camadas

- Talvez a ideia da divisão em camadas não foi uma ideia tão boa no final das contas.
- As camadas tradicionais estabelecidas desde 1970 não estão nos mostrando o caminho.
- Deve haver algo que não estamos vendo.
- O que o problema nos diz?

Escutando o Problema

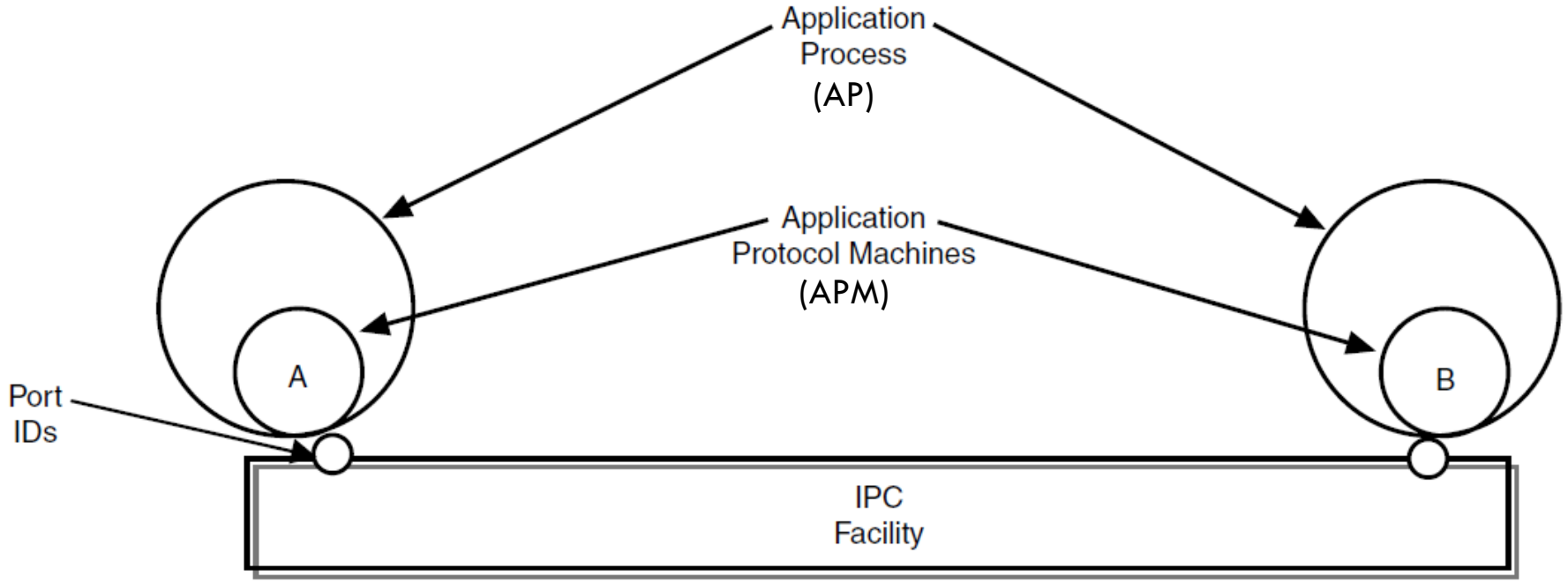
Primeiros Passos

- Vamos começar com os elementos fundamentais da comunicação entre computadores:
  - Dois processos de aplicação tentando se comunicar e
  - Um mecanismo para obter a comunicação dentro de um *mesmo* sistema de processamento.
- Depois iremos sucessivamente expandir o domínio do problema para a comunicação entre dois sistemas, mais do que dois pares de aplicações, mais do que dois sistemas, etc.

Primeiros Passos

- Mostraremos que a partir dos seguintes elementos conseguimos construir todo o resto:
  - Processos de aplicação comunicantes e
  - Recurso de IPC distribuído consistindo de:
    - Protocolo que proveja um mecanismo de IPC e
    - Protocolo para gerenciar um IPC distribuído.

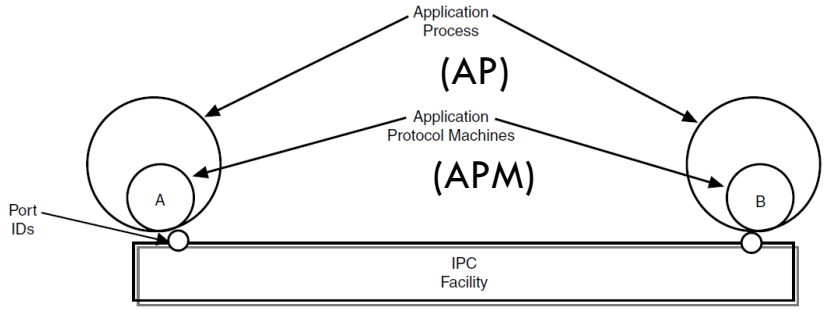
Comunicação em um Mesmo Sistema



- Toda as interações com o IPC são realizadas pelas APMs em nome da aplicação.

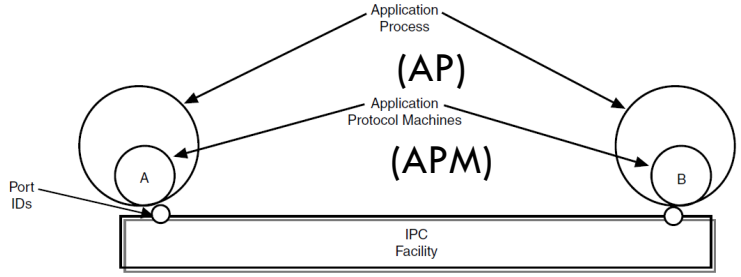
API Simples e Genérica (APM ↔ IPC)

- <result> ← Allocate(<destination-application-name>, <port_id>, <properties>)
- <result> ← Send(<port-id>, <buffer ptr>)
- <result> ← Receive(<port-id>, <buffer ptr>)
- <result> ← De-allocate(<port-id>)



Propriedades dos Elementos Principais

- Nomeação para IPC:
  - Os nomes das aplicações são não ambíguos dentro do sistema como um todo.
  - Os nomes das aplicações são usados entre aplicações e pelo IPC para identificar os destinos.
  - As port-ids têm um escopo menor e são não ambíguos apenas entre um AP e o recurso de IPC [pelo menos este é o requisito mínimo]



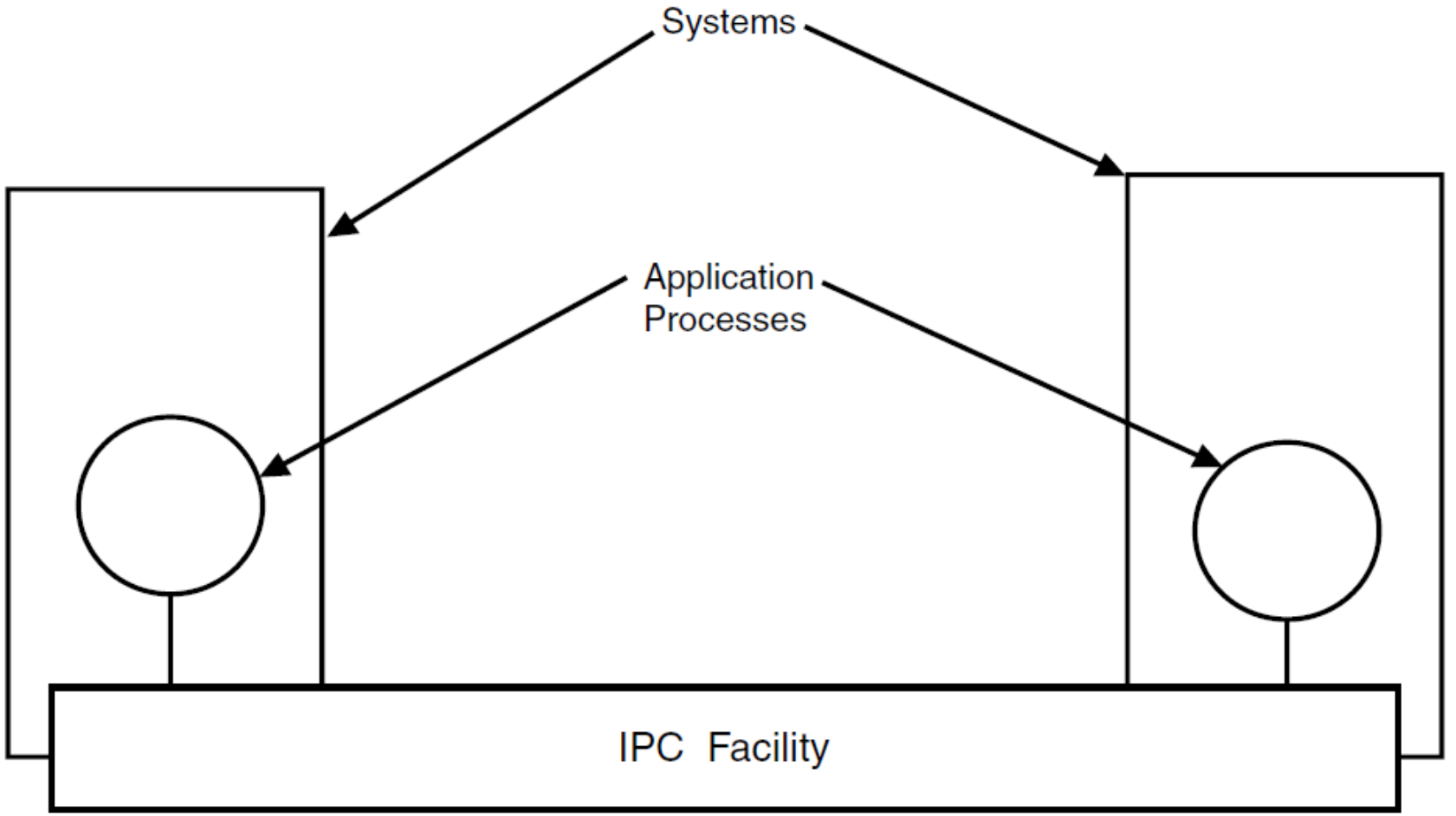
Propriedades dos Elementos Principais

- O processo de aplicação e a máquina de protocolo de aplicação:
  - A finalidade principal de um AP é realizar alguma tarefa para a qual seja necessária a comunicação.
- O recurso de IPC:
  - Assume-se que a sequência de estabelecimento seja assimétrica
  - O recurso IPC move confiavelmente mensagens do espaço de memória de um AP para o espaço de memória do outro.
  - Os SOs fazem isto de diversas formas. Muitos usam a semântica de passagem de mensagens.

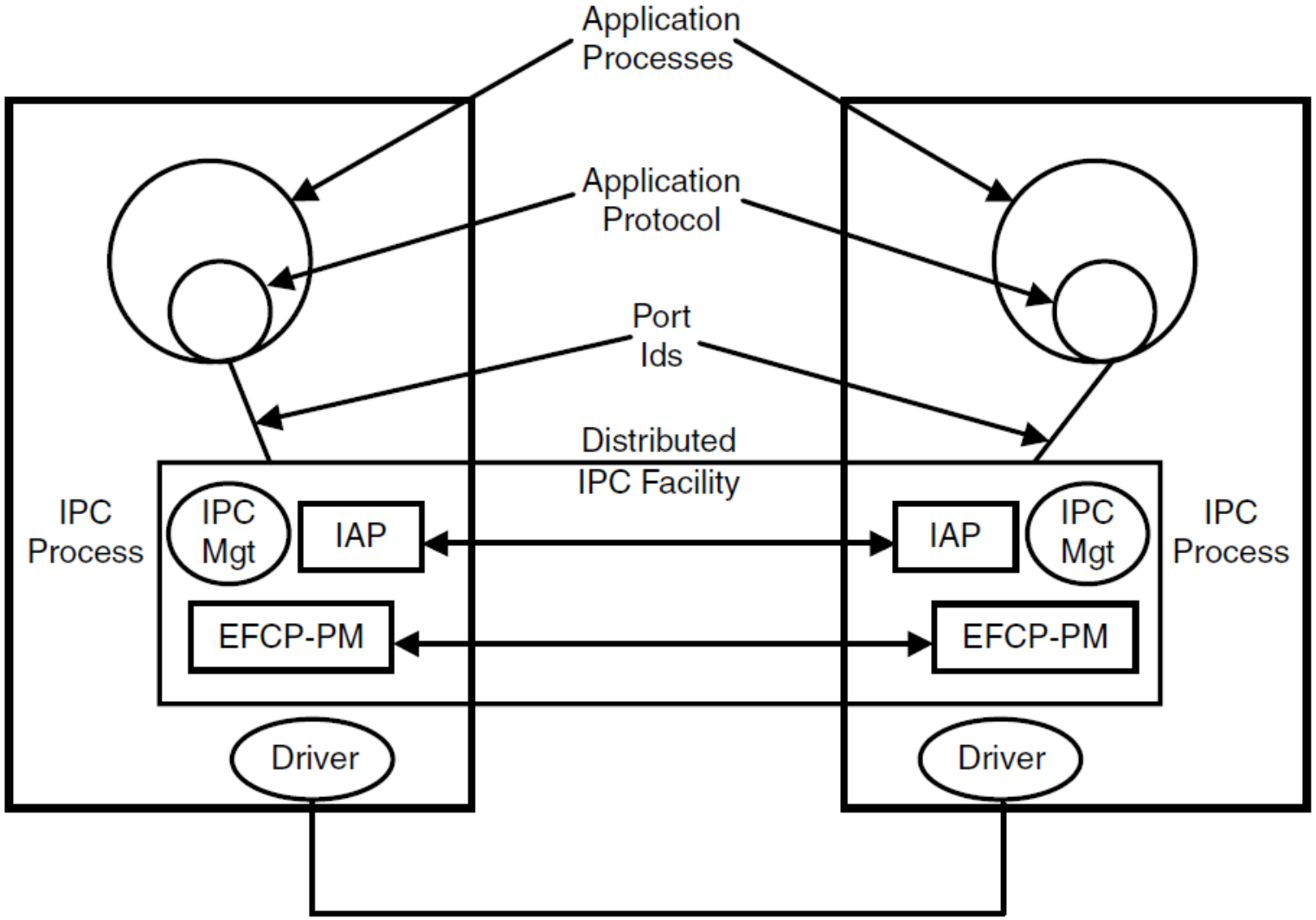
Propriedades dos Elementos Principais

- O recurso de IPC (cont.):
  - Invocações sucessivas do “Allocate” com o mesmo par (origem, destino) produzirá a alocação de port-ids distintos.
  - O recurso de IPC mantém uma associação entre os port-ids dos dois pontos terminais do fluxo. As mensagens são entregues na mesma ordem em que foram enviadas.
  - Em muitos casos o controle de fluxo é implícito.
  - Controle de erro não é uma preocupação importante, pois é improvável que as mensagens sejam perdidas.

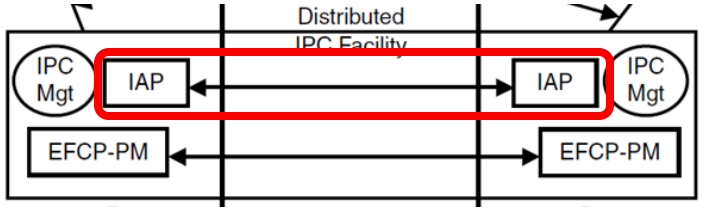
Comunicação entre Dois Sistemas



Novos Elementos Necessários para a Comunicação entre Dois Sistemas

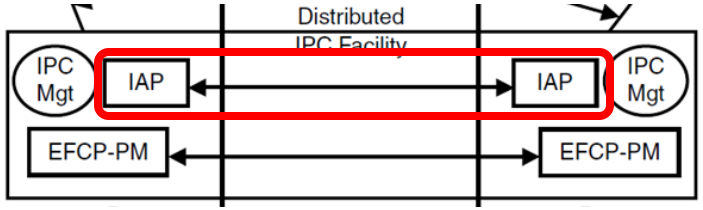


Comunicação entre Dois Sistemas



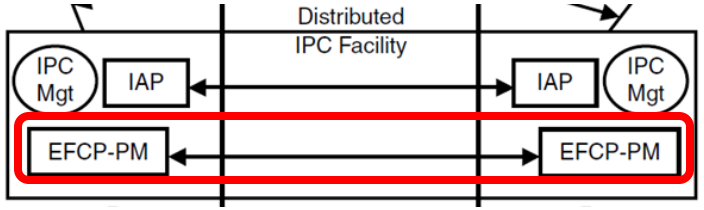
- Temos que determinar se a aplicação está no outro sistema e se o solicitante tem permissão para abrir uma conexão até lá.
- Precisamos de um mecanismo para perguntar ao processo IPC remoto se uma dada aplicação existe no seu sistema e passar informação suficiente sobre a aplicação origem de modo a saber se deve permitir a comunicação.
- Isto requer um protocolo para transportar os nomes das aplicações e outras informações:
  - Isto é feito pelo IAP *(IPC Access Protocol)*
    - Na RINA passou a ser chamado de "*Flow Allocator*"

IAP:
IPC Access Protocol



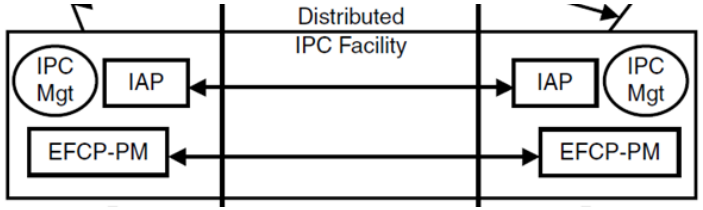
- Pode ser um protocolo de pergunta/resposta bem simples que deve transportar os nomes das aplicações origem e destino assim como informações de controle de acesso.
- O IAP deve incluir autenticação?
  - Não. A autenticação deve ser parte da Aplicação
  - Apenas a aplicação origem pode confirmar que a aplicação destino seja quem ela alega ser.
  - Portanto, a autenticação deve ser realizada diretamente entre as aplicações origem e destino.
  - O controle de acesso em um SO (e por extensão no IPC) pode apenas determinar se a aplicação origem tem acesso a uma aplicação de destino que ele acredita ser aquela que está sendo solicitada.

EFCP – *Error and Flow Control Protocol*



- Passando agora para sistemas distintos, não temos mais uma memória compartilhada.
- O nosso meio de comunicação é sujeito a erros e temos que ter protocolos de controle de erro e de fluxo entre os dois sistemas.
- O recurso de IPC cria um estado compartilhado com o correspondente no outro lado usando mecanismos tais como CRCs, FECs, números de sequência e janela deslizante para prover confiabilidade e controle de fluxo.
- O EFCP produz mensagens de protocolo que são entregues ao *driver* para envio no meio físico.

O IAP não deveria fazer parte do EFCP?



- Os dois protocolos estão fazendo duas funções bem diferentes com objetivos bem distintos:
  - Um está gerenciando a IPC entre dois APs
  - O outro está preocupado simplesmente em fornecer um canal de comunicação com algumas propriedades.
- Poderíamos enviar o pedido do IAP como parte do estabelecimento do EFCP se assim quiséssemos.
  - Mas, como veremos depois, há vantagens e flexibilidades oriundas de mantê-los separados.

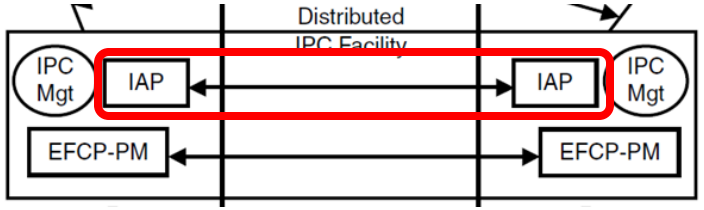
Hipóteses Invalidadas

- O gerenciamento do espaço de nomes não está mais sob controle de um mesmo sistema
- O mesmo programa pode existir em ambos os sistemas, portanto os nomes não podem ser ambíguos.
- O SO não conhece todas as aplicações que podem existir no outro sistema, nem as permissões associadas com as mesmas, portanto é necessário um mecanismo para fornecer esta informação.
- Não se pode mais contar com os mecanismos de controle de acesso do SO local para fornecer a autorização e autenticação.

Hipóteses Invalidadas

- É necessária uma sincronização explícita porque não há mais instruções ou memória compartilhada disponível.
- O mecanismo de IPC não pode mais usar uma memória comum. Consequentemente os dados podem ser corrompidos ou perdidos.
- Pelos mesmos motivos, o controle de fluxo deve agora ser explícito e não implícito.

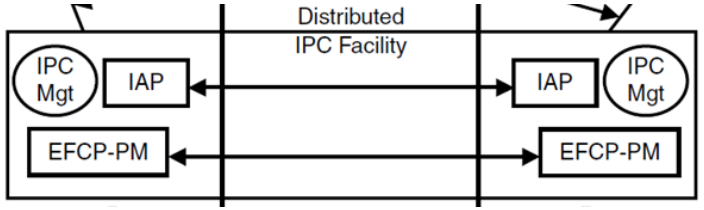
Novos Elementos



- Protocolo de controle de acesso IPC (IAP):
  - Protocolo usado para comunicar os nomes das aplicações origem e destino e outras informações necessárias para determinar se a comunicação pode ser estabelecida.

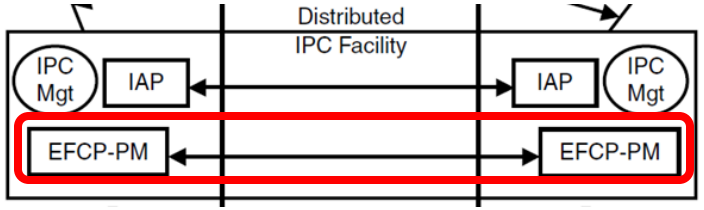
Op	Dest Appl Name	Src Appl Name	QoS	Capability

Novos Elementos



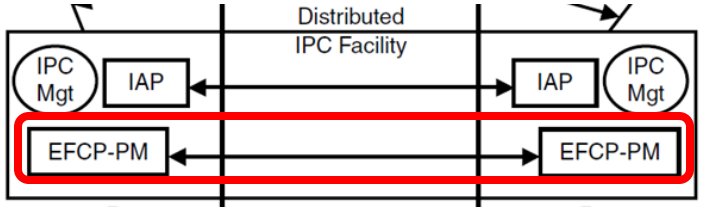
- Processo IPC:
  - Processo em um sistema que implementa e gerencia o IPC.
    - Isto envolve a coordenação com outros processos IPC em outro sistema, tanto para efetuar a comunicação como para gerenciar o recurso IPC distribuído.
- Recurso IPC distribuído:
  - Um recurso IPC abrangendo dois ou mais sistemas com pelo menos um processo IPC em cada sistema participante.

Novos Elementos



- Protocolo de controle de erro e de fluxo (EFCP):
  - Protocolo necessário para substituir os mecanismos de memória compartilhada de um mesmo sistema, para garantir confiabilidade e controle de fluxo na comunicação entre os dois sistemas.
  - Uma EFCP PM é uma tarefa do processo IPC

Novos Elementos

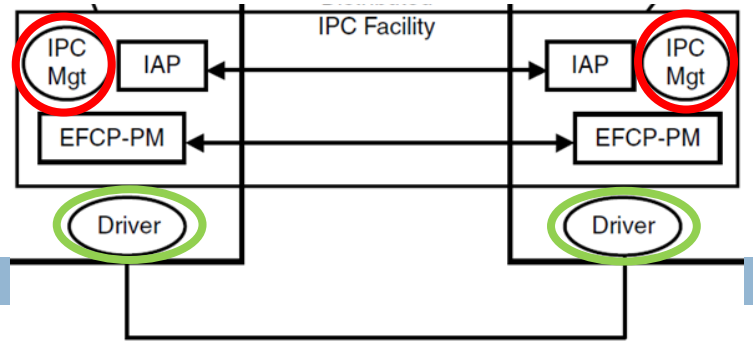


- Protocolo de controle de erro e de fluxo (EFCP):

EFCP Data transfer PDU

EFCP Control PDUs: Synch, Ack, Flow Control

Novos Elementos



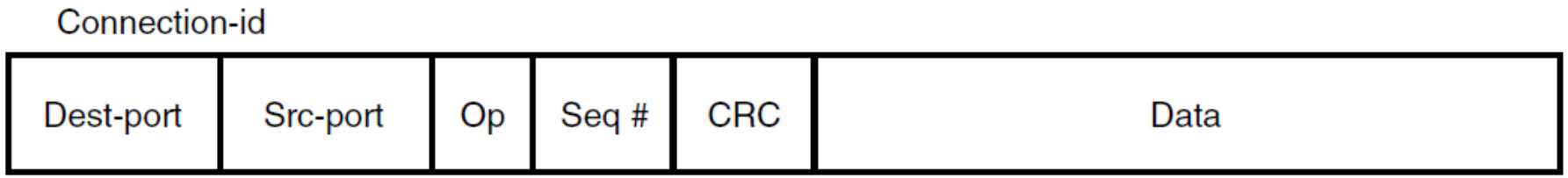
- Tarefa de gerenciamento do IPC:
  - Uma tarefa do processo IPC que gerencia o IPC e as interfaces para as regras de busca e controle de acesso do SO local.
  - Ela também gerencia os recursos IPC.
- Driver:
  - Uma aplicação ou biblioteca de procedimentos no SO que interfaceia o meio físico.
  - Apesar de ambos os sistemas comunicantes possuírem drivers interfaceando o meio físico, os drivers não são máquinas de protocolos porque não compartilham nenhum estado.

Comunicação Simultânea entre Dois Sistemas

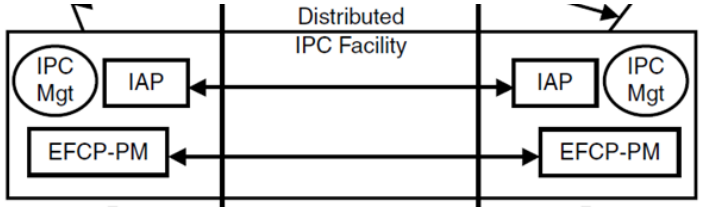
- Se as aplicações estivessem no mesmo sistema, iríamos criar múltiplas instâncias do mecanismo IPC, uma para cada diálogo.
  - Para dois sistemas fazemos o mesmo.

Comunicação Simultânea entre Dois Sistemas

- O recurso IPC deve ser capaz de suportar múltiplas instâncias do EFCP ao mesmo tempo.
  - As PDUs necessitam então de um identificador de conexão.
  - Como cada um dos lados pode iniciar a conexão, temos que garantir que não terminem usando o mesmo identificador de conexão.
  - Como as portas são únicas em cada sistema e estamos enviando os dados para um único sistema, basta incluí-las:



Opções do Processo IPC na Alocação



- Cada solicitação *Allocate* da API gerará uma solicitação do IAP.
  - O pedido do IAP deverá levar os port-ids a serem usados para esta conexão e transportar os port-ids do destino na resposta.
- Ações possíveis por parte do processo IPC:
  - Iniciar a criação de uma nova instância do EFCP com as políticas apropriadas associadas aos port-ids, o equivalente ao estabelecimento de uma conexão TCP ou HDLC.
  - Alocar recursos e criar associações às port-ids, equivalente ao UDP.
- Ou:
  - Cada sistema pode manter um conjunto de conexões EFCP já criadas e simplesmente associar às mesmas as portas alocadas.

Gerenciamento do Meio Físico (mesmo recurso)

- Precisamos de uma nova aplicação para moderar o compartilhamento de um mesmo recurso (a definição clássica de um SO!)
  - Ela deve:
    - Manter uma tabela com os port-ids ativos
    - Aceitar PDUs das diversas instâncias do EFCP e
    - Determinar a ordem em que deve enviar as PDUs no meio físico.
  - A sua função principal é a de gerenciar o compartilhamento ou multiplexar o meio físico.

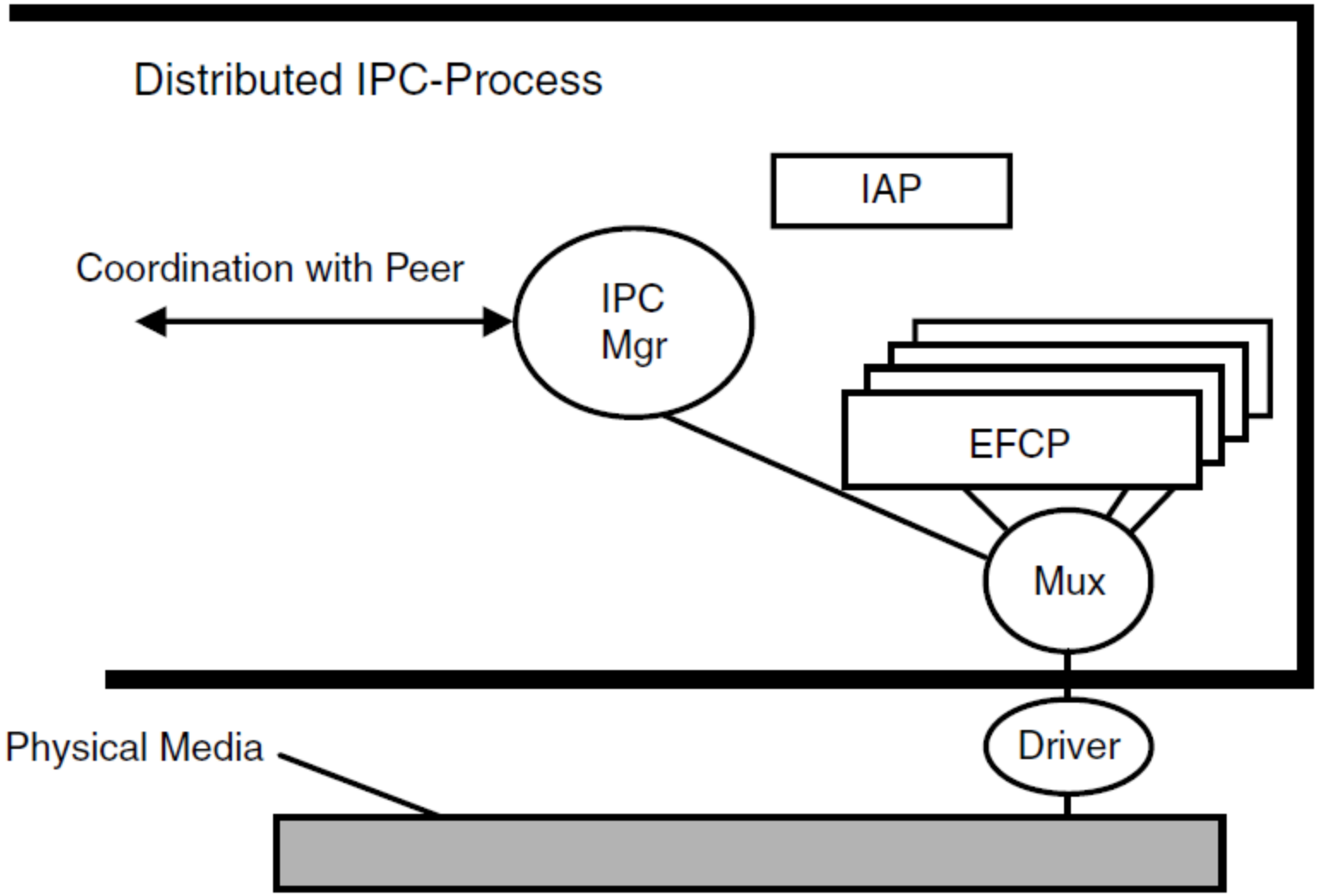
Gerenciamento do Meio Físico (mesmo recurso)

- Esta tarefa de multiplexação pode ainda:
  - Otimizar o uso do meio físico concatenando PDUs, etc.
  - Prover prioridades para certas classes de usuários ou para processamento especial entre os fluxos.
    - A aplicação solicitaria parâmetros de QoS (largura de banda, atraso, *jitter*, taxa de erros, etc.) ao solicitar a alocação de recursos de comunicação.

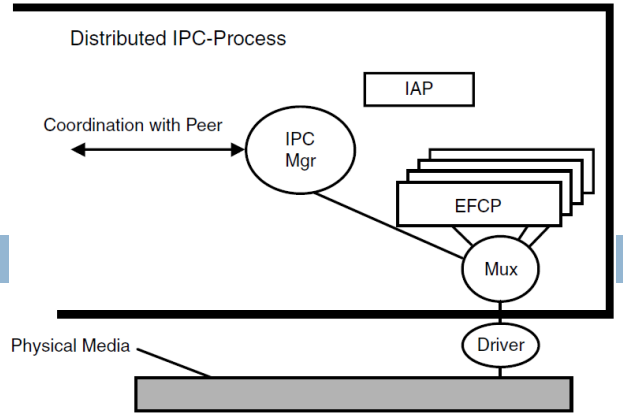
Controles Disponíveis para QoS

- Temos basicamente dois “controles” para alcançar estas características e balanceá-las com as demandas de outras solicitações:
  - Os mecanismos do EFCP ou
  - Alocação de recursos pela tarefa de multiplexação:
    - Ordem e taxa de atendimento nas filas (escalonamento de processos)
    - Gerenciamento do comprimento das filas (gerenciamento de memória)

Tarefa de Gerenciamento do IPC



Tarefa de Gerenciamento do IPC

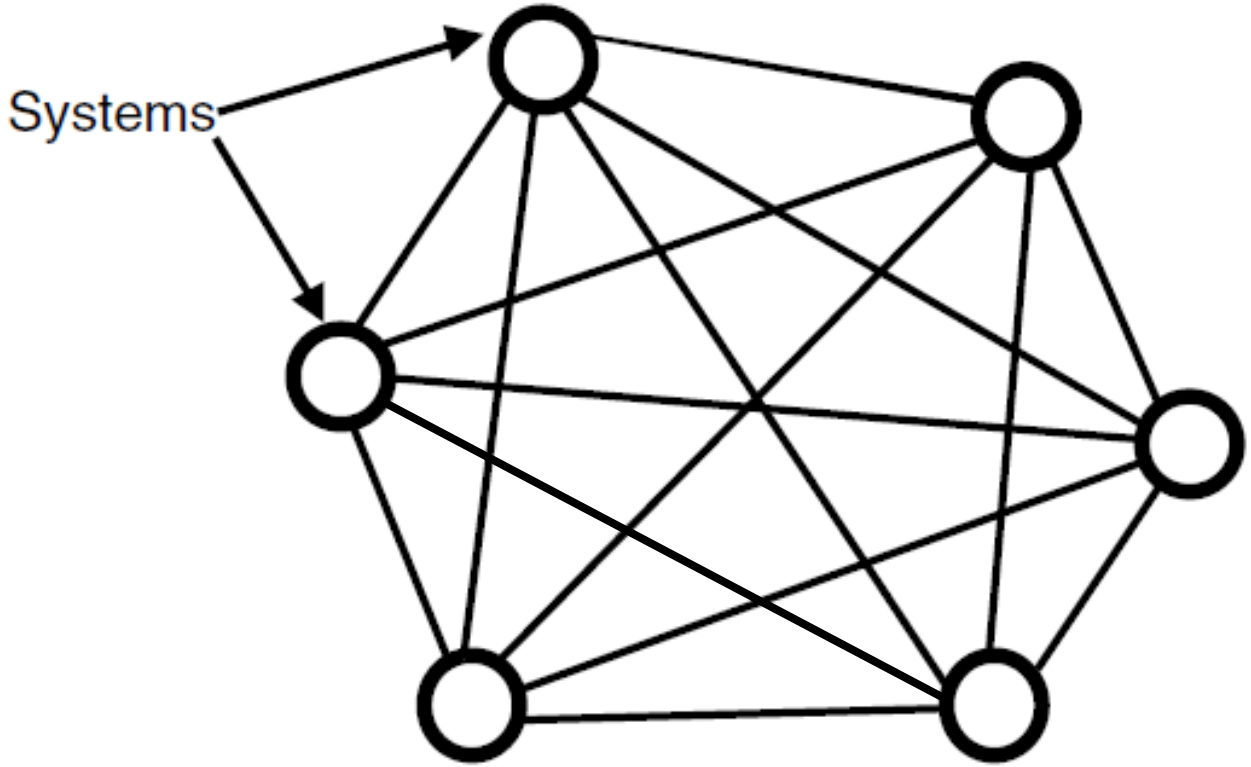


- Irá gerenciar as interações entre a tarefa de multiplexação e as instâncias do EFCP que a alimentam.
- Precisará traduzir as solicitações de Alocação das aplicações em políticas e balanceá-las com as políticas operacionais do sistema.
- Precisará ainda coordenar informações de gerenciamento de recursos com o outro sistema.

Novos Elementos

- Identificador de conexão:
  - Um identificador para distinguir uma instância do IPC (conexão) da outra.
- Tarefa de Multiplexação:
  - Uma aplicação para gerenciar a utilização das interfaces por múltiplas instâncias do protocolo IPC e outras aplicações.

Comunicação com *N* Sistemas



Comunicação com *N* Sistemas

- No caso de dois sistemas, podíamos assumir que se a aplicação solicitada não fosse local, estaria no outro sistema.
- Agora temos muitos outros locais para procurar!
  - Para *N* pequenos poderíamos enviar *N*-1 consultas e ver quem responde positivamente.
    - Mas este é um desperdício e não escala.
  - Para outros valores de *N* seria mais eficiente manter uma base de dados com quais aplicações estão aonde, ou seja um *diretório*.

Diretório

- Um diretório é a extensão das regras de busca dos SOs para cobrir os outros sistemas.
- A função de diretório pode ser realizada de diversas formas:
  - Cada sistema pode enviar para os demais uma lista completa de todas as aplicações em potencial que possui e depois atualizar sobre qualquer alteração;
  - Um sistema busca localmente o nome da aplicação e se não encontrar, consulta o outro sistema;
  - Algo intermediário (i.e., uso de *cache* e consulta)

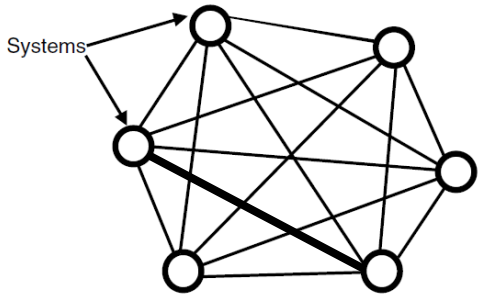
Diretório

- Para grandes valores de *N* pode ser imposta alguma organização nos nomes ou nos próprios diretórios para facilitar:
  - a busca, ou
  - a ordem de busca ou
  - a implementação de uma estratégia de cache.
- A função de diretório deverá usar o recurso IPC para realizar a sua tarefa.
- Precisaremos de um protocolo para atualizar e consultar a base de dados do diretório.

Diretório

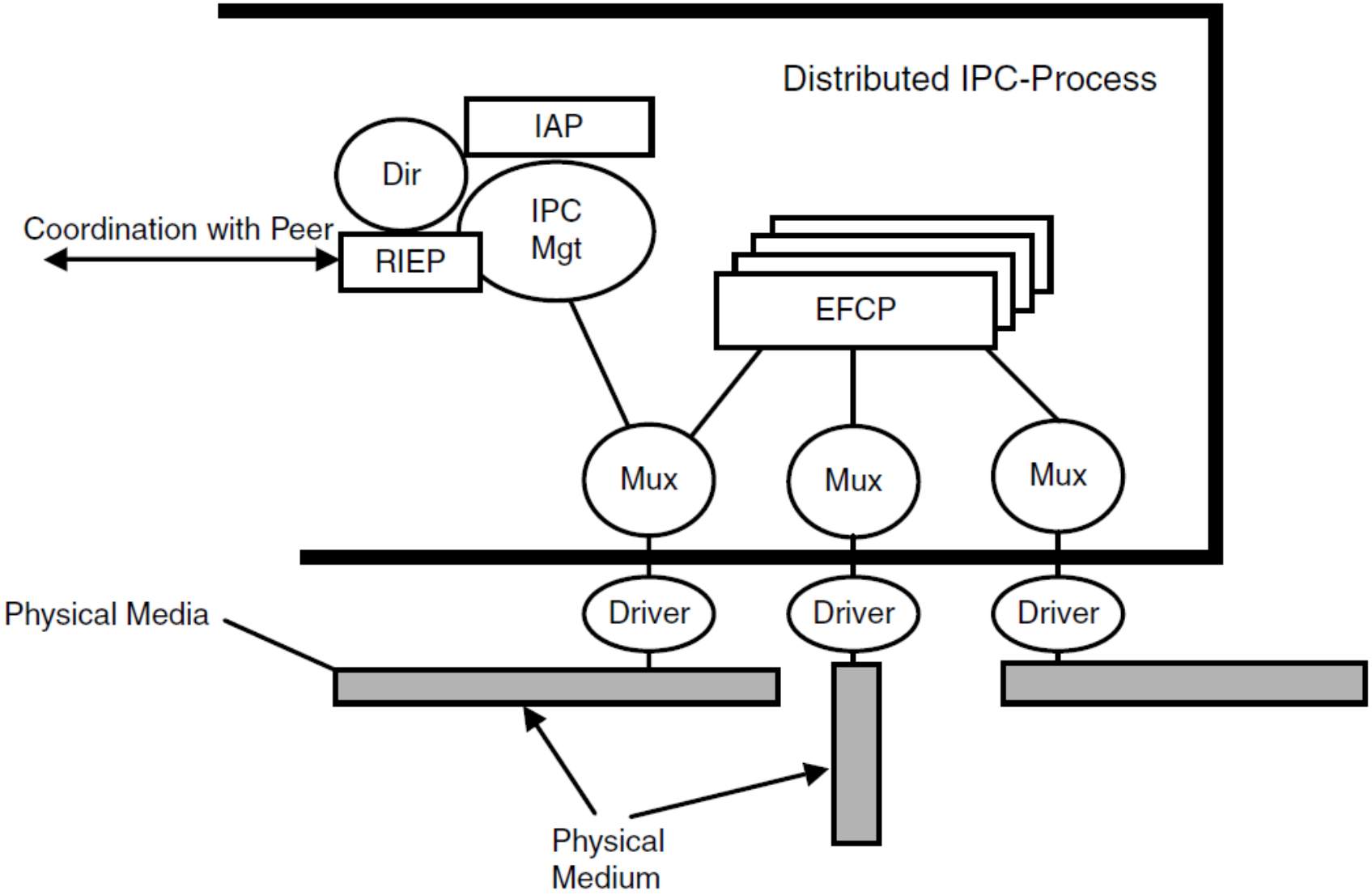
- Naturalmente haverá também outras informações relacionadas a recursos que gostaríamos de manter e que outros sistemas vão querer consultar.
- Basicamente precisaremos:
  - Manter uma base de dados com informações relacionadas à IPC que possa ser consultada por outros processos IPC em outros sistemas e
  - Uma tarefa de gerenciamento que possa notificar mudanças importantes para os demais sistemas.
    - Chamaremos este protocolo de RIEP – *Resource Information Exchange Protocol*

Escolha do Destino

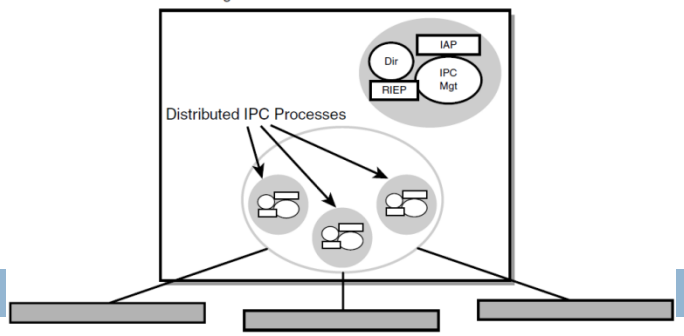


- Um destino pode ser escolhido pela simples seleção da interface física local apropriada (dado que todos os nós estão diretamente conectados).
- Cada sistema precisará associar um identificador *local* a cada interface.
  - Por exemplo, poderá ser o nome da aplicação de multiplexação para aquela interface.
  - Há uma tarefa de multiplexação separada para cada interface. Portanto, os identificadores locais identificam a que tarefa de multiplexação está associada uma dada instância EFCP.

Cada Interface é gerenciada por uma tarefa de multiplexação e outra de gerenciamento de IPC

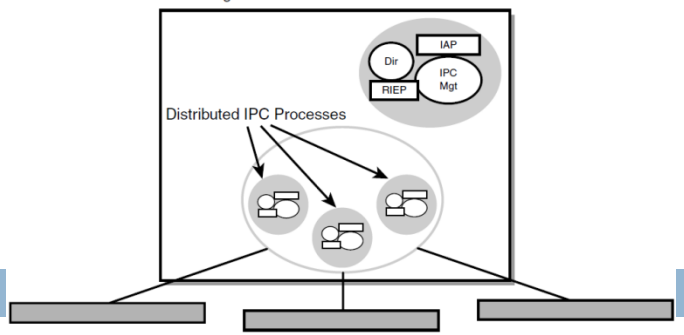


Proliferação de Processos IPC: Reorganização



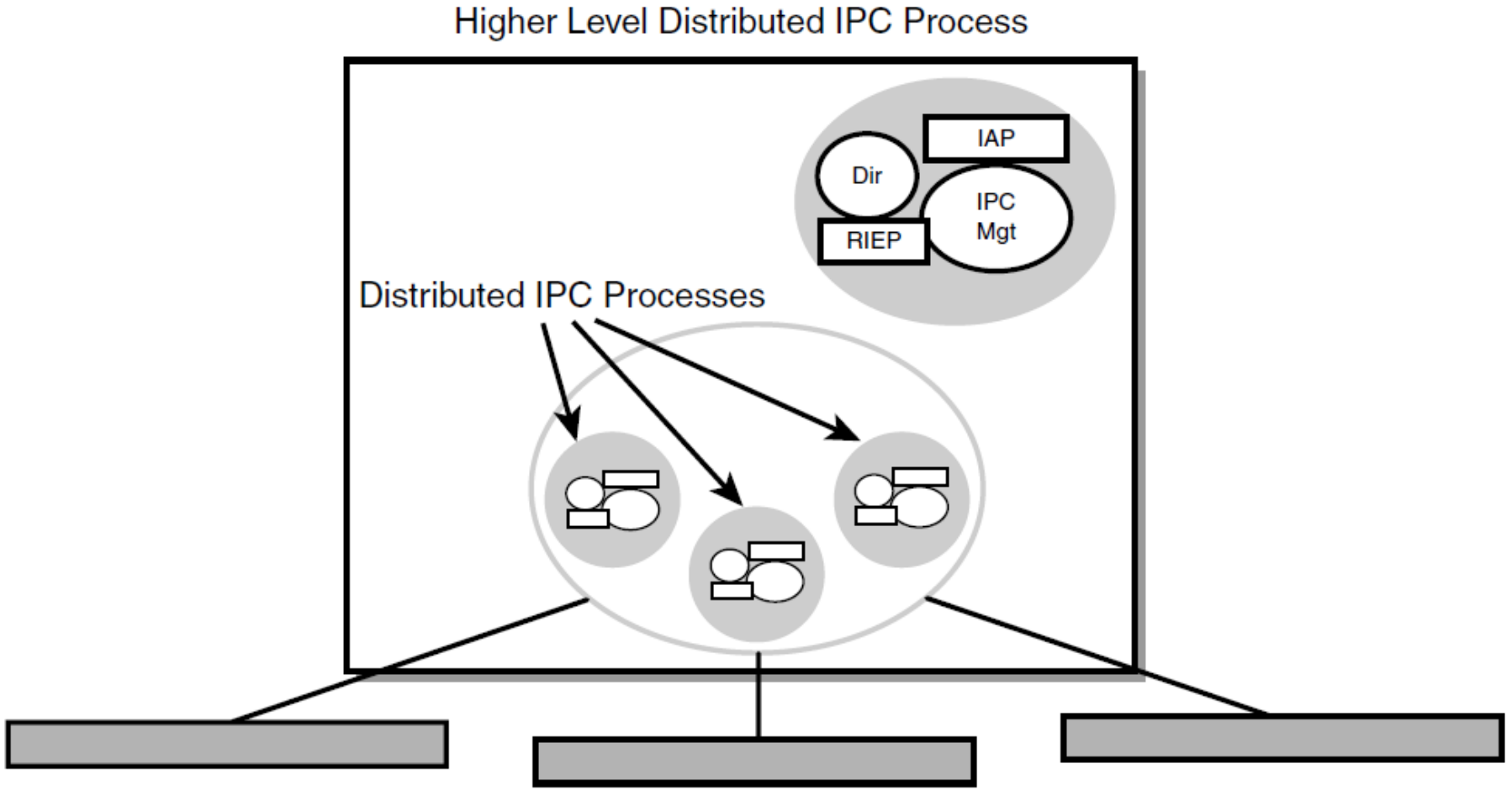
- Proliferação para tratar de diferentes fios.
- Dado que cada interface pode suportar diferentes tipos de meio, isto implicaria em instâncias de protocolos com diferentes políticas.
- Parece termos um processo IPC separado para cada interface, com as suas instâncias associadas de EFCP e gerenciamento.
- Podemos juntar estes elementos em um módulo separado e chamá-los de Recurso IPC Distribuído (DIF – *Distributed IPC Facility*).

Proliferação de Processos IPC: Reorganização



- Para isolar mais efetivamente as aplicações destas diferenças nas interfaces e gerenciar o seu uso, vamos colocar uma nova tarefa de gerenciamento IPC "acima" destes módulos DIF para:
  - Processar as chamadas das APIs pelas aplicações
  - Consultar nomes de aplicações no diretório
  - Selecionar a DIF apropriada para aquela interface.
- Mas, esta abordagem não escala muito bem e ficaria logo cara e impraticável mesmo para pequenos valores de *N*.
- Precisamos de uma rede mais complicada, mas mais barata.

Recurso IPC Distribuído (DIF – *Distributed IPC Facility*)



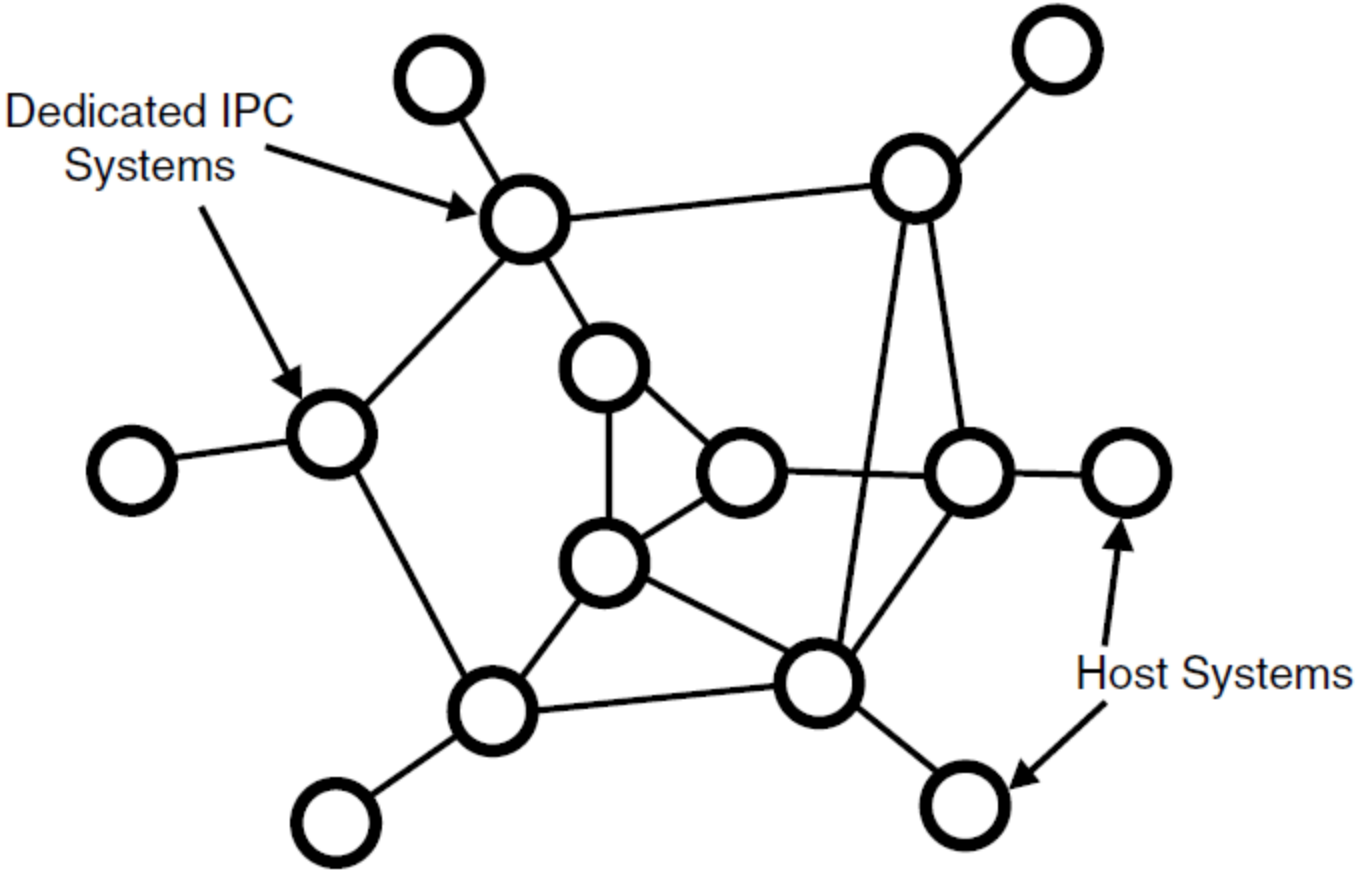
Novos Elementos

- Diretório:
  - Uma base de dados residente em cada sistema e que mantém um mapeamento entre os nomes das aplicações e as interfaces através das quais elas estão disponíveis.
- Protocolo de Troca de Informações de Recursos (RIEP):
  - Um protocolo usado para atualizar o diretório e outras informações mantidas pelo recurso IPC sobre o estado das suas conexões, alocação de recursos, etc.
  - Pode ser consultado por outros sistemas ou feita uma assinatura para receber as atualizações.

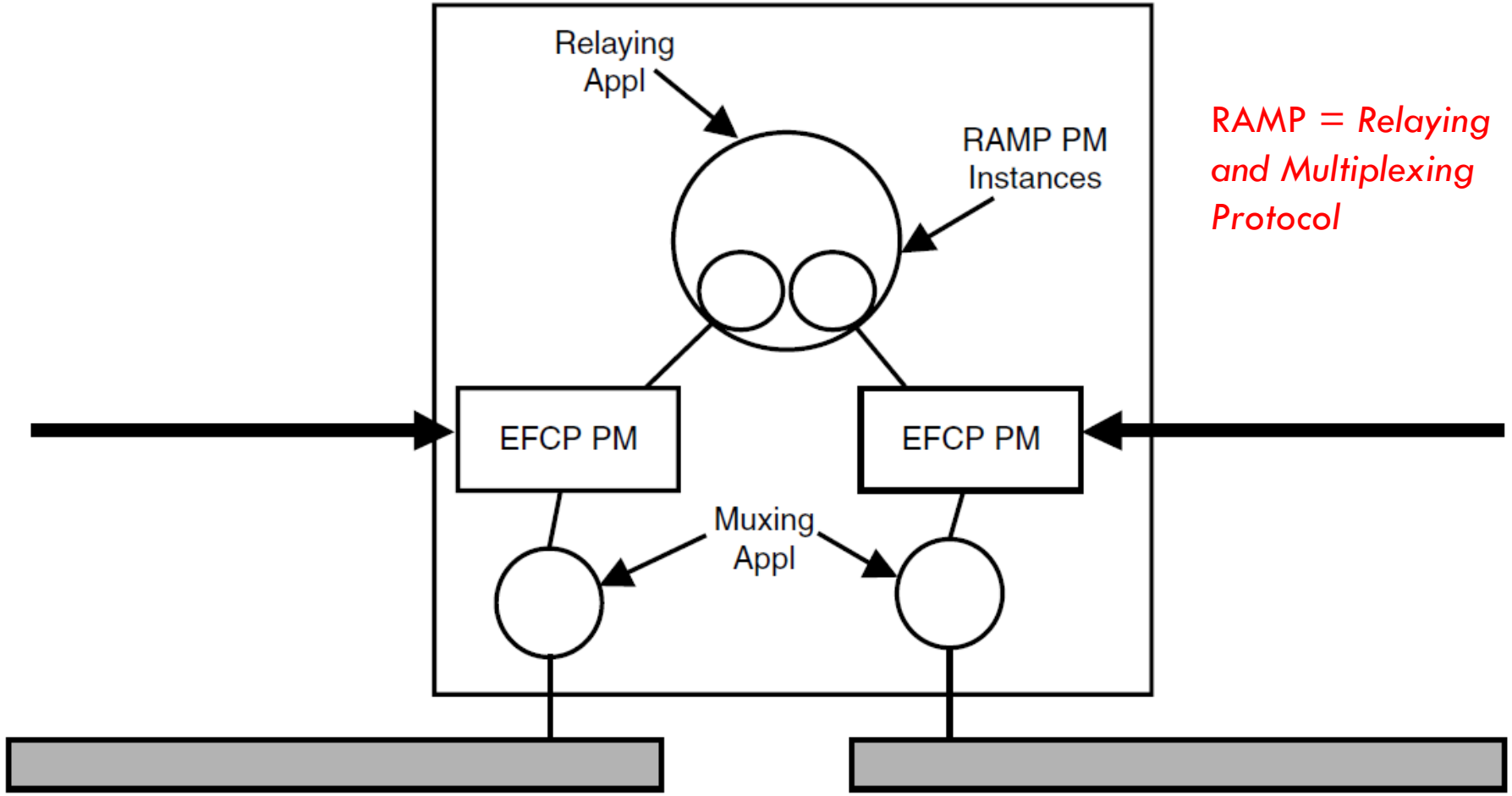
Novos Elementos

- Recurso IPC Distribuído (DIF):
  - Uma coleção de processos IPC cooperando para fornecer um serviço IPC distribuído às aplicações.
  - Em muitos casos, os processos IPC que compõem o DIF estão em sistemas diferentes.

Comunicação mais barata entre *N* Sistemas

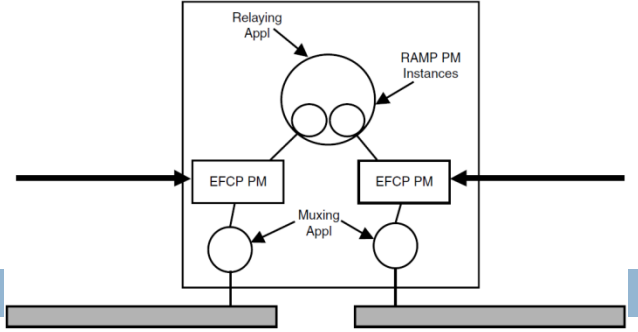


Sistema IPC Dedicado



RAMP = *Relaying and Multiplexing Protocol*

Sistema IPC Dedicado



- As aplicações de repasse recebem PDUs de instâncias do protocolo de aplicação, toma a decisão de repasse (escolhendo a instância do protocolo de aplicação através da qual encaminhará a PDU), e entrega a PDU a esta instância.
- Anteriormente a identificação da terminação local do "fio" era equivalente a identificar o destino. Agora não é mais suficiente.
- Precisamos atribuir identificadores a todos os processos IPC de forma não ambígua entre os membros do DIF.
  - Precisamos nomear os processos IPC destino (nos *hosts*) porque é para lá que estaremos enviando os dados.

Sistema IPC Dedicado

- Por que precisamos nomear as aplicações de repasse?
  - Pode haver mais do que um caminho para o próximo nó. Precisamos identificar interfaces e nós.
  - Os *relays* devem conhecer quem são os seus vizinhos para saber para onde enviar as PDUs.
  - Cada PDU precisará transportar o identificador do processo IPC destino.

Fase de Registro

- Anteriormente ela sempre foi uma coleção de procedimentos ad hoc que preferíamos ignorar.
- Mas, neste modelo ele é uma parte integral e natural da sua operação.
- O estabelecimento da aplicação fornecerá aos outros processos IPC o nome do processo solicitante junto com informações de controle de acesso que serão usadas para determinar se o solicitante está habilitado a entrar.

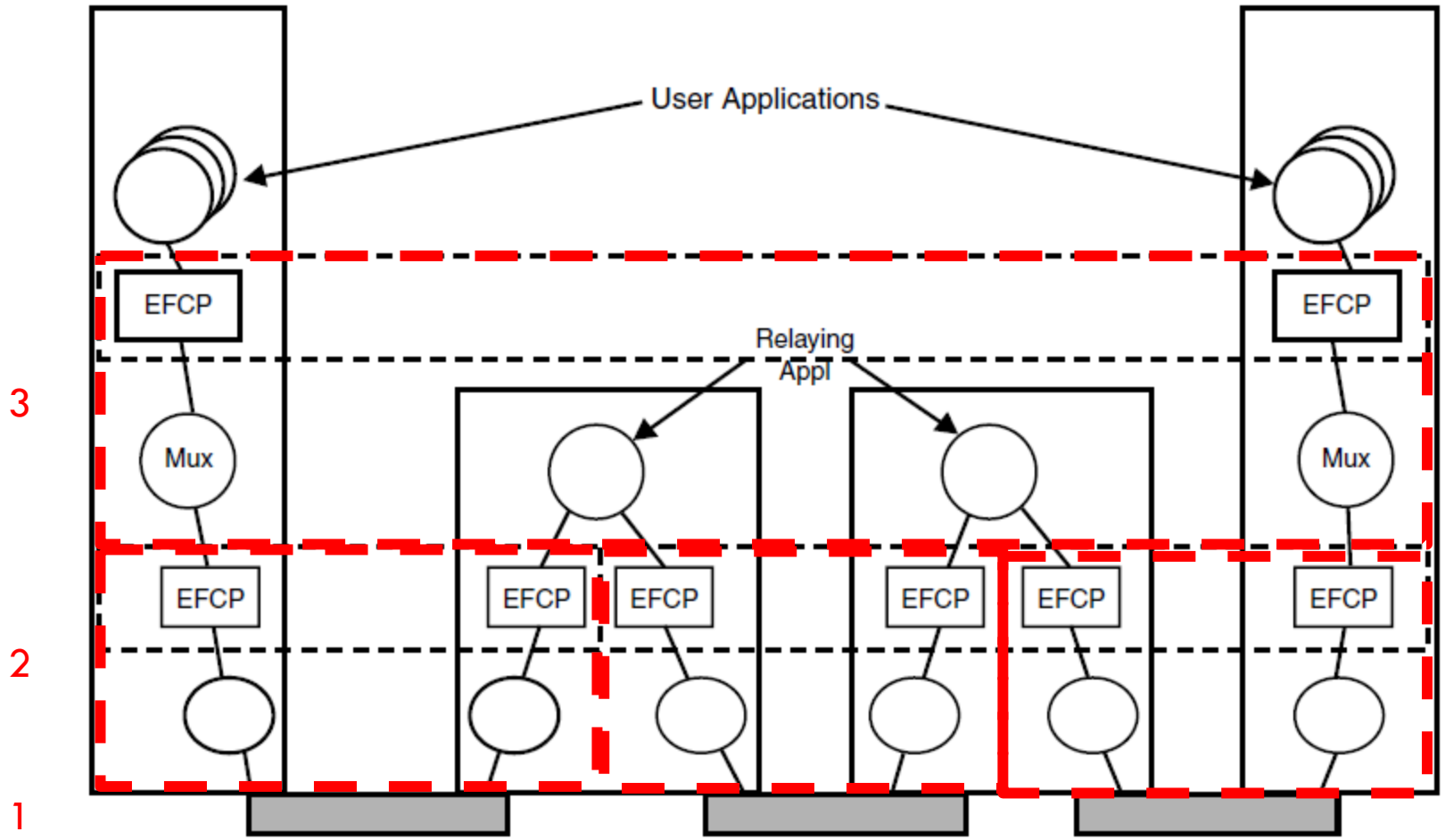
Endereços

- Será alocado pelo DIF um identificador para um novo processo IPC.
- Este identificador só precisa ser não ambíguo dentro do DIF e é usado para encaminhar PDUs entre os EFCPs.
- Este identificador é frequentemente chamado de *endereço*.
  - Endereços são identificadores internos ao DIF (escopo da camada)

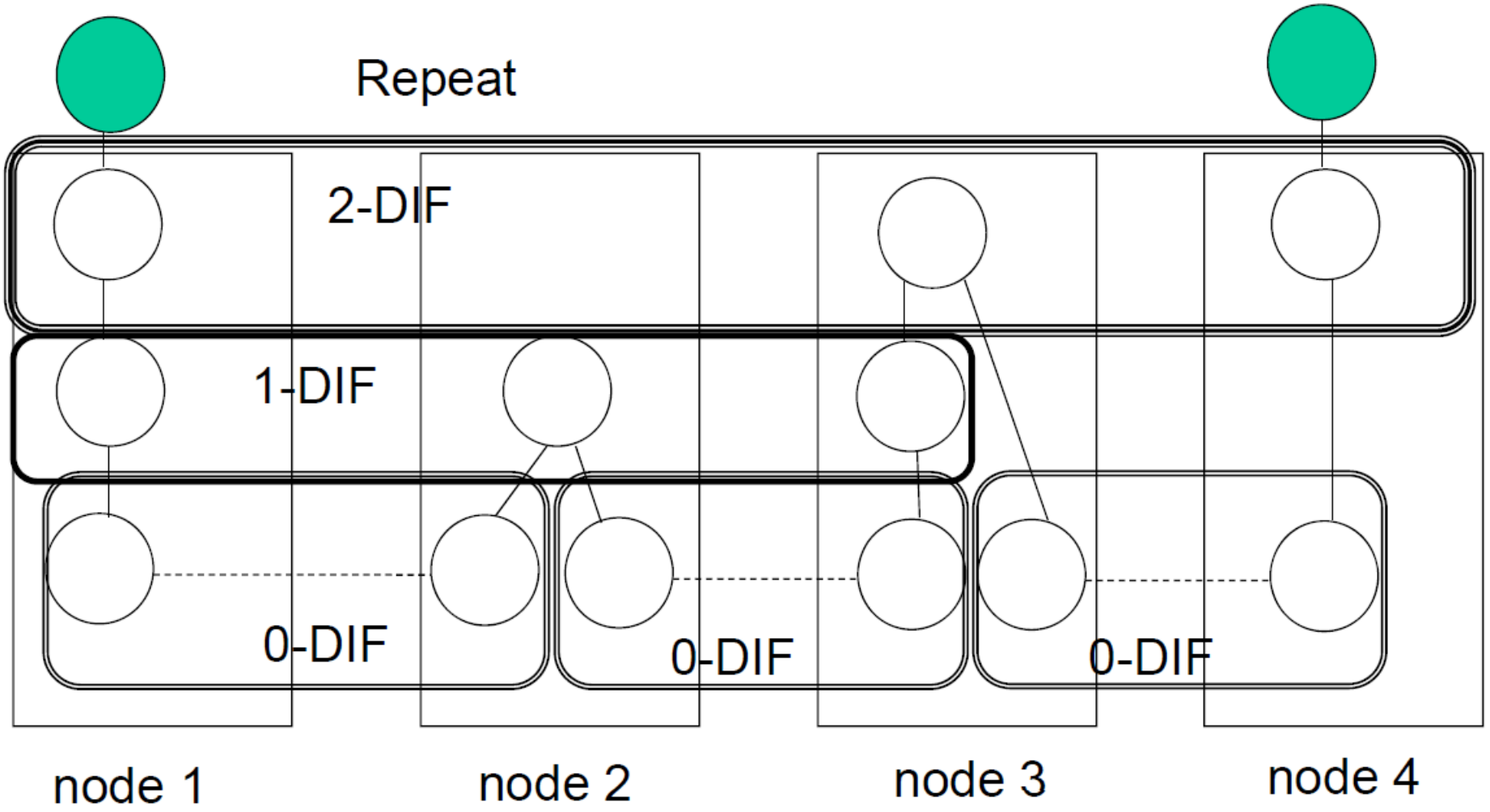
Pacotes fora de ordem e QoS

- Nesta configuração com múltiplos caminhos entre os mesmos pontos, as PDUs podem chegar fora de ordem e os *relays* podem experimentar congestionamento e descartar algumas PDUs.
- É recomendável que os sistemas origem e destino utilizem um protocolo EFCP para garantir a QoS e prover controle de fluxo entre eles.

Três Camadas de IPC



Três Camadas de IPC



Considerações

- Pode-se notar que desenhamos uma figura tradicional de uma arquitetura de redes em camadas.
  - Mas as linhas encontram-se em locais diferentes
- Criamos um sistema de três camadas de elementos repetidos, onde a unidade de repetição é um recurso IPC distribuído.

Novos Elementos

- Camada-(N)
  - Um recurso IPC distribuído em um dado nível
- Subsistema-(N):
  - A instanciação de uma camada em um sistema, um processo IPC
- Escopo de uma camada-(N):
  - O conjunto de todos os processos de aplicação capazes de se comunicar diretamente sem um repasse através de um DIF de ordem mais alta.

Conclusões Iniciais

- Redes são IPCs distribuídos e apenas IPC.
- Há vantagens em mantermos uma distinção clara entre a IPC e o seu gerenciamento.
  - Se não for uma parte direta do IPC, pertence a outro lugar.
- Estes processos IPC seguem a mesma estrutura de serem aplicações que usam uma IPC de suporte e assim por diante, até que a IPC seja um “fio” físico.

Conclusões Iniciais: Camadas

- Uma camada é um recurso IPC distribuído.
  - O recurso IPC é composto por processos IPC normalmente em sistemas distintos.
  - Mas os processos IPC são apenas aplicações usando a camada abaixo.
  - Padrões de demandas diferentes simplesmente levam a padrões de distribuição diferentes da informação e políticas diferentes.
  - A necessidade tanto para endereços de ponto de conexão (PoA) e endereços de nós é mais clara
    - E são relativos.
    - Os endereços de nó de um DIF são os endereços PoA do DIF abaixo.

Conclusões Iniciais: Relacionamento de Redes com SOs

- Este exercício revelou relações que não foram antecipadas.
- A relação com SOs, embora aparente desde o início, se revelou muito mais forte do que reconhecida anteriormente.
- Devemos tratar a coleção de subsistemas de uma mesma camada como um único sistema, algumas vezes muito mais fracamente acoplado do que os SOs, mas mesmo assim seria um sistema.

Conclusões Iniciais: Camadas Específicas

- Nesta seção foram justificadas as existências de pelo menos duas camadas:
  - Uma camada dedicada ao meio
  - Uma camada gerenciando os recursos físicos e
  - Uma camada gerenciando os recursos lógicos.
- A complexidade inerente é acomodada através da repetição da estrutura de DIFs, simplesmente pela criação de mais DIFs.

Conclusões Iniciais: Separação das Camadas

- A divisão entre TCP e IP foi feita no local correto:
  - O TCP é o protocolo que provê IPC básico de conexão/canal/fluxo para cada instância;
  - O IP contém a PCI para o repasse e multiplexação de uma aplicação distribuída.
- No entanto, esta mesma análise indica que foi um erro separar as camadas de rede e de transporte
  - Porque são partes integrantes de um *mesmo* recurso IPC distribuído!

Conclusões Iniciais: A Camada de Rede

- A camada de rede sempre foi confusa:
  - O nome implica que se trata apenas da rede e não dos *hosts* (e temos a tendência de tratá-la desta forma).
  - Mas, ela está presente também nos *hosts*. Precisa estar!
- O conceito de camada de rede é o último vestígio da mentalidade das contas em um cordão.
- A camada de rede deve sumir.
  - Ela nos faz pensar como cabeças de sino (*bellheads*).

Conclusões Iniciais: Fase de Registro

- A fase de registro (*enrollment*) nada mais é do que o estabelecimento normal de conexões de aplicações com a opção de autenticação e alguma inicialização, incluindo alocação de endereços.

Conclusões Iniciais: Endereços

- Os nomes necessários para o roteamento (i.e., endereços) não são apenas nomes de aplicações especializadas chamadas de máquinas de protocolos,
  - São nomes *internos* ao recurso IPC distribuído para a coordenação interna do IPC.
  - Eles são usados apenas pelos processos IPC para as suas próprias finalidades.
  - Qualquer outra propriedade que quisermos atribuir-lhes seria sobrecarregar a semântica do identificador.
    - Nenhuma outra aplicação tem a necessidade de conhecer a sua existência!

Conclusões Iniciais: Endereço

- Cada processo IPC possui um nome de aplicação *externo* e um endereço *interno*.
- Como se chega a algo antes que ele tenha um endereço?
  - Pelo seu nome externo da aplicação usando o IPC subjacente.

Conclusões Iniciais: Como não vimos isto?

- Quando o modelo original de camadas foi proposto as redes eram muito pequenas e lentas.
- A estrutura foi proposta após serem tentados apenas três ou quatro alternativas similares:
  - Em SOs haviam sido feitas 20 a 30 tentativas antes de chegarmos a um modelo comum.
- Falhamos em agir como cientistas questionando continuamente as nossas estruturas fundamentais.
  - *A Engenharia deve sempre fazer compromissos. Mas a engenharia é baseada na ciência que fornece os princípios do que é "certo", de modo que a engenharia possa fazer compromissos inteligentes. Mas, caímos na engenharia sem a ciência.*

Recapitulando

Para que servem as camadas?

- Duas finalidades para caixas pretas ou camadas:
  - Beneficiar o usuário ao esconder a complexidade e
  - Estruturação para organizar as funções para realizar aquele serviço.
- Para o usuário ou aplicação, a camada provê uma interface abstrata independente do hardware/meio para o IPC, puro e simples.
  - As comunicações com um processo no mesmo sistema ou num sistema diferente deveriam ser tão semelhante quanto possível (transparência).

Divisão em Camadas

- A divisão em camadas criando a ilusão desta transparência (invisibilidade) forneceu os meios para:
  - Decompor o problema
  - Gerenciar a complexidade e
  - Atingir a abstração (algumas vezes em estágios) para isolar a aplicação das características específicas do hardware (que são funções clássicas de SOs), mas focadas em IPC.

Padrões ou Invariantes da Organização dos Protocolos

- O escopo das “camadas” tende a crescer quando subimos.
- As mesmas funções de protocolo aparecem recorrentemente com diferentes políticas desde o meio físico até o que são consideradas as “aplicações” e isto não de forma arbitrária, mas repetindo os padrões.
- Há necessidade de uma PCI comum, pelo menos para endereçamento, tanto nas camadas mais baixas como nas aplicações.
- Conexão ou sem conexão podem ser obtidas através de um modelo comum.

Camada como um DIF

- A experiência mostrou que há frequentemente benefícios para a gerência de recursos se o protocolo de controle de erro e a tarefa de repasse e multiplexação puderem compartilhar informações sobre as condições que eles estão observando.
  - Colocar estes protocolos em camadas distintas impossibilitou este compartilhamento de informações.
  - Este modelo resolve isto e elimina alguma duplicação
  - A nossa "derivação" de rede como IPC mostra de forma conclusiva que eles são parte do mesmo recurso IPC e portanto devem ser parte da mesma camada.
    - *Uma camada é um recurso IPC distribuído.*

Alocação de Recursos

- O não reconhecimento da necessidade de protocolos do "tipo-IAP" e do seu papel na alocação de recursos, foi outro fator no não reconhecimento da estrutura.
- À medida que nos dirigimos para o *backbone* com a multiplexação de mais e mais tráfego, a largura de banda tende a crescer.
  - Os valores de largura de banda das aplicações num *host* raramente são tão grandes como as observadas em toda a rede.

O que são Camadas

- Todas estas são indicações do que deve:
  - Compor uma camada
  - Ser a organização da camada
- E, de certo modo, quantas camadas devem compor uma arquitetura.
- Mas, a finalidade das camadas não é tanto a de isolar funções diferentes e sim diferentes alcances do problema de alocação de recursos.
  - Isto nos leva a uma arquitetura escalável e que não requer novas construções para cada nova tecnologia ou capacitação que for inventada.

A Estrutura que estávamos buscando

- Aplicações operando sobre um recurso IPC, que depois se repete, subdividindo o problema
- Cada camada possui a mesma funcionalidade, mas esta funcionalidade é otimizada para lidar com um subconjunto da faixa de todo o problema de alocação de recursos.

A Arquitetura IPC de Rede

NIPCA – *The Network IPC Architecture*

atualmente

RINA – *Recursive InterNetwork Architecture*

Funções das Camadas

- Sempre vimos as diferentes camadas como se executassem diferentes funções
  - Com as limitações de recursos seria mais eficiente se as funções pudessem ser feitas num único lugar e não se repetissem.
  - Mas, observamos que as mesmas funções aparecem em diversas circunstâncias, especialmente por causa de camadas com escopos diferentes:
    - Roteamento faz parte da camada de rede, mas as LANs também roteiam.
    - CRC é necessário na camada de enlace, mas existem também na camada de transporte, etc.
  - Agora temos que subdividir o problema de alocação de recursos para sermos mais efetivos.

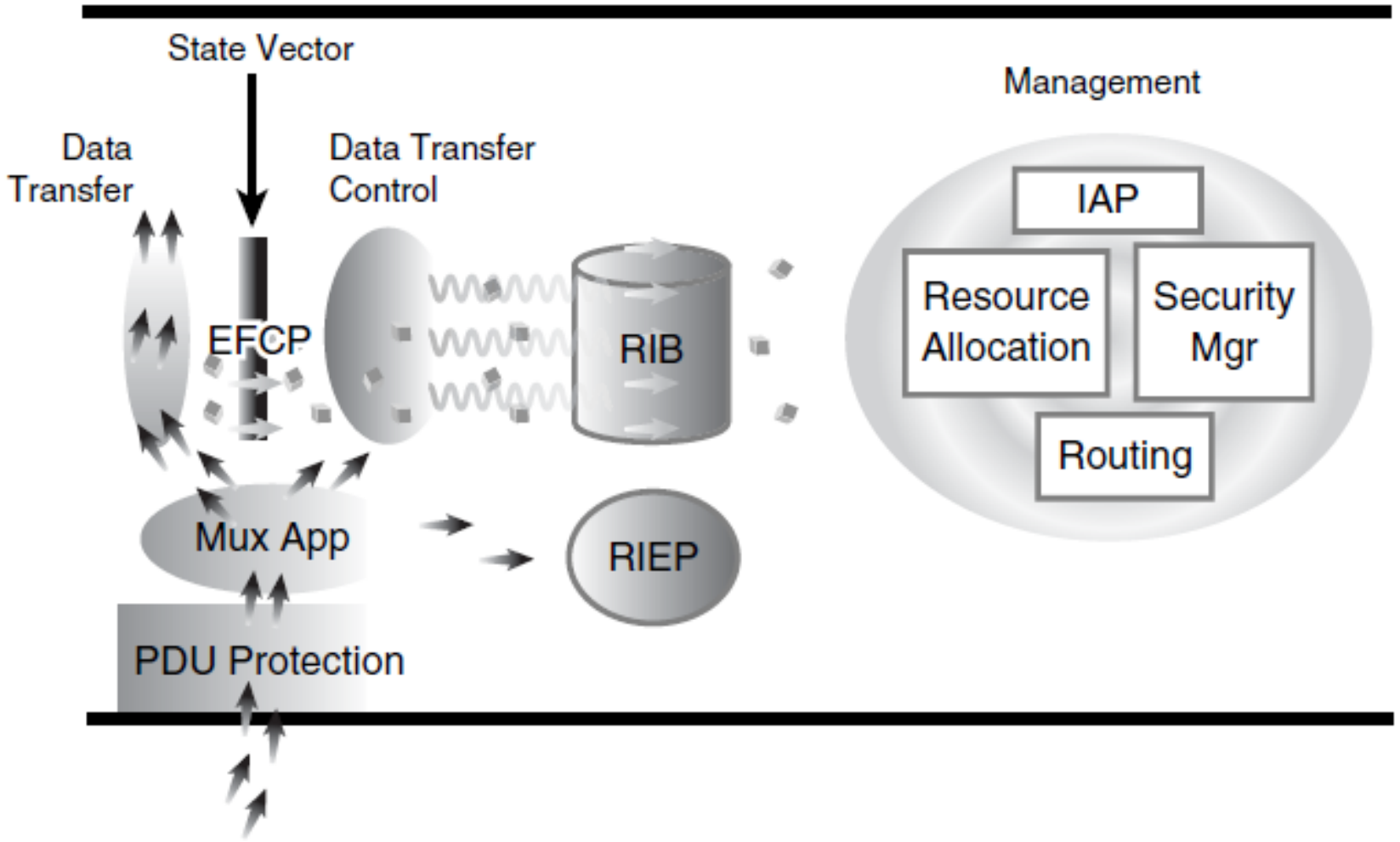
Mudança de Pensamento

- Todas as camadas têm a mesma funcionalidade.
- Elas diferem no escopo e na faixa de largura de banda e QoS que suportam.
- Portanto, embora os protocolos nestas diferentes camadas sejam os mesmos, eles têm diferentes *políticas* e, possivelmente, *sintaxe* adequada àquele escopo ou faixa de largura de banda e QoS.

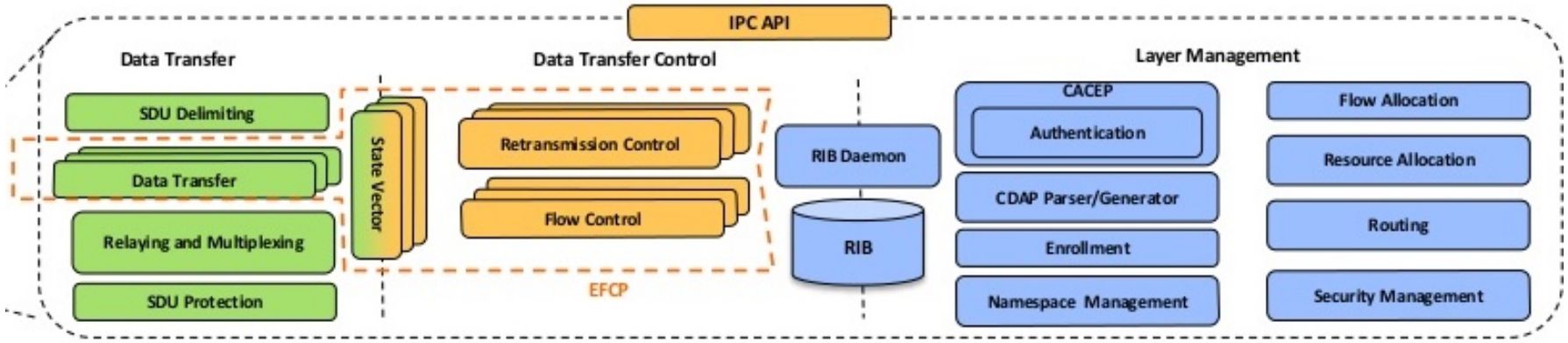
O que é uma camada

- Uma camada é uma coleção de processos IPC cooperantes.
- Os processos IPC são simplesmente aplicações com nomes de aplicação tirados de um espaço de nomes suficientemente não ambíguo para identificar todas as aplicações alcançáveis através do recurso distribuído de suporte.
- Um espaço de endereços interno ao recurso IPC distribuído é usado entre processos IPC para coordenar e gerenciar a IPC (i.e, a transferência de dados).

Elementos de uma Camada (antigo)

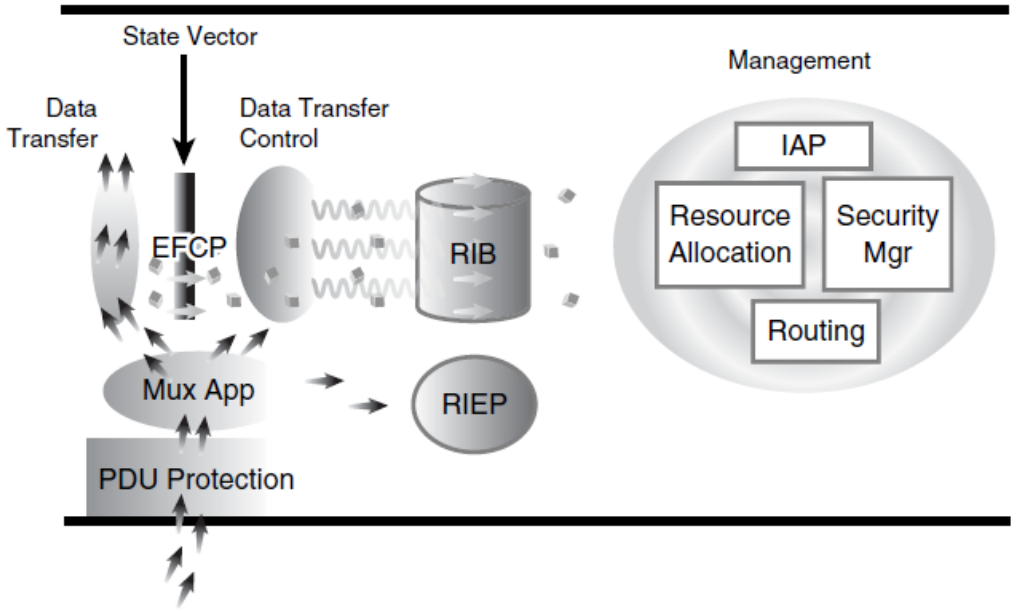


Elementos de uma Camada (novo)



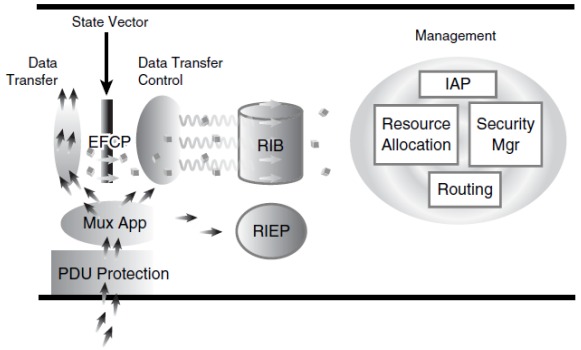
- Apenas dois protocolos:
  - EFCP = *Error and Flow Control Protocol*
    - Composto por dois "subprotocolos": DTP e DTCP
  - CDAP = *Common Distributed Application Protocol*
    - Inclui o CACEP *(Common Application Connection Establishment Phase)*

Elementos de uma Camada



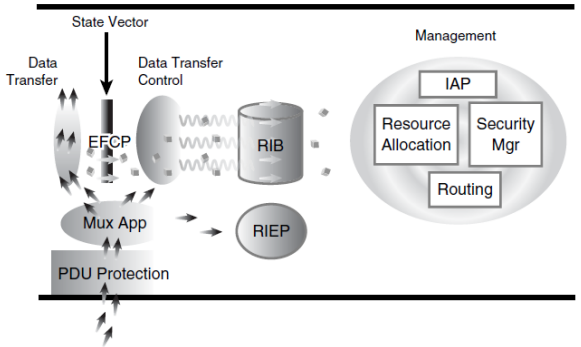
- Um protocolo de transferência de dados da IPC
- Um protocolo de controle da transferência de dados da IPC
- O gerenciamento da IPC

Elementos de uma Camada



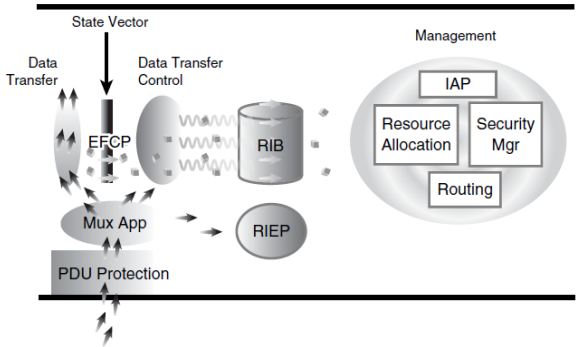
- Um protocolo de transferência de dados da IPC:
  - Uma função de "escalonamento" que acrescenta a PCI de transferência de dados comum e provê multiplexação e repasse com quaisquer políticas necessárias para gerenciar os buffers e filas e
  - Uma tarefa de transferência de dados por fluxo que apenas lida com PDUs de Transferência (i.e., mecanismos fortemente acoplados).

Elementos de uma Camada



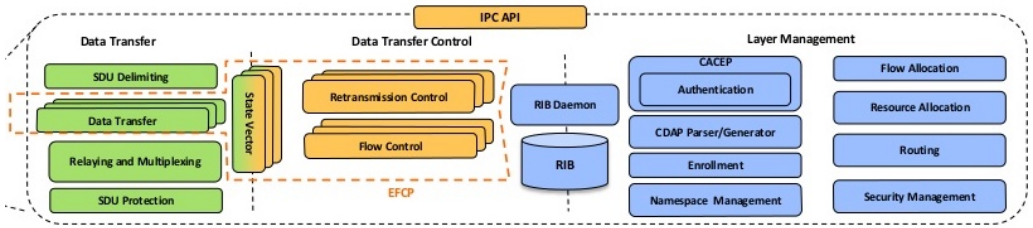
- Um protocolo de controle da transferência de dados da IPC:
  - Para fornecer todos os mecanismos fracamente acoplados necessários com as políticas apropriadas instanciados com base nos requisitos de cada pedido de alocação pela aplicação solicitante.
  - O EFCP estará presente na medida em que a diferença entre a QoS solicitada pelo usuário acima e a QoS fornecida pela camada abaixo sejam suficientemente diferentes para necessitar de medidas adicionais.

Elementos de uma Camada



- O gerenciamento da IPC (velho):
  - Um protocolo de troca de informações do recurso (RIEP – *Resource Information Exchange Protocol*) que fornece serviços de consulta e atualização de uma MIB (*Management Information Base*) para um conjunto de tarefas de gerenciamento de recursos para roteamento, gerenciamento de recursos de segurança, alocação de endereço, etc., necessários para gerenciar e manter a IPC distribuída.

Elementos de uma Camada



- O gerenciamento da IPC (novo):
  - O estado de um processo IPC é modelado como um conjunto de objetos armazenados na *RINA Information Base* (RIB).
  - O *RIB Daemon* provê todas as funções de gerenciamento da camada:
    - *Enrollment, namespace management, flow allocation, resource allocation, security coordination, etc.*
  - A troca de informações é realizada através do CDAP

Informações Necessárias para a Comunicação

- Para qualquer comunicação, a única informação que uma aplicação de usuário possui ou necessita é:
  - a sua port-id local e
  - o nome da aplicação destino
- Ela nunca necessita nem tem acesso a:
  - Endereços conhecidos
  - Portas bem conhecidas
  - Etc.

Segurança na Comunicação

- Com recursão, a infraestrutura de rede ficará impenetrável a ataques pelos *hosts*, porque os *hosts* simplesmente não pode endereçá-la.
- A confiança necessária para um recurso IPC distribuído de um IPC de suporte é muito pouca:
  - Apenas que a comunicação de suporte tentará entregar as PDUs a algo.
  - Não depende de mais nada para isto.

Segurança na Comunicação

- É responsabilidade do DIF durante o registro determinar que outros processos IPC são membros válidos e sua responsabilidade em proteger sua comunicação contra modificações ou escutas.
- Para se juntar a um DIF, o novo processo IPC deve ser autenticado e alocado um endereço.
  - A força desta autenticação e o modo de cumprimento das políticas irá depender do DIF.

Organizando as Camadas

Uso de uma Arquitetura

- Duas formas de uso:
  - Como guia para que estruturas existentes de protocolos ganhem uma melhor compreensão daquilo que foi feito e porque e como fazer melhor no futuro; e
  - Para definir um novo conjunto de protocolos e arquiteturas para resolver problemas.
- Nós iremos considerar esta arquitetura nestes dois sentidos acima.

Organização de Múltiplas Camadas

- Quantas são as camadas?
  - Devemos ter pelo menos uma camada de IPC
    - Isto corresponde a aplicações rodando sobre o que foi tradicionalmente chamado de camada de enlace de dados ou LAN.
    - Ok, mas o escopo é bem restrito:
      - Com o crescimento, as LANs ficam mais difíceis de ser gerenciadas
    - Tivemos dificuldade em lidar com congestionamento e alocação de recursos.
      - Isto não é uma surpresa pois tentamos gerenciar larguras de banda que ampliaram-se por seis ordens de magnitude com os mesmos protocolos e políticas!
      - E a única solução que demos foi usar mais hardware!

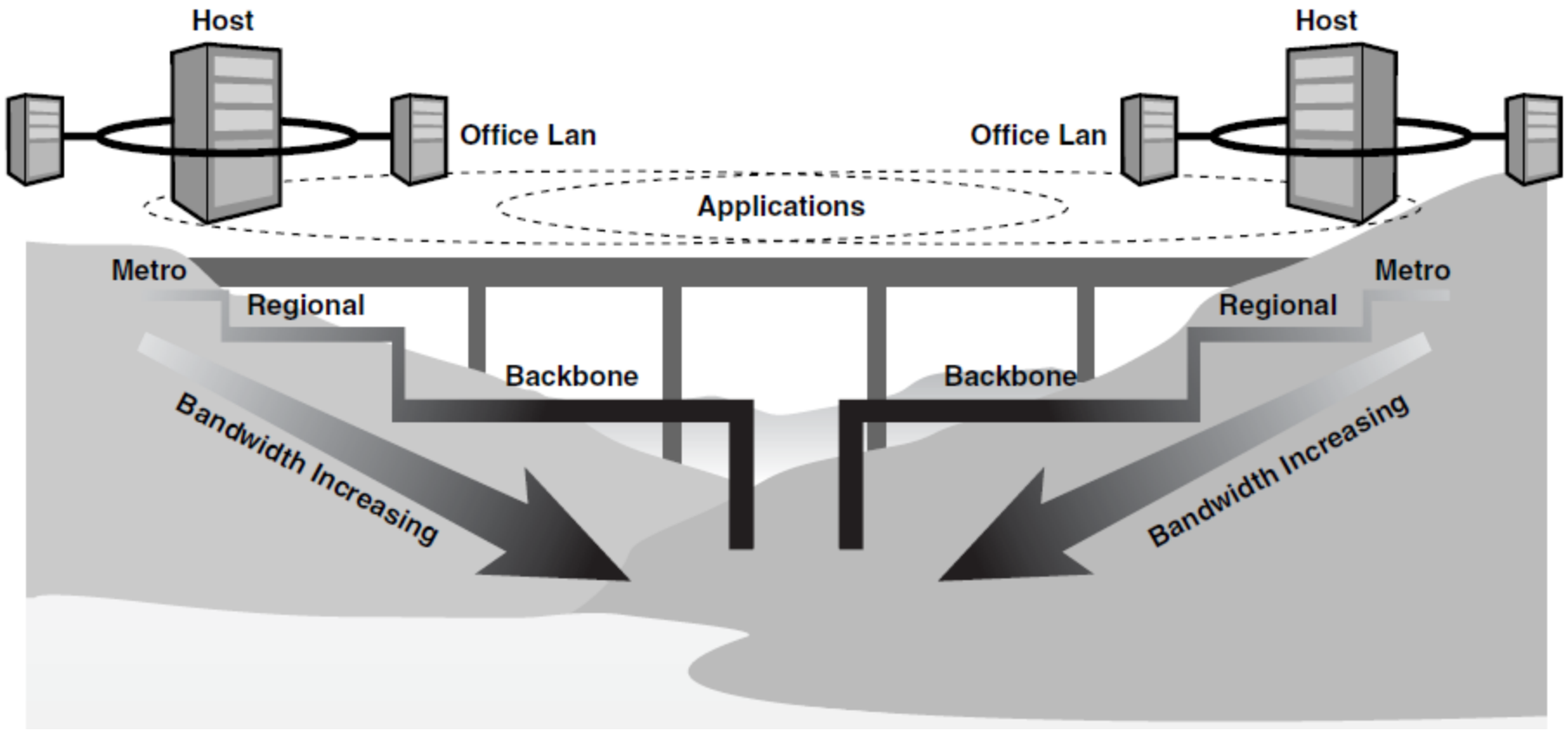
Problemas de Escala

- Apesar de conseguirmos montar grandes redes LAN com *bridges*, estas redes são "difíceis" de ser gerenciadas. Fica mais fácil com roteadores.
  - Mas, as vantagens advêm da decomposição causada pelo repasse numa camada mais alta e que permite que as funções de gerenciamento e os efeitos das falhas estejam dentro de domínios de tamanho gerenciáveis onde possam ser aplicadas políticas de gerenciamento similares.
- Mais camadas melhorarão a capacidade de gerenciar recursos.
- O gerenciamento de recursos pode ser muito mais efetivo e menos subótimo se for aplicado num escopo menor.

Problemas de Escala

- Tunelamento trata os sintomas mas não a substância do problema.
- As questões de escalabilidade que temos com a arquitetura de redes não têm a ver com a transferência de dados e sim com o gerenciamento.
- Portanto, se ao invés de usarmos recursivamente o protocolo de repasse, usarmos a *camada*, teríamos o aparato necessário para gerenciar os recursos da própria recursão.
- Uma camada então gerenciaria uma dada faixa de largura de banda e parâmetros de QoS, numa faixa gerenciável e não numa faixa de seis ordens de magnitude.

Metáfora da Colina e do Vale



Número de Camadas

- Quanto mais larga for a faixa de largura de banda entre as aplicações e o *backbone*, maior será o número de camadas.
- Quanto menos larga, menor o número de camadas.
- Estas camadas intermediárias são criadas em incrementos de largura de banda
  - Com faixas a serem determinadas experimentalmente
- Dados a partir de aplicações com pouca largura de banda passarão por mais camadas do que os dados de aplicações que requeiram uma maior largura de banda.

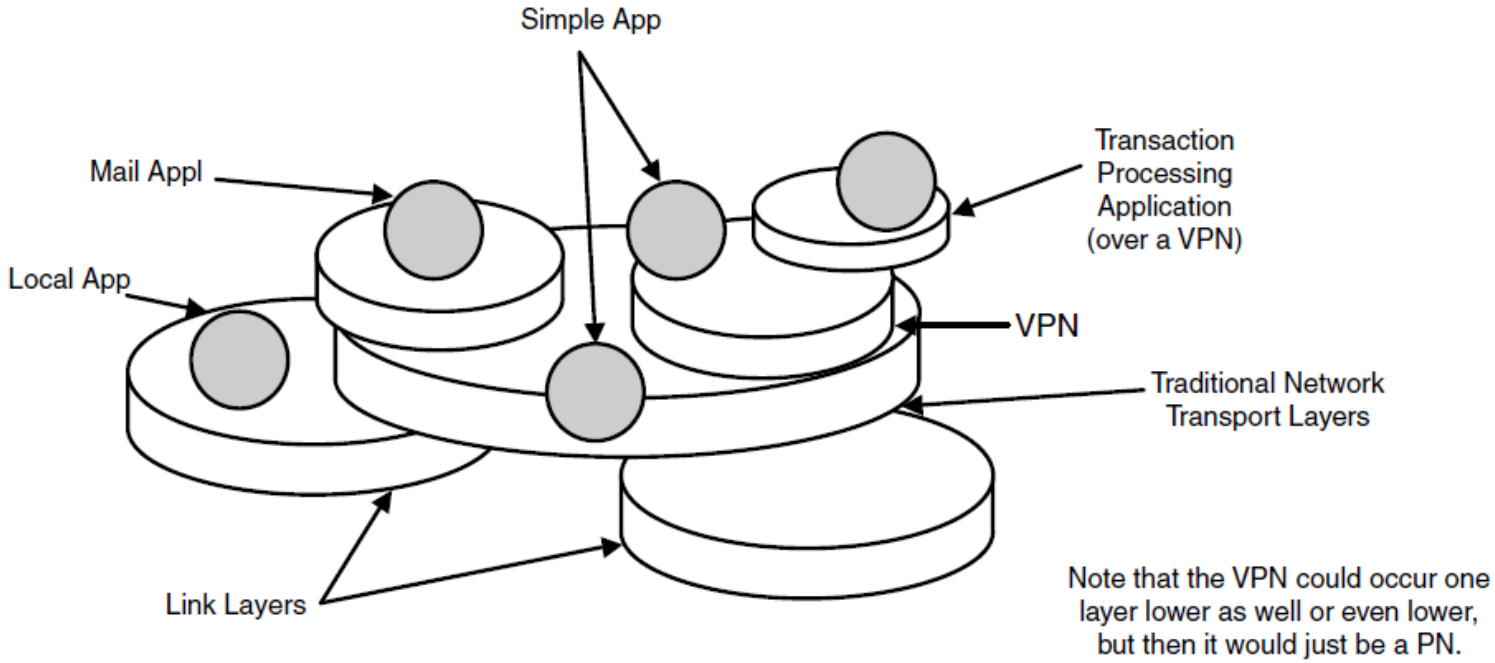
Número de Camadas

- Uma camada pode reunir um conjunto de fluxos da camada superior em uma faixa particular de largura de banda e multiplexa-las para que juntas formem um fluxo na faixa de largura de banda da sua camada.
  - Concatenando as PDUs enquanto elas se movem até o núcleo da rede reduzindo assim o *overhead* de comutação.
- Algumas destas camadas intermediárias podem:
  - ser baseadas em um provedor e representar um escopo gerenciado por um único provedor
  - Enquanto uma rede pública mais ampla ou VPN flutuam acima de sub-redes do provedor.

Resumindo

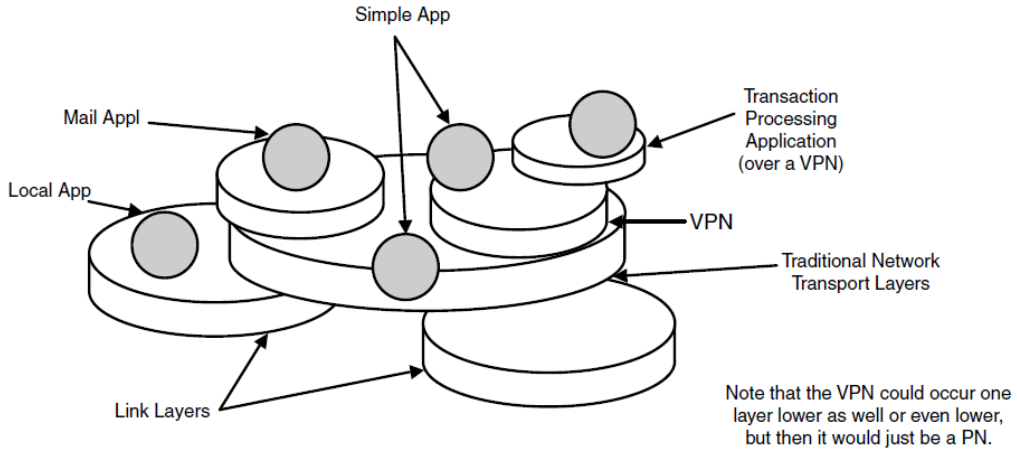
Quantas camadas existem?	Tantas quanto você precisar
Quantos tipos de camadas existem?	Uma, um recurso IPC distribuído
O que faz uma camada?	Transfere dados para um conjunto de usuários dentro de um recurso IPC distribuído com uma dada QoS

Exemplo de Empilhamento de Camadas



- Uma arquitetura de rede consiste de aplicações e é suportada por um recurso IPC configurado apropriadamente acima de tantos recursos IPC distribuídos (DIFs) quantos forem necessários.

A aplicação é a camada mais alta?



- Sim para aquela aplicação.
- Podem haver aplicações que necessitam de um escopo maior e que estarão acima desta aplicação.
- Podem haver aplicações que necessitam de um escopo menor e operam em camadas abaixo desta.
- Qual o escopo que uma aplicação deve ter?
  - O suficiente para se comunicar com todos que necessite.

Onde estão as aplicações de gerenciamento de redes?

- Acima de qualquer camada que tenha escopo suficiente para acessar os *sistemas* (não necessariamente as camadas) a serem gerenciados.
  - Poderíamos ser tentados a fazer coincidir os domínios de gerenciamento e os escopos das camadas onde as aplicações executam.
- Camadas mais "públicas" teriam controles mais frouxos, incluindo segurança, enquanto que aquelas mais privadas teriam controles mais rígidos.

Conclusões

O Nosso Modelo

- Parece que alcançamos os objetivos das propriedades de Newton para uma boa teoria:
  - O modelo requer poucos conceitos simples
    - Para gerar as características das redes como as conhecemos
  - Mostra quantos aspectos de rede são acomodados como consequência da estrutura
    - E que hoje necessitam de mecanismos completos especializados.

Vantagens do Modelo

- Resolve os problemas sem a criação de uma miríade de mecanismos, protocolos, remendos, e sem aumentar o "número de peças".
- A estrutura deste modelo cria um comportamento mais previsível e que resolve problemas que não tinham sido considerados no início.
- Muitos problemas atuais de rede são facilmente endereçados em muitos casos sem um esforço adicional e em outros apenas com uma escolha cuidadosa da política.
- O modelo IPC e a compartimentação que ele cria também traz benefícios para a segurança.

Conclusões

- Não existe uma arquitetura das camadas superiores, existe apenas a distinção entre a aplicação e a máquina de protocolo de aplicação.
- Nunca há necessidade de um protocolo-(N-1) identificar o protocolo-(N).
  - O equivalente de port-id é tudo o que as camadas adjacentes precisam ter em comum.