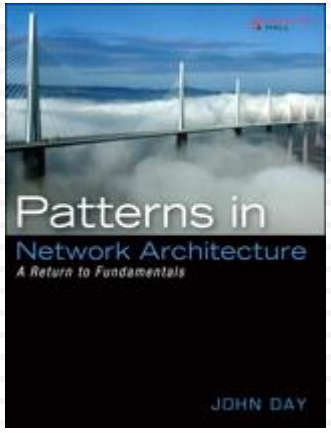
Padrões em Protocolos

Capítulo 3

Patterns in Network Architecture



Introdução

- Neste capítulo, começamos a encontrar padrões na arquitetura de redes.
- Padrões que nos ajudam a fazer previsões ou que proveem novas compreensões (*insights*).
- A tarefa é mais difícil pois "construímos o que medimos". Ou seja, é difícil distinguir entre que padrões são fundamentais e não artefatos daquilo que construímos.
- Encontrar o problema no núcleo de um conjunto de problemas nem sempre é óbvio.

O problema das balas de canhão

- Um dos mais problemas do século XVI era prever onde as balas de canhão iriam cair.
- Ao invés de propor um projeto elaborado e caro para explorar exaustivamente o comportamento delas com uma coleção altamente instrumentada de canhões de diversos fabricantes calibres e quantidade de pólvora e a partir destes dados determinar as equações que iriam prever o caminho das balas de canhão, Galileu teve a compreensão de que a resposta não estaria em disparar canhões mas em pensar muito em uma abstração simples que estava no núcleo do problema.

O problema das balas de canhão

- A chave para Galileu foi romper com Aristóteles e imaginar algo que nunca tinha sido visto ou que ninguém teria nenhuma razão para acreditar que existisse: *movimento sem atrito*.
- E a partir daí formular o que hoje conhecemos como a primeira lei do movimento: "Um corpo em repouso ou em movimento tenderá a permanecer em repouso ou em movimento...".
- Imaginem o quão absurdo e idealista esta construção deve ter parecido para os seus colegas.
- Devemos também procurar o modelo no núcleo do nosso problema e encontrar os conceitos que reúne tudo. E como Galileu iremos descobrir que pensar bastante é mais produtivo e menos caro!

Orientações dos princípios de Newton

- A natureza é essencialmente simples. Portanto, não devemos introduzir mais hipóteses do que aquelas suficientes e necessárias para explicar os fatos observados. [Regra da simplicidade]
- Portanto, até onde for possível, devemos atribuir efeitos similares às mesmas causas. [Princípio da uniformidade da natureza]
- As propriedades comuns a todos os corpos ao alcance dos nossos experimentos são assumidos (ao menos tentativamente) como pertencentes a todos os corpos em geral.
- Proposições em ciência obtidas através de indução devem ser consideradas exata ou aproximadamente verdadeiras até que fenômenos ou experimentos mostrem que devem ser corrigidas ou estejam sujeitas a exceções.

Advertências

- Assumiremos que o que os outros fizeram teve um bom motivo e oferecem dicas para padrões que podem ter permanecido obscuros.
- Mas, teremos que pensar bastante!
- Devemos colocar de lado noções preconcebidas para vermos aonde nos leva um novo caminho; e algumas coisas que pensávamos que fossem fatos, eram artefatos da nossa forma antiga de pensar.
- Não é que as formas antigas estavam erradas. Foram necessárias em grande parte para que progredíssemos.
- Tínhamos que ver como o problema se comportava para ter uma melhor compreensão dos princípios que o governava.

Grande Guerra Religiosa de Redes

- A guerra gira em torno de dois tópicos que não são apenas técnicos mas também históricos, políticos e, ainda pior, econômicos!
- Conflito entre os dois maiores paradigmas arquiteturais:
  - "Contas em um cordão" (*Beads-on-a-string*) e camadas
  - Conexão e Sem conexão

Conexões e Sem Conexões

- Normalmente, tanto uma como outra fazem sentido. Afinal, as arquiteturas que suportam sem conexões também têm conexões.
- Devemos entender quando uma ou outra é preferida.
- Eu acreditava que deveria haver algo que não estávamos vendo: um modelo no qual conexões e sem conexões fossem ambas casos extremos (degenerados).

Os dois principais paradigmas de arquiteturas

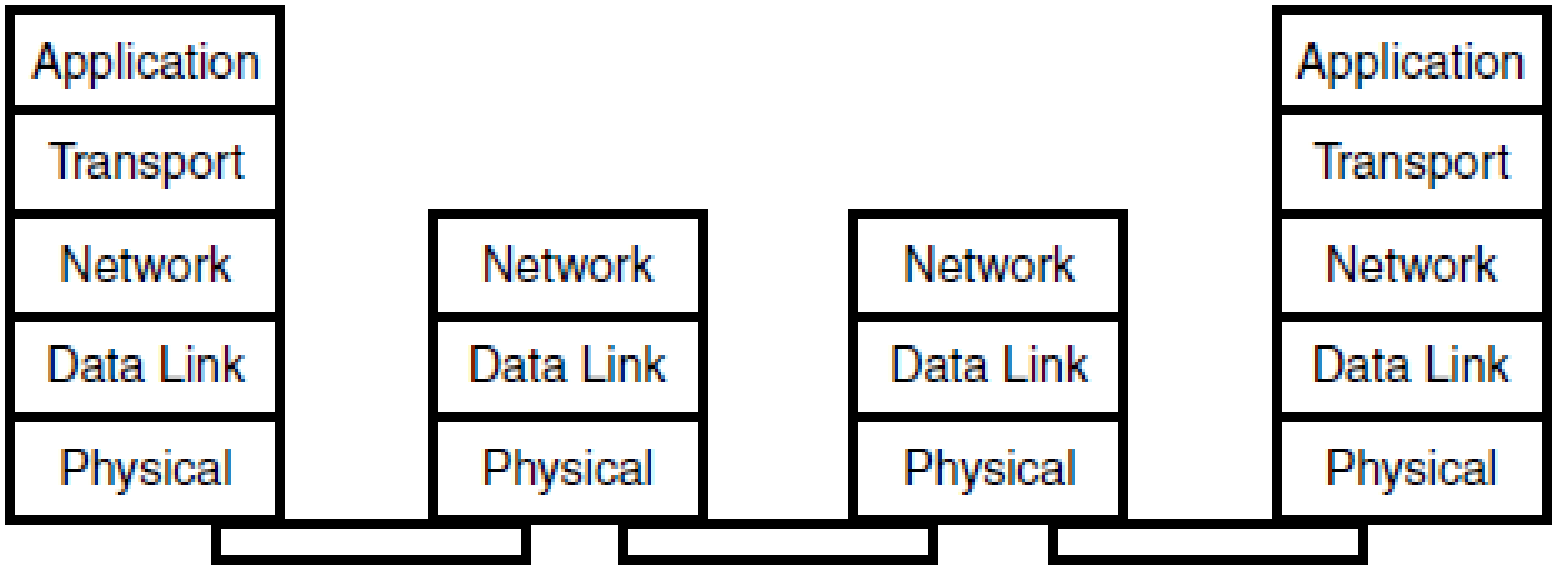
O Modelo de Camadas

- As principais redes de computadores (ARPANET, CYCLADES e NPLnet) foram construídas por especialistas em computação (em particular, sistemas operacionais) e não especialistas em comunicações.
- Em 1970 a engenharia de software ainda era muito jovem e os sistemas operacionais eram os programas mais complicados da época.
- Não é, portanto, uma surpresa que o artigo de Dijkstra no projeto elegante e simples do sistema operacional THE e o Multics tenham influenciado as tentativas iniciais de encontrar uma estrutura para as novas redes.
- Isto combinado com a justificativa de que a ARPANET era uma rede de compartilhamento de recursos serviu para receber uma grande influência dos sistemas operacionais.
- As primeiras aplicações foram modeladas como provendo as funções principais de um sistema operacional numa rede.

Finalidade das camadas de Dijkstra

- As mesmas de qualquer abordagem de “caixa preta”:
  - Prover uma abstração das funções de baixo e isolar os usuários das funções mais específicas de como a função era implementada ou detalhes específicos do hardware.
  - Camadas mais altas proviam maiores abstrações.
  - Isto permitia a modificação das funções de uma camada sem afetar as camadas acima ou abaixo.
  - Por outro lado as restrições de recursos levaram Dijkstra a acreditar que não havia motivo para que as funções fossem repetidas.
  - Modelo adequado especialmente para redes de compartilhamento de recursos distribuídos.

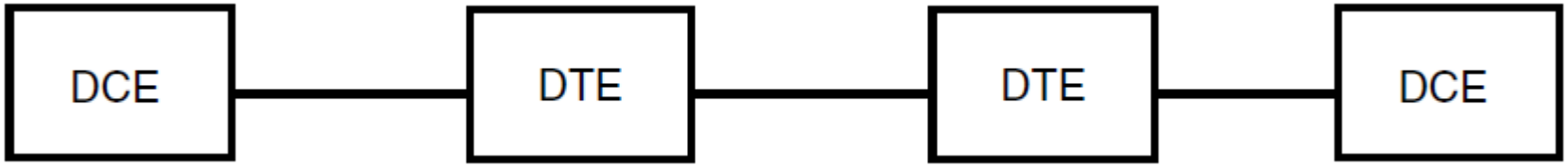
Arquitetura de Cinco Camadas (1974)



Arquitetura de Cinco Camadas (1974)

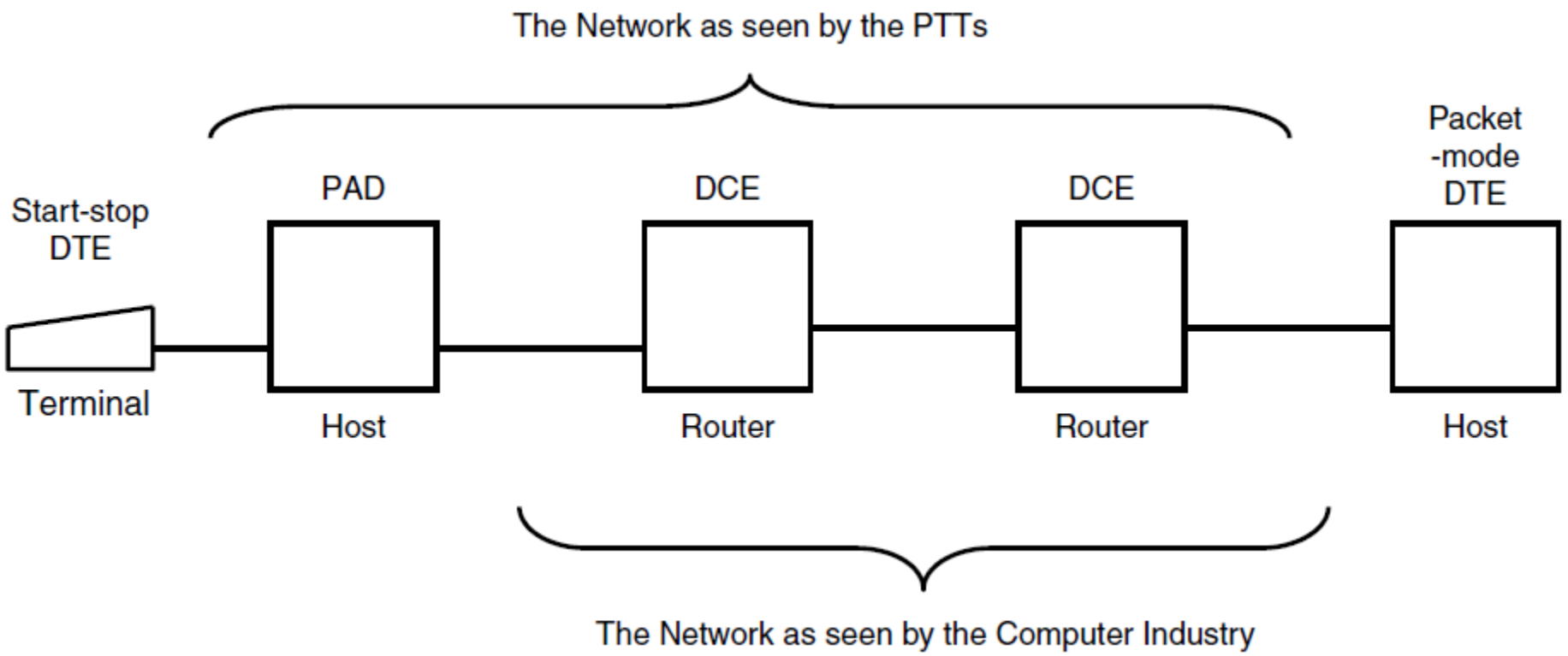
1. Uma camada física consistindo dos fios conectando os computadores
2. Uma camada de enlace fornecendo controle de erro e de fluxo nas linhas que conectam os computadores
3. Uma camada de repasse para encaminhar o tráfego para o destino correto
4. Uma camada de transporte responsável pelos controles de erro fim-a-fim e de fluxo.
5. Uma camada de aplicação para realizar o trabalho realmente.

O Modelo de "Contas em um Cordão"



- Reflete o ambiente único ocupado pelas empresas de telecomunicações.
- Principal objetivo: definir quem possui o que.
- Primeira diferença com o modelo de camada: definição de interface.
  - Camadas: interface entre duas camadas internas ao sistema
  - Contas em um cordão: interface entre duas caixas.

Visões diferentes da mesma coisa



Visões diferentes

- Empresas de telecomunicações:
  - Viram uma oportunidade para redes de valor adicionado
  - Nos EUA viram a oportunidade de entrar no mercado de computação.
  - Mas não gostaram da ideia de competição ou de empresas criando suas próprias redes.
  - Não gostaram do modelo de camadas pois ficariam relegados a um mercado de *commodities*.
- Tendência: "contas em faixas" (Internet atual)

Debate Sem Conexão/Conexão

- "Contas em um cordão" é naturalmente associada com a abordagem orientada a conexões.
- Modelo de camadas é geralmente associado a tecnologias sem conexão
- Mas há também mistura das duas...

História

- ARPANET baseada em ideias de Paul Baran, foi precursora de redes orientadas a conexões como a X.25 dos anos 1970s:
  - Em caso de falhas os pacotes seriam automaticamente reroteados ao redor da falha, como erros corrigidos a cada etapa.
- CYCLADES (França): dado que os hosts não iriam nunca confiar na rede e iriam verificar os erros, a rede não precisaria ser perfeitamente confiável e, portanto, poderia ser menos cara e mais econômica. Rede datagrama ou sem conexão.
- Logo após, a ARPANET adicionou um modo sem conexão.

Guerra religiosa

- Sem conexões (Datagramas):
  - A comunidade de ciência da computação abraçou a elegância e simplicidade do conceito (e sua compatibilidade com computadores)
- A guerra sem conexão/conexão tem sido uma Guerra de Trinta Anos (quase tão ruim como a primeira).
- As PTTs foram incapazes de conceber (ao menos, como muitos debateram em muitas reuniões) a existência de comunicação sem uma conexão.
- Os "protestantes" da comunidade Internet, concluíram que tudo deveria ser sem conexões...

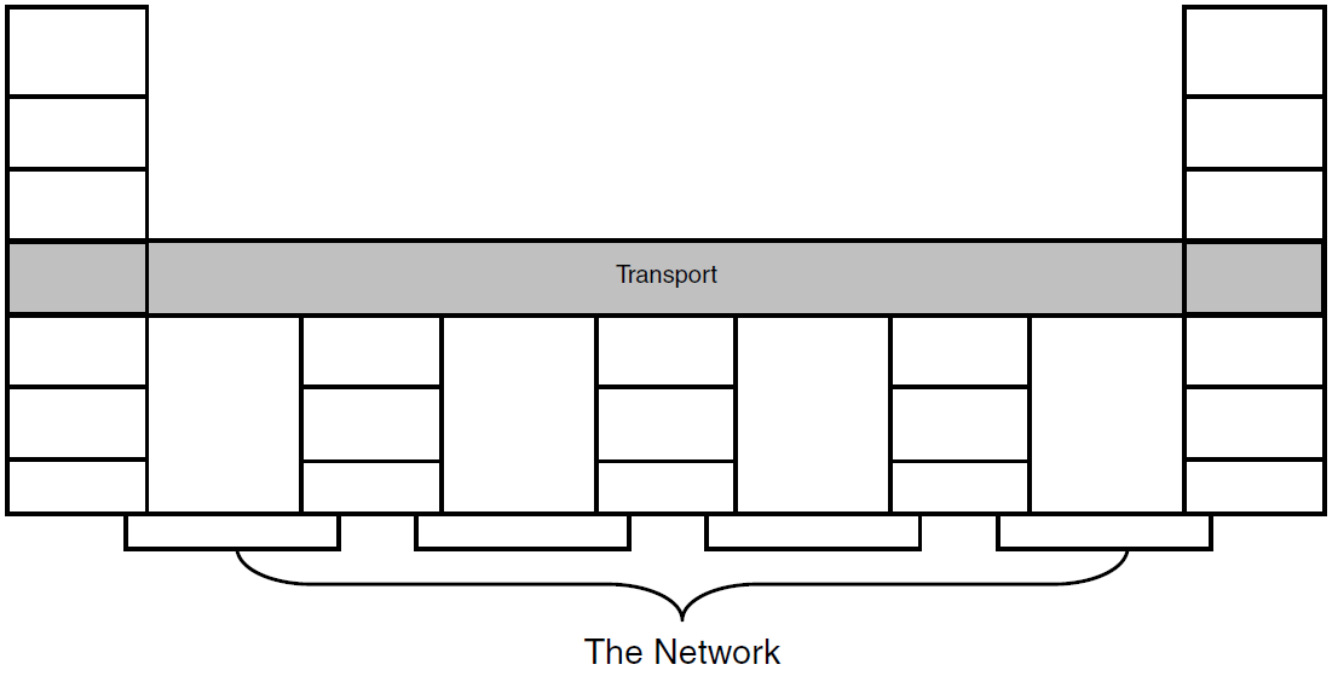
Primeira Batalha: X.25

- X.25: as tropas sem conexão tardaram em descobrir este avanço feito pelas companhias de telefonia em redes de dados.
- Não estavam preparados para as táticas do processo de padronização do CCITT (*Comité Consultatif International de Télegraphie et Téléphonie*) [Hoje, ITU-T]
- X.25 definia uma interface para a rede e não o modo de funcionamento interno da rede.
- Serviço datagrama:
  - *Fast Select*: pacote que abria, transferia os dados e fechava a conexão.
  - Apesar de ter sido incluído foi raramente usado.

Primeira Batalha: X.25

- Primeiro foco do debate X.25:
  - O controle de erro etapa por etapa seria tão confiável como um controle de erro fim-a-fim ou um protocolo de transporte como o TCP seria sempre necessário?
  - Algum diretor de TI que quisesse manter o seu emprego iria simplesmente assumir que a rede nunca perderia nada? Claro que não.
  - Portanto, se os hosts estavam fazendo verificação de erro fim-a-fim, a rede poderia fazer menos verificação de erros.

Segundo Problema (para as empresas de telecomunicações)



- A camada de transporte isola a rede das aplicações com grandes margens de lucro.

Segunda Batalha: OSI

- Trabalho do OSI (*Open Systems Interconnection*)
- A delegação europeia capitulou sob a influência das PTTs.
- As empresas de computação da Europa não quiseram adotar nada feito nos EUA, para não dar-lhes uma vantagem no mercado.
- Os europeus tinham um bom protocolo de transporte (desenvolvido para a CYCLADES), que havia sido selecionado pela IFIP como uma proposta para um protocolo de transporte fim-a-fim internacional que foi aceito como a base para o protocolo de transporte OSI (Classe 4).

Segunda Batalha: OSI

- Os europeus insistiram no X.25 como o protocolo de redes orientado a conexões.
- Após uma batalha amarga, os EUA conseguiram inserir o paradigma sem conexão na arquitetura, mas tiveram que aceitar restrições que impediam a interoperação entre elas.
- Apesar de não ser um problema tão sério, levou os EUA a desenvolver roteamento e protocolos de transferência de dados sem conexão.

Resultados

- O debate conexão/sem conexão entre EUA e Europa; e
- As divisões dentro dos EUA para dominar a direção do OSI (DEC, IBM, COS, NIST, MAP, etc.)
- Terminaram autodestruindo o OSI.

- A Internet, uma rede de pesquisa, se tornou uma rede de produção sem nunca ter se tornado um produto!

Encontrando uma Síntese: A Parte Fácil

- Qualquer solução deveria resolver dois problemas:
  1. Uma abordagem unificada do serviço. (A visão externa de "caixa preta" deveria ser a mesma.)
  2. Uma abordagem unificada para a função, um único mecanismo que incluísse os dois extremos.

Modelo Unificado

- Para que a comunicação sem conexão seja possível o transmissor deve ter alguma razão para acreditar que haverá uma instância do protocolo associada ao endereço de destino que irá entender a mensagem quando ela for recebida, e que haja alguma ligação com um usuário do protocolo no destino de modo que os dados sejam entregues a alguém.
- Tanto a abordagem com ou sem conexão necessita de algum tipo de inicialização.
- Ambos requerem que antes os endereços nos quais desejam se comunicar sejam conhecidos. Isto é feito na fase de registro.

Modelo Unificado

- Quando o usuário estiver pronto para enviar ou receber dados, devem ser alocados recursos para dar suporte à comunicação e devem ser criadas associações entre uma aplicação e máquina de protocolo abaixo.
- A única forma de ter uma interface comum seria que ambas apresentassem o mesmo comportamento para o usuário.
- Quanto mais estado compartilhado for mantido, mais orientada a conexão será a comunicação.
- O usuário do serviço não tem necessidade de saber quais mecanismos são usados pelas máquinas de protocolo, apenas as características resultantes vistas pelo usuário.

Modelo Unificado

- É necessário um comportamento comum na interface:
  - Parecido com o comportamento da interface de conexão: criar, enviar/receber, deletar.
  - Como não criam conexões, termos como *conecte* e *estabeleça* não estão corretos.
- Conceito mais neutro:
  - O usuário solicita ao serviço abaixo que "aloque" recursos de comunicação. Alocar não implica na existência de conexões.
  - A camada deve decidir se utilizará uma conexão e não o usuário da camada!
  - O usuário deve apenas escolher as características que os recursos devem ter.

Modelo Unificado

- O usuário deve solicitar recursos de comunicação com certas características: largura de banda, atraso, taxa de erros, etc.
- É tarefa da camada determinar como satisfará o pedido, dadas as características do serviço que dá suporte e todos os demais pedidos.
- Como ele faz não deveria ser uma preocupação do usuário.
- A escolha é um compromisso entre a alocação estática ou dinâmica dos recursos.

Modelo Unificado

- *Quanto mais determinístico for o tráfego (menor variância), mais a alocação de recursos será tipo conexão e estática;*
- *Quanto menos determinístico for o tráfego (maior variância), mais a alocação de recursos será sem conexão e dinâmica.*

- Insistir que uma das duas seria a melhor em todos os casos é loucura.

Os Tipos de Mecanismos

Tipos de Campos da PCI associadas com os Mecanismos

- **Campos fortemente acoplados:** aqueles que estão associados aos dados do usuário.
  - Ex.: números de sequência para ordenação, CRC para a detecção de corrupção dos dados.
- **Campos fracamente acoplados:** aqueles que não estão associados aos dados do usuário
  - Ex.: associados com sincronização, controle de fluxo, reconhecimentos que podem ser, mas não têm que estar associados à PDU de transferência.

Acoplamento de Mecanismos

- Mecanismo fortemente acoplado:
  - mecanismo que é função apenas de campos fortemente acoplados.
- Mecanismo fracamente acoplado:
  - mecanismo que é função de pelo menos um campo fracamente acoplado.

Quantas PDUs deve ter um protocolo?

- Uma das principais decisões a serem tomadas no projeto de um protocolo é o número e formato das PDUs.
- Deve haver pelo menos uma PDU para transportar os dados do usuário [PDU de Transferência]
- Um princípio de projeto frequentemente citado recomenda que o controle seja separado dos dados.
- Associar mais e mais funcionalidades a uma única PDU corresponde a “sobrecarregar o operador”.
  - Ex.: TCP

Quantas PDUs deve ter um protocolo?

- Para protocolos de transferência de dados:
  - Número mínimo de tipos de PDUs é um.
  - Número máximo parece ser $O(m+1)$:
    - Uma PDU de transferência e m outros tipos, um para cada mecanismo fracamente acoplado.
  - Para protocolos assimétricos o máximo pode ser $O(2m)$, porque muitas funções consistem de PDUs de Pedido e Resposta.
  - Protocolos simétricos têm $O(m)$ tipos de PDUs

Os Tipos dos Protocolos

- Há mecanismos que podem aparecer em qualquer protocolo.
  - Ex.: delimitação, alocação, negociação de política, detecção de dados corrompidos, etc.
- Semelhanças entre protocolos de transporte e protocolos de camada de enlace:
  - Se preocupam primeiramente com controle de erro fim-a-fim e controle de fluxo.
  - A diferença é que os "fins" estão em lugares diferentes. Possuem diferentes *escopos*.

Os Tipos dos Protocolos

- Os protocolos de rede e MAC são semelhantes enquanto lidam primeiramente com o repasse e a multiplexação.
- Nos protocolos de repasse e multiplexação, a política é sempre imposta pelo transmissor.
- Nos protocolos com controles de erro e de fluxo, a política é sempre imposta pelo receptor: são mecanismos de realimentação.
- A distinção entre estado compartilhado fracamente ou fortemente acoplados depende da presença de realimentação.
- Estes dois tipos de protocolos tendem a se alternar nas arquiteturas...

Observações

- O repasse sempre cria a oportunidade para que PDUs sejam perdidas. Portanto, para garantir confiabilidade deve sempre haver um protocolo de controle de erro sobre um protocolo de repasse.
- Isto indica que há praticamente apenas três tipos fundamentais de protocolos:
  - Dois protocolos de transferência de dados com diferentes políticas:
    - Protocolos de repasse e multiplexação e
    - Protocolos com controle de erro e de fluxo.
  - Protocolos de Aplicação.

Alternância dos Protocolos

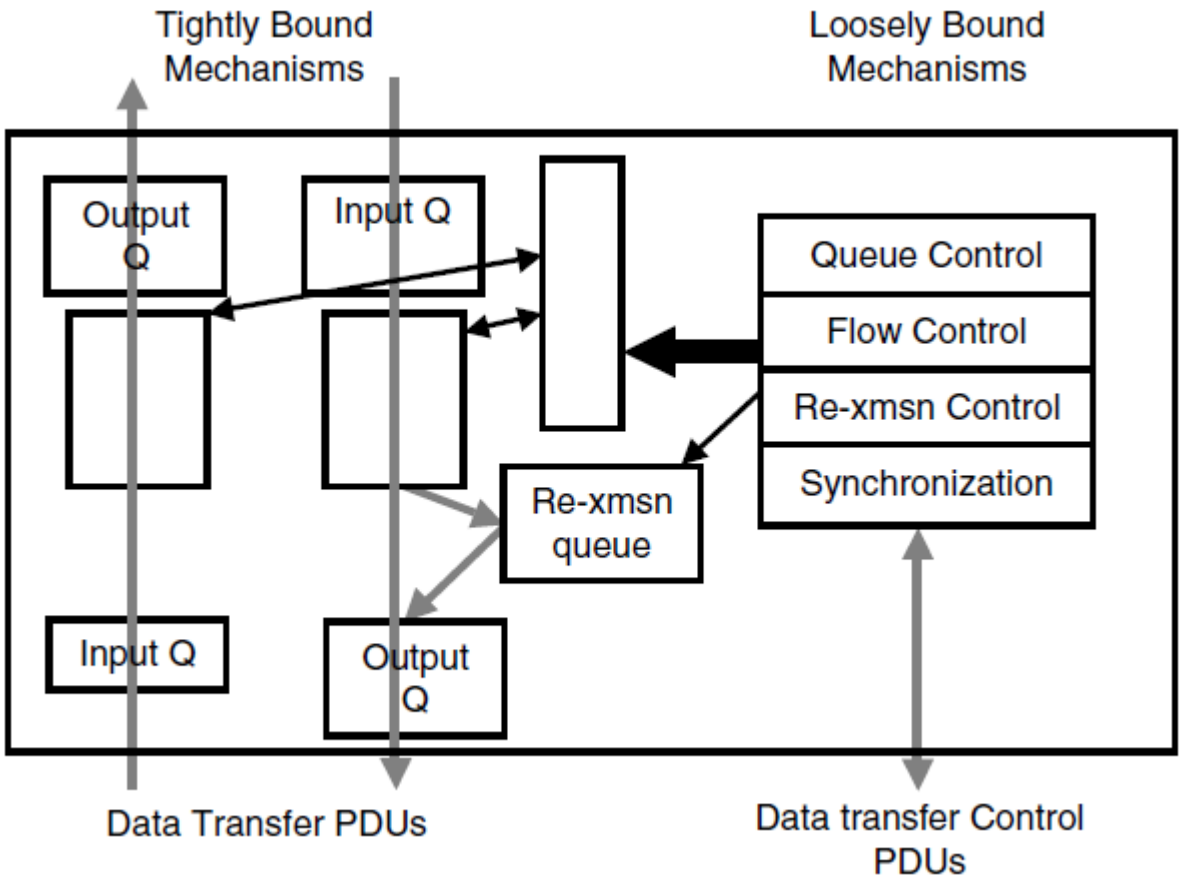
- Os protocolos de repasse e os de controle de erro devem se alternar na arquitetura (para não duplicar esforços). Se não se alternam, é provavelmente um acidente da história.
- Os protocolos de repasse trabalham normalmente na fronteira do congestionamento e PDUs são descartadas quando não se consegue evitar o congestionamento.
- Em geral é prudente recuperar erros numa camada menos abrangente. Mas, se a probabilidade de erros for baixa, o protocolo de controle pode ser omitido.

A Arquitetura de PMs de Transferência de Dados

Arquitetura das PMs

Fragmentação/
remontagem
Ordenação
Enfileiramento

(praticamente
sem políticas)



Alocação
Controle de
fluxo
Reconhecimentos

(todo baseado
em políticas)

Os dois lados são praticamente independentes!

Consequências

- Se olharmos o formato de uma PDU de transferência apenas com mecanismos fortemente acoplados, vemos que possui uma grande semelhança com protocolos como o UDP.
- Podemos então pensar numa única estrutura de protocolo que acomode toda a faixa de protocolos desde puramente sem conexão a completamente com conexão, meramente mudando a política!

Encontrando uma Síntese: A Parte Difícil

- Para o modelo de serviço/interface, abstraímos o problema num modelo de alocação de recursos e exigimos que o usuário especificasse os parâmetros da comunicação que estava sendo solicitada dentro da caixa preta.
- Agora temos que encontrar um modelo comum para as funções de conexão e sem conexão!

Diferenças entre com e sem conexão

- Quantidade de estado compartilhado e grau de consistência (o quão fortemente acoplado deve ser o estado para a sua correta operação?)
- Com e sem conexão são extremos num contínuo entre estados fracamente acoplados a mais fortemente acoplados.
- Se esta hipótese for verdadeira então devem haver outros pontos de operação ao longo deste contínuo.

Outros Pontos de Operação

- Análise de Belnes (1976):
  - Requisitos para a entrega confiável de uma mensagem:
    - Troca em 5 vias entregará confiavelmente uma mensagem mesmo que um dos hosts reinicialize.
    - Uma troca em 4 vias é suficiente desde que nenhum dos hosts reinicialize.
    - Isto dá uma ideia da quantidade de estado compartilhado necessário para uma entrega confiável.

Outros Pontos de Operação

- Abordagem baseada no tempo (Watson, 1981):
  - Para um protocolo de transporte
  - Baseado na perspectiva de que todas as conexões existem e sempre existiram.
  - As informações a seu respeito são mantidas numa cache apenas quando estiverem ativas.
  - A troca de PDUs serve apenas para refrescar a cache.
  - Watson provou que os temporizadores no protocolo delta-t eram necessários e suficientes.

Outros Pontos de Operação

- Problema dos Generais Bizantinos (Lamport, 1982):
  - Para redes, o resultado desta análise é que dentro de um único protocolo funcionando num meio não confiável, é impossível determinar se a última mensagem chegou ao seu destino.
  - Numa arquitetura apropriada de camadas com o estabelecimento e liberação tanto na camada de aplicação como na de transporte, os generais saberiam em que estado terminou sua conversa, mas não seriam capazes de saber se o canal de comunicação confiável foi terminado corretamente.
    - A última mensagem do general não é a última mensagem transmitida!

Outros Pontos de Operação

- Spector (1982) considerou o mesmo problema que Belnes de outra perspectiva:
  - Dada uma troca em particular, o que podemos afirmar sobre se a operação ocorreu?
  - Considerou a perspectiva de PDUs de pedido/resposta para aplicações cliente/servidor
  - Semântica das operações remotas na presença de mensagens perdidas:
    - "Talvez", uma única PDU sem resposta
    - "Pelo menos uma vez", uma pergunta e uma resposta
    - "Apenas uma vez", saudação em três vias

Outros Pontos de Operação

- Clark (1988) caracteriza o problema do estado compartilhado em termos mais qualitativos de estados macios x duros e se as falhas estão sujeitas a "compartilhamento do destino".
- A abstração que ele considerou para contrastar foi a natureza *hop-by-hop* do X.25 com o controle de erros fim-a-fim do TCP.

Estados Macios (*Soft States*)

- Ping et al. (2003) descrevem cinco casos de "dureza" crescente de um estado puramente macio a um estado duro:
  - Estado macio puro (ss)
  - Estado macio com remoção explícita (ss+er)
  - (ss+rt) – rt = *reliable trigger*
  - (ss+rtr) – rtr = *reliable trigger/removal*
  - Estado duro (hs)

Estado macio puro (ss)

- O transmissor envia um PDU de disparo ao receptor
- O transmissor inicia um temporizador e refresca o estado sempre que o temporizador expirar, enviando o valor atual da informação de estado em uma nova PDU de disparo.
- O receptor grava o conteúdo da PDU de disparo quando ela chega e inicia o seu próprio temporizador que é reiniciado sempre que chegar novas mensagens.
- Se o temporizador do receptor estourar, ele deleta o estado.

Estado macio puro (ss)

- Ping distingue um estado de "falsa remoção" de estado quando o temporizador do receptor estoura antes da chegada de uma nova PDU de disparo.
- Como podemos saber a intenção do transmissor?
  - Talvez tenha havido uma mudança de estado por lá.
  - Como o transmissor sabe que o estado está sendo descartado? Ele se importa com isto?
  - Do ponto de vista do transmissor ou do receptor, não há "falsa remoção".
  - As ações das PMs devem se basear apenas nas entradas que ele recebe.
  - Nenhuma hipótese pode ser feita ou deve ser assumida sobre o que gerou estas entradas.

Estado macio com remoção explícita (ss+er)

- Uma PDU de remoção de estado explícita (ER) é enviada pelo receptor quando estourar o seu temporizador.
- Forma fraca de "pelo menos uma vez": Se uma ER for recebida, o transmissor saberá que pelo menos uma das suas PDUs de disparo foi recebida.
- A PDU de ER não seria necessária pois o transmissor irá enviar uma PDU quando estourar o seu temporizador.

Estado macio com disparo confiável (e remoção explícita) (ss+rt)

- O receptor envia um *ack* para cada disparo:
  - O transmissor inicia um temporizador de retransmissão quando a PDU de disparo é enviada.
  - Se expirar o temporizador do transmissor sem que tenha sido recebido um *ack*, este retransmite a mesma PDU de disparo.
  - A remoção de estado é explícita como no segundo caso.
  - Corresponde a uma forma forte do caso "pelo menos uma vez" ou a uma forma fraca do "apenas uma vez" de Spector.

Estado macio com disparo/remoção confiável (ss+rtr)

- Uso de mensagens confiáveis para lidar não apenas com estabelecimento/atualização de estado mas também com remoção de estado.
- Semelhante ao caso anterior:
  - A única informação que o ER adiciona à semântica é o conhecimento de que pelo menos um disparo foi recebido, e não há indicação de que o transmissor necessita desta informação.

Estado duro (hs)

- Uso de PDUs confiáveis para estabelecer, atualizar e remover o estado, mas sem especificar como ele é tornado confiável.
- Um problema significativo com estado duro é a remoção de estados órfãos e a dependência de sinais externos para a sua remoção.

Análise

- Há basicamente três casos para os protocolos de transferência de dados:
  - (talvez) um protocolo sem conexões puro
  - Um protocolo com apenas mecanismos fortemente acoplados requerendo apenas uma saudação em duas vias para sincronização
  - Um protocolo com mecanismos fracamente acoplados fornecendo realimentação (pelo menos uma vez) e requerendo uma saudação de três vias para a sincronização (apenas uma vez).
- Mas, três pontos não caracterizam um contínuo.

Enigma

- Sem conexão sempre foi associado a um serviço de "melhor esforço".
- Tem crescido a necessidade de níveis de serviço além do "melhor esforço".
- Superdimensionamento não é uma solução de longo prazo para esta questão.
- Isto nos leva ao modelo com conexões?
- Não haveria uma síntese?
- E há também problemas conhecidos do modelo com conexão:
  - Alocação estática de recursos, problemas de complexidade e escalabilidade.

Grau de Estado Compartilhado

- Como identificamos apenas três casos, esta é uma indicação de que o grau de estado compartilhado não é uma abordagem que levará à síntese que estamos buscando.

Retorno à caracterização de Clark

- Problema com o estado duro: “por conta da natureza distribuída da replicação (de estados),
  - algoritmos para garantir a replicação robusta são eles mesmos difíceis de construir, e
  - poucas redes com informação de estado distribuída provê algum tipo de proteção contra falhas”.
- A replicação está principalmente associada à alocação de recursos e controle de fluxo.
- Temos que olhar todo o problema: devem ser incluídos também a sinalização e o “plano de controle”.

Paradoxo

- A abordagem orientada a conexão tenta minimizar a quantidade de estado compartilhado!
  - A fragilidade vem da sua inabilidade em responder às falhas.
  - Se um link ou nó ao longo de um circuito virtual quebra, os nós que participam do cv não têm informação suficiente para agir.
  - Não têm informação de estado *suficiente* para isto.

Paradoxo

- A abordagem sem conexão evita isto distribuindo tudo para todos:
  - Todos sabem tudo sobre o roteamento de qualquer PDU e
  - Cada PDU contém toda a informação necessária para que o sistema a encaminhe!
- A abordagem sem conexão possui o *máximo* de estado compartilhado e não o mínimo!

Compartilhamento de Estado

- O conceito de compartilhamento de estado sempre incluiu tanto a quantidade de estados como o grau do acoplamento.
- Protocolos orientados a conexão se livram da complexidade para minimizar recursos (estado) tornando o sistema mais frágil no processo.
- Enquanto que os protocolos sem conexão disseminam informação (estado) amplamente em favor de características mais simples e resilientes.

Eixos Ortogonais

- Sem conexão representa o máximo de estado em termos de espaço,
- Enquanto que conexões são o máximo de estado em tempo.
- A abordagem sem conexão foca inteiramente em roteamento e ignora o problema da alocação de recursos.
- Enquanto que a abordagem orientada a conexão centraliza o roteamento e concentra na alocação de recursos ao longo do caminho.

Teoria Unificadora

- Queremos ser capazes de dar suporte a fluxos com diferentes características (QoS) mantendo a flexibilidade e a resiliência da abordagem sem conexão.
- Distribuímos informação de conectividade para todos, mas sempre insistimos em distribuir informação de alocação de recursos apenas ao longo dos caminhos!
  - Não é de estranhar que as nossas tentativas de prover QoS sempre pareceram conexões, estamos fazendo o que elas fazem!
- Temos que tratar a alocação de recursos como tratamos roteamento: de forma distribuída.

Nosso contínuo?

- Probabilidade de que uma PDU seja processada exatamente como a última PDU no seu fluxo varia entre 0 e 1.
- Baseado na informação recebida de outros roteadores, cada roteador calcula a probabilidade de que esteja no caminho para este fluxo e, neste caso, que recursos deveriam ser “alocados”.

Novo Modelo

- Fluxos são espalhados entre diferentes caminhos e os recursos são alocados baseados na probabilidade de seu uso.
- Se assumirmos que há m caminhos entre A e B:
  - uma conexão será representada por um caminho que tem probabilidade 1 de ser utilizado.
  - No caso sem conexões, cada um dos caminhos será usado com probabilidade $1/m$.
- A questão se torna então que recursos devem ser alocados de acordo com estas probabilidades dada uma certa QoS desejada?

Extensão do modelo de Watson

- Todas as conexões sempre existem; mas os estados só são "cacheados" caso sejam alocados recursos para elas.

Considerações Finais

- Este é um modelo que tenta unificar a abordagem com conexão e a sem conexão.
- Como pensar em termos deste modelo muda a nossa forma de olhar para o problema?
- Ainda não é o momento de fazermos isto.
- Precisamos continuar a nossa jornada até termos todo o ferramental necessário para aplicar esta abordagem à solução do problema.