PRC 2009 – Resumo de Aula
A camada de Transporte – parte 1

- ## Características gerais da camada de transporte
  - **Comunicação lógica entre processos.**
    - **Os processos normalmente estão em hosts diferentes, mas podem até estar no mesmo**
    - **Ou seja, a camada de transporte não vê a rede**
  - **Só há 2 protocolos de transporte dentro da internet TCP e UDP**

- **Outros elementos**

Camada	Comunicação entre
Aplicação	Funções lógicas
Transporte	Processos
Rede	Máquinas
Enlace	Interfaces

Camada	PDU - Unidade de Dados do Protocolo
Aplicação	Mensagem
Transporte	Segmentos
Rede	Datagrama
Enlace	Quadro

- **A função de MUX e deMUX da camada de transporte**
  - **MUX = receber segmentos dos várias aplicações e enviá-los para a rede**
  - **DeMUX = receber segmentos da rede e distribuí-los às aplicações devidas**
  - **Como os processos são endereçados?**
    - **Número das portas origem e destino (parece com o número do processo do SO)**
    - **Porquê o segmento tem que ter as duas portas?**
      - **O processo que recebe as vezes tem que devolver a resposta**
        - **Cliente** **Servidor**

          Origem = x **origem = y**

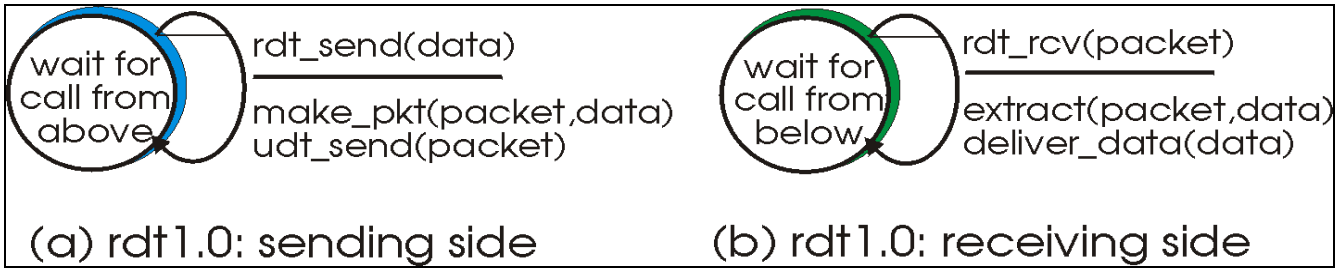
          Destino = y **destino = x**
      - **servidor pode usar a mesma porta para atender vários clientes e não pode ser o mesmo processo. Exemplo HTTP sempre é 80. Um servidor muito usado tem que ter vários processos.**

- **Portas e endereços IP**
    - **Clientes diferentes podem gerar portas origem iguais. E daí????**
        - **Não basta os números de porta. Tem que usar o endereço IP**
        - **Na verdade o trio (IP origem, porta origem, porta destino) identifica o processo destino**
    - **Quais as portas reservadas???**
        - **0 a 1023 - ver RFC 1070**
        - **1024 a 65535 (16 bits) podem ser usadas**

- **O protocolo UDP - User Datagram Protocol**
    - **Adiciona muito pouco ao pacote de dados**
    - **Formato do pacote - slide 8**
    - **Tem o checksum - descarta o pacote se não bater**
    - **Porquê o UDP?**
        - **Sem conexão (ganha tempo)**
        - **Menor cabeçalho (menos overhead)**
        - **sem controle de congestionamento (menos processamento)**
    - **Como usar o UDP em aplicações que precisam de garantia de entrega?**
        - **Controle fim a fim - na aplicação**

- **Princípios de transferência de dados confiável**

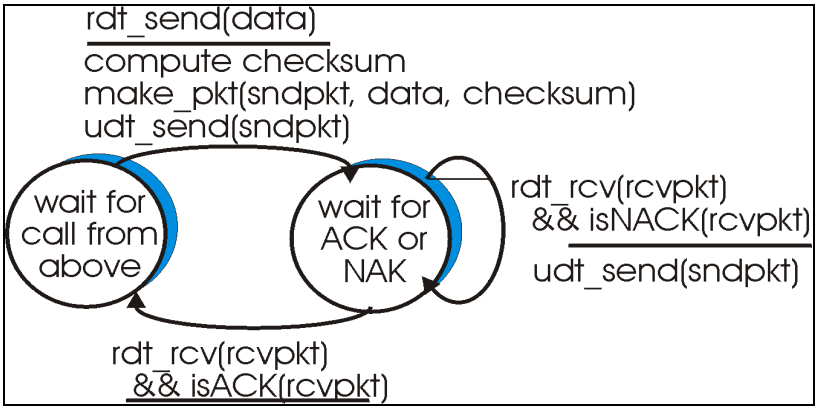
Versão 1.0 - supondo que não há erro de transmissão (canal confiável)

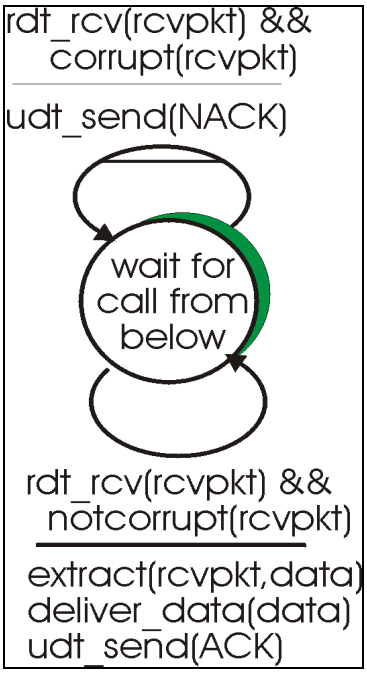


(a) rdt1.0: sending side (b) rdt1.0: receiving side

Princípios de transferência de dados confiável

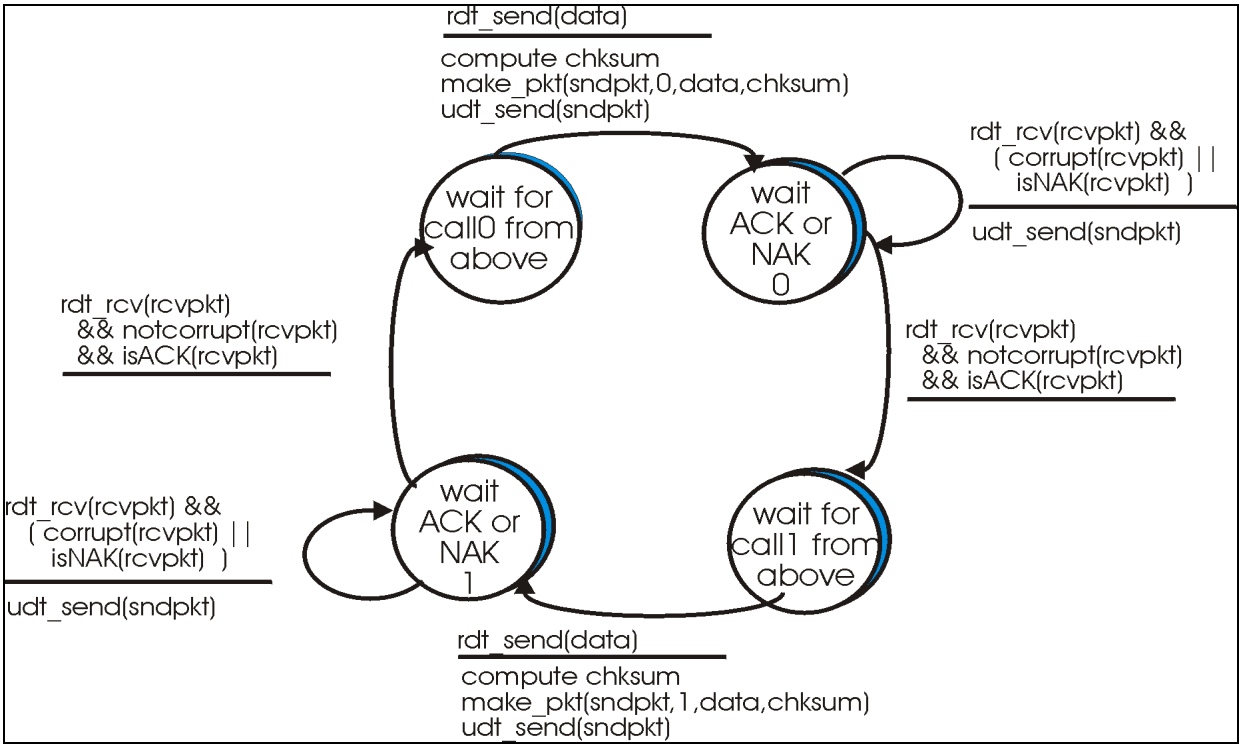
- **Versão 2.0 - supondo que pode haver erro (canal não confiável)**
- **Lado que recebeu envia:**
  - **ACK se chegou bem**
  - **NACK se chegou com erro**
- **Lado que enviou**
  - **Se recebeu NACK retransmite**

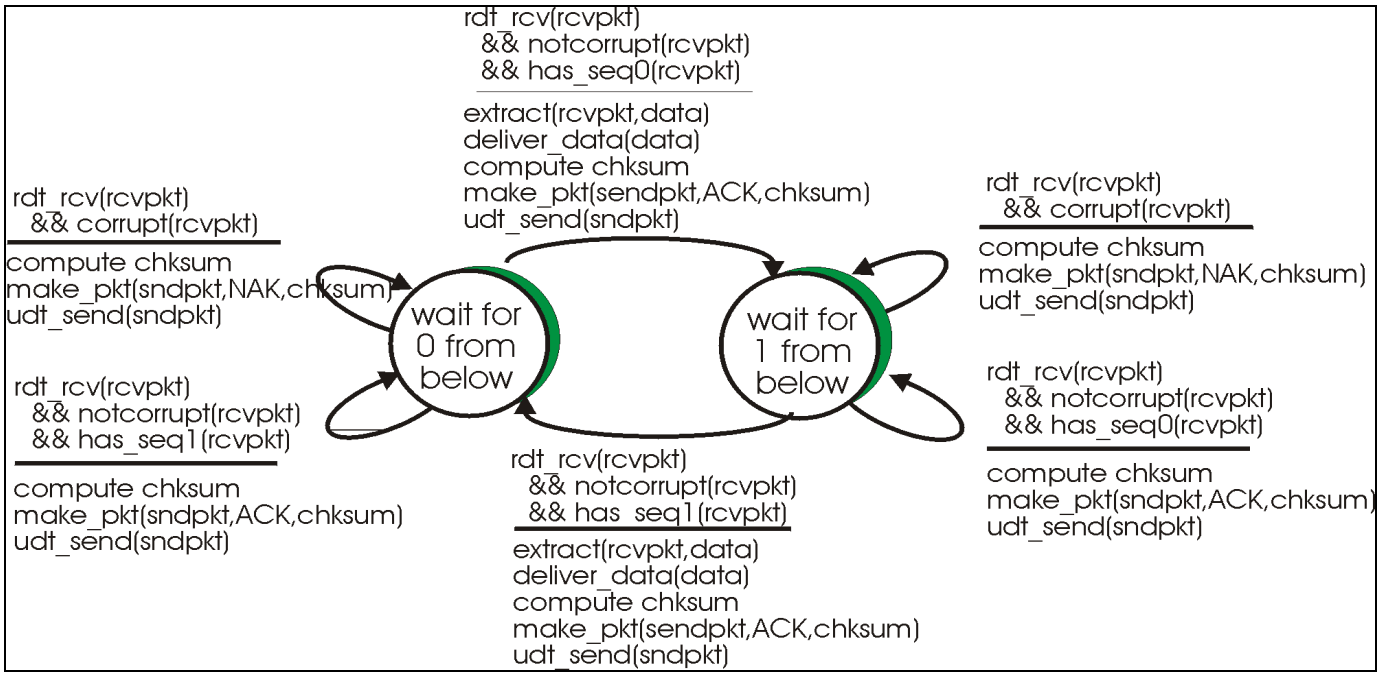




Princípios de transferência de dados confiável

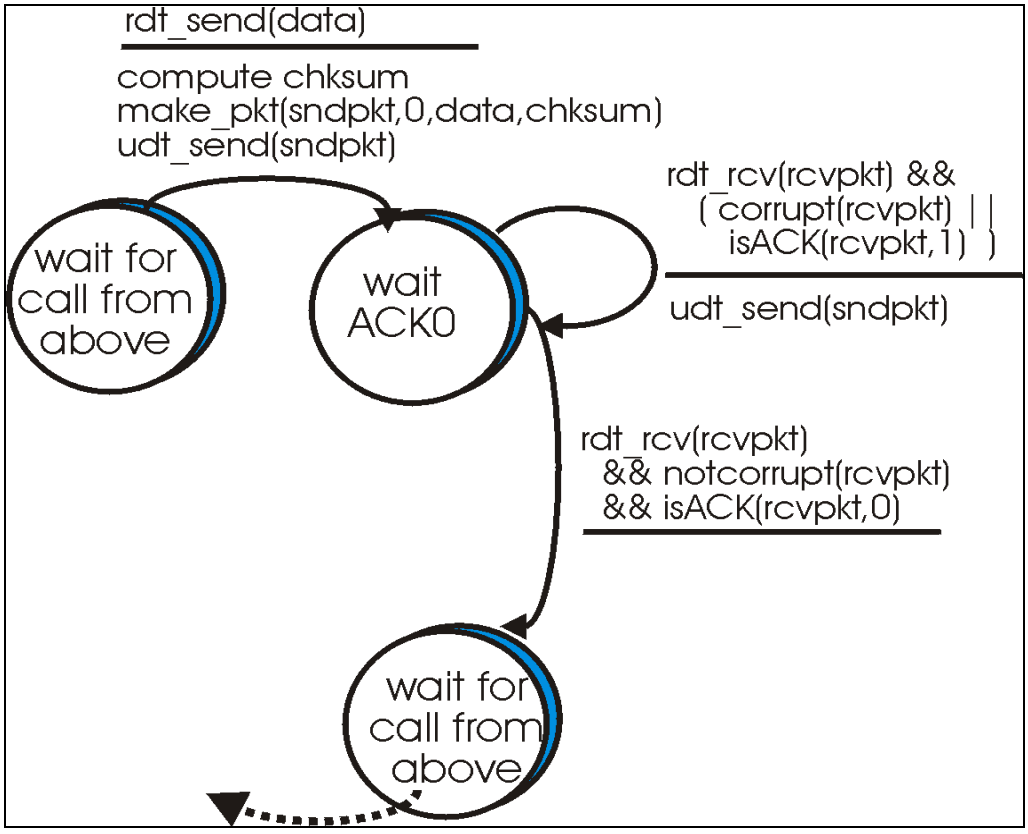
- **Problemas na versão 2.0 –**
  - **E se o ACK/NACK se corromperem no caminho?**
    - **Solução - checksum no pacote com ACK ou NACK**
  - **Outro problema - se o ACK/NACK se corromperem no caminho e o lado que envia reenviar, o receptor pode receber duas vezes o mesmo pacote**
    - **Solução – numerar os pacotes (0 e 1)**
- **Versão 2.1 – checksum nos ACK/NACK e numeração dos pacotes**





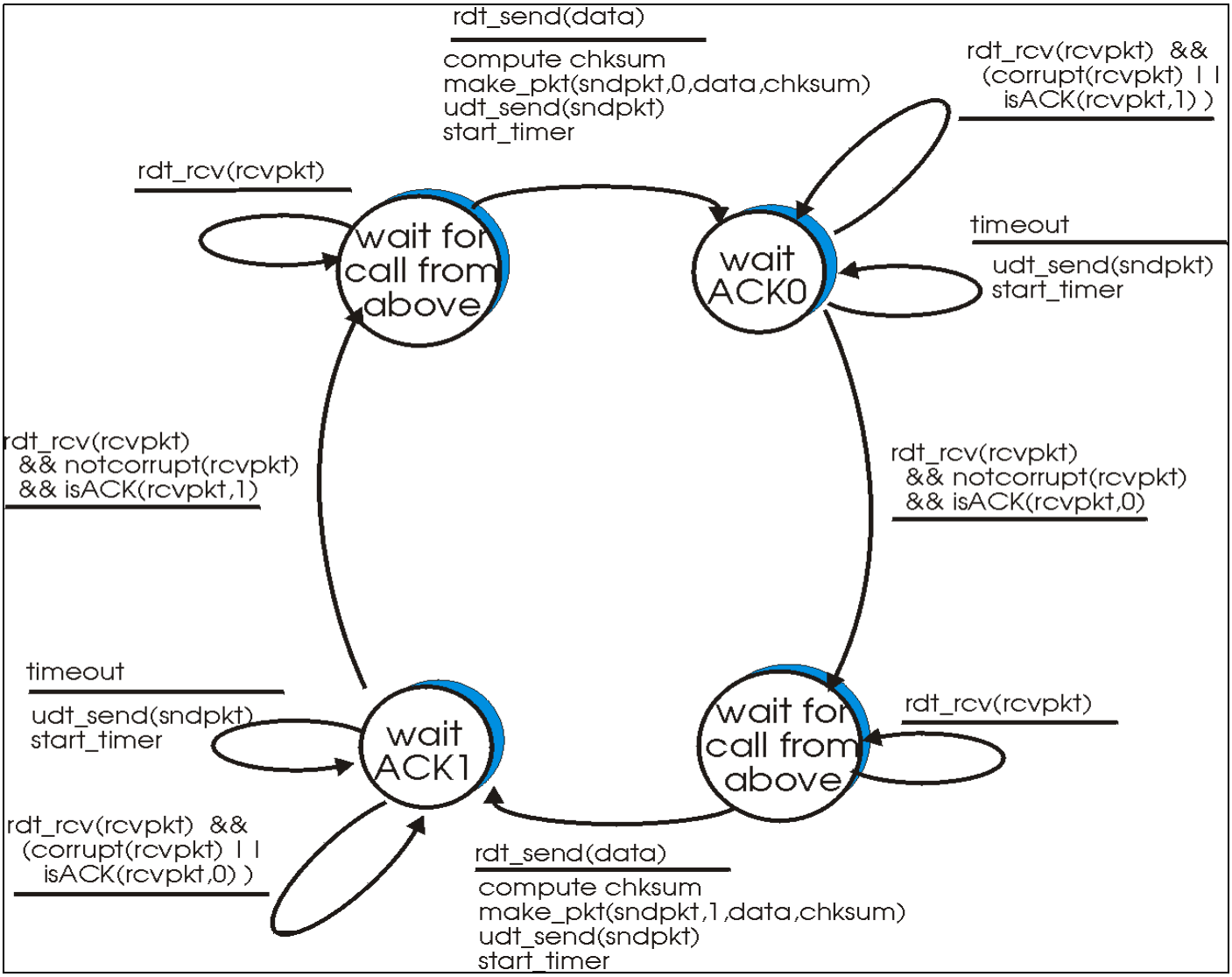
Princípios de transferência de dados confiável

- **Eliminar o NACK**
  - **Substituir por ACK n – n é o último pacote bem recebido**
  - **Em nosso caso tem ACK0 e ACK1**
- **Versão 2.2 – sem NACK – com ACK0 e ACK1**



Princípios de transferência de dados confiável

- **Problema – E quando há perda na rede????**
  - **Pacotes (dados ou ACK) podem não chegar**
  - **Como prever????**
  - **Solução – timeout**
    - **Esperar algum tempo pelo ACK. Se não chegar retransmite o último pacote**
- **Versão 3.0 – com timeout**



Princípios de transferência de dados confiável

- **Resumo**
  - **Checksum com ACK/NACK – verifica erros inclusive nos pacotes ACK/NACK**
    - **Problema – pacotes duplicados; se um lado enviou ACK e o ACK foi corrompido no caminho, o outro lado repete o envio do mesmo pacote**
  - **Número de sequência – para evitar duplicação de pacotes**
  - **Timeout – sumiço de dados**

Princípios de transferência de dados confiável

- Problema na versão 3.0 (com timeout)
  - É um protocolo do tipo stop-and-wait
    - Tem que esperar o ACK para continuar
    - E se o ACK demora, quando o atraso de propagação é grande
      - Exemplos (satélite, fibra longa)
    - Pode ficar muito tempo inativo esperando ACK, sem usar o meio de transmissão
- Solução – enviar muitos pacotes antes de receber um ACK (pipeline ou paralelismo)
- Consequências do paralelismo
  - os pacotes tem que ter números de sequência
  - pacotes que ainda não receberam ACK tem que ser buferizados

Princípios de transferência de dados confiável

- Duas formas principais para esse tipo de protocolo
  - Voltar ao pacote N (GBN)
    - slide 30
  - Repetição seletiva (SR)
    - slide 35
- O TCP – híbrido de GBN com SR