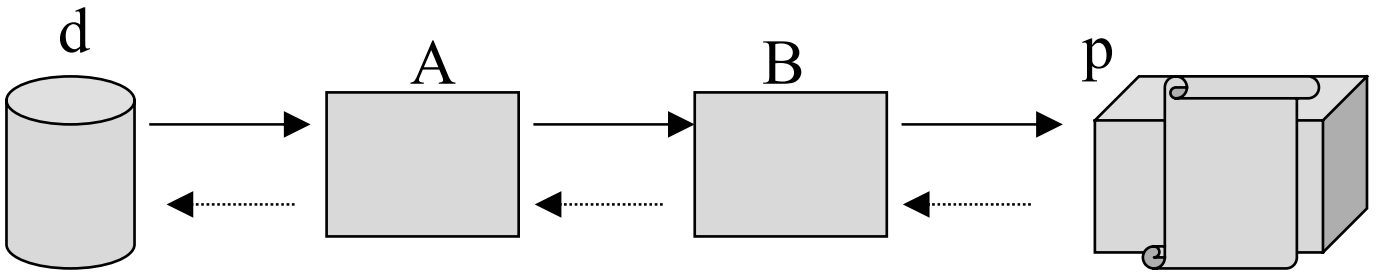
Tema 2. Estructura de un protocolo

- Contenidos:
  - Objetivo del capítulo
  - Definición de protocolo. Ejemplo
  - Niveles de abstracción
  - Lenguajes de descripción de protocolos
  - Elementos de un protocolo
  - Las diez reglas del diseño de protocolos

Objetivo de la Ingeniería de Protocolos

- Elegir un conjunto de reglas a la vez completo y consistente
  - Especificar todos los aspectos relevantes
  - Separar aspectos independientes entre sí
- Estrategias
  - Modularidad
  - Estructura

Ejemplo: Protocolo de transferencia de ficheros (W.C. Lynch)



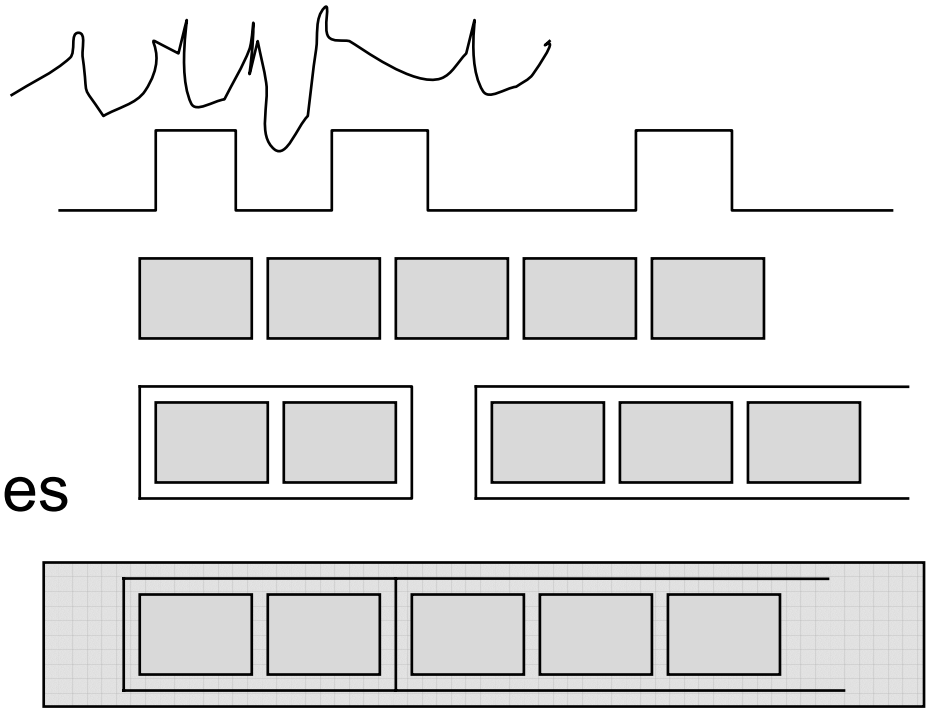
- Aspectos que hay que especificar
  - Conexión, velocidad binaria y señales
  - Codificación de los caracteres
  - Mensajes de control (*start*, *suspend*, *resume*, *conclude*,...)
  - Control de errores

Definición de Protocolo

- Conjunto de reglas, formatos y procedimientos acordados entre los dos extremos de la transmisión (A y B) para que ambos se puedan comunicar.
- Formaliza la interacción entre A y B normalizando el uso del canal de comunicación
  - Inicialización y terminación del intercambio de datos
  - Sincronización de transmisor y receptor
  - Detección y corrección de los errores de transmisión
  - Formatos y codificación de los datos

Descripción en varios niveles de abstracción

- Señales eléctricas
- Bits
- Símbolos o caracteres
- Campos de los mensajes
- Paquetes o tramas



Lenguajes para la descripción de protocolos

- Lenguaje natural
  - Todavía bastante común
  - Problemas de interpretación
  - Dificultad para generar descripciones rigurosas
  - Es necesario definir exactamente el significado de algunos términos (*must*, *may*, etc.)

Lenguajes para la descripción de protocolos (II)

- Lenguajes semiformales
  - Pseudocódigo
  - Diagramas de transiciones, diagramas de flujo
  - No se eliminan del todo los problemas de interpretación.

Lenguajes para la descripción de protocolos (III)

- Lenguajes formales
  - LOTOS, ESTELLE, SDL, Z, ...
  - Fuerte base matemática, soporte para verificación, validación, etc.
  - Problema: disponibilidad de herramientas, paso de la especificación a la implementación final.

Elementos de un protocolo

- Para que el protocolo sea completo, debe definir:
  - El *servicio* proporcionado
  - Las *suposiciones* sobre el *entorno* donde se ejecuta el protocolo
  - El *vocabulario* de los mensajes usados para la implementación del protocolo
  - Los *formatos* de los mensajes del vocabulario (codificación)
  - **Las *reglas* que garantizan la consistencia de los intercambios de mensajes.**

Elementos de un protocolo (II)

- Definir un protocolo =
  - Definir y validar un conjunto de reglas
  - Definir y validar un lenguaje completo y no ambiguo

Servicio	⇒	Semántica
Vocabulario	⇒	Vocabulario
Codificación	⇒	Sintaxis
Reglas	⇒	Gramática

Ejemplo: Protocolo de transferencia de ficheros

- Servicio:
  - Transmisión *full-dúplex* de ficheros de texto como secuencias de caracteres a través de una línea telefónica, con protección ante errores. A y B pueden enviar reconocimientos positivos y negativos (ACK y NACK). Los mensajes tienen dos partes:
    - contenido, en el canal directo
    - control, sobre el canal de retorno
  - Se pueden detectar todos los errores de transmisión

Ejemplo: Protocolo de transferencia de ficheros

- Suposiciones sobre el entorno:
  - 2 usuarios y un canal de transmisión
    - Uno de los usuarios envía un fichero y espera a que la transmisión termine.
    - El canal puede distorsionar los mensajes arbitrariamente, pero no pierde, reordena, duplica o inserta mensajes.
    - Algún elemento de nivel inferior captura todas las modificaciones de los mensajes y las convierte en un nuevo mensaje no modificado, llamado *error*.

Ejemplo: Protocolo de transferencia de ficheros

- Vocabulario:
  - V = {ack, nack, err}
    - ack: contenido y asentimiento positivo
    - nack: contenido y asentimiento negativo
    - err: error
- Formato:

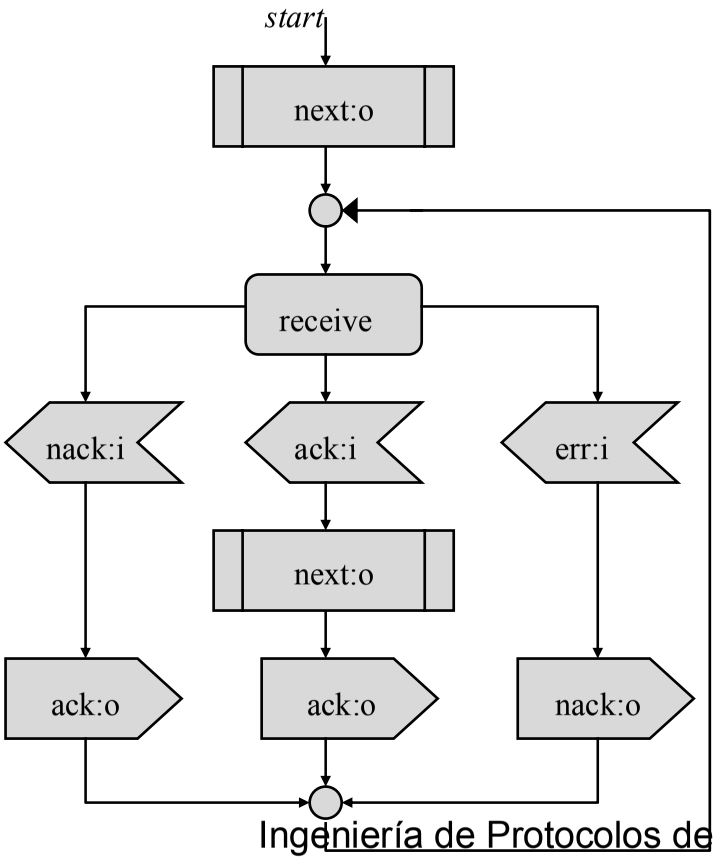

```
enum control {ack, nack, err};
struct message {
  enum control tag;
  unsigned char data;};
```

Ejemplo: Protocolo de transferencia de ficheros

- Reglas
  - Descripción informal:
    1. Si no hubo error en la recepción anterior, el próximo mensaje de retorno incluirá un ACK. Si hubo un error, dicho mensaje incluirá un NACK.
    2. Si la recepción anterior incluía un NACK o tenía un error, se retransmite el mensaje anterior. Si no, se toma un nuevo mensaje para su transmisión

Ejemplo: Protocolo de transferencia de ficheros

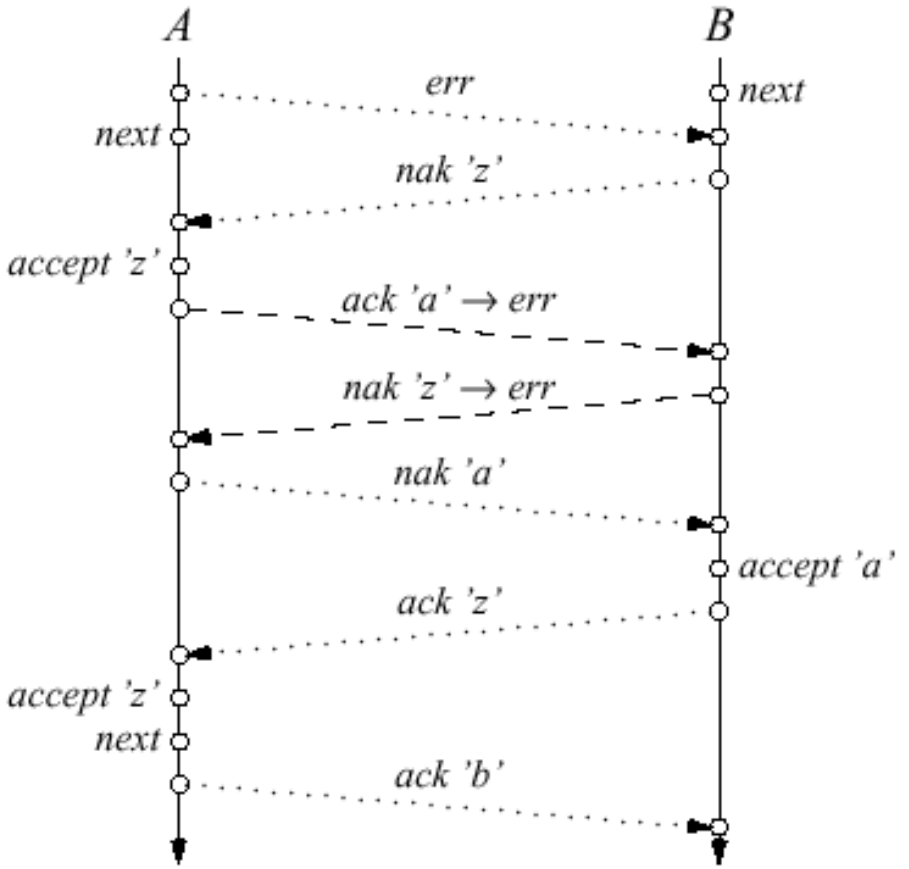
- Reglas: descripción semiformal



Ejemplo: Protocolo de transferencia de ficheros

- Fallos del diseño
  - La transferencia de datos en una dirección sólo puede continuar si hay datos en la contraria
    - Mensajes de relleno
  - Falta indicar cómo se inicia y termina la comunicación
    - Envío de un mensaje de error falso (problemas de sincronización)
    - La terminación exige mensajes de control adicionales
  - El receptor no puede decidir si un dato correcto recibido debe ser aceptado
    - Ejemplo: ¿qué pasa si se producen dos errores de transmisión en secuencia?

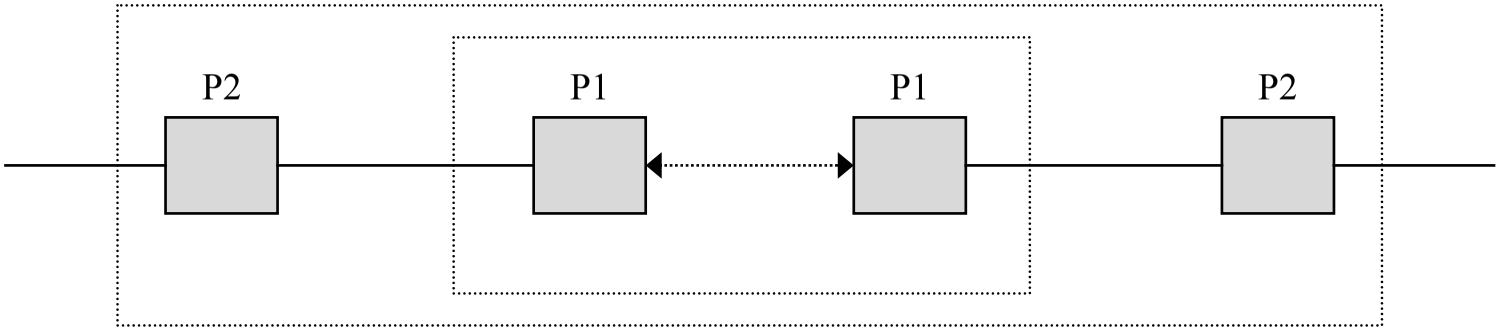
Ejemplo: Protocolo de transferencia de ficheros



El servicio y el entorno

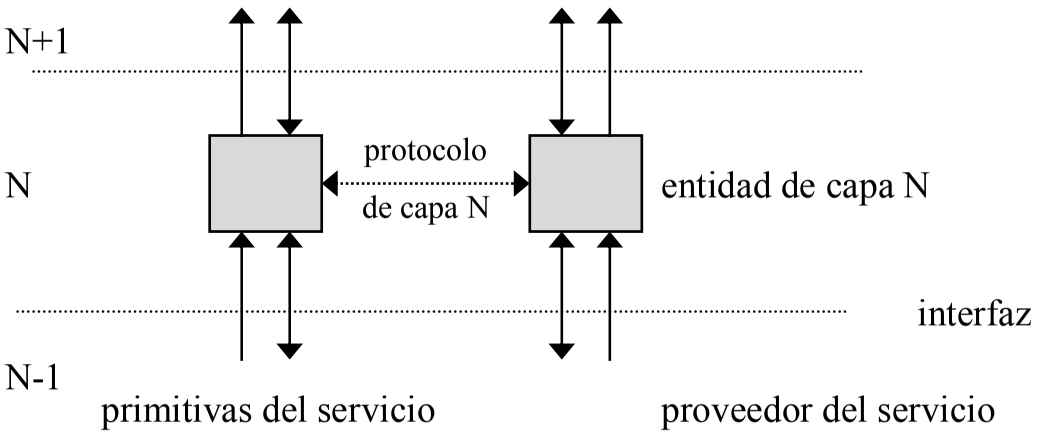
- La especificación del servicio está directamente relacionada con las suposiciones realizadas sobre el entorno
  - Conveniencia de orientar el diseño en varios niveles de abstracción: el entorno en un nivel determinado es consecuencia del servicio proporcionado por un nivel inferior
  - Aparece el concepto de capa (*layer*): Un protocolo de una capa implementa un servicio para las capas superiores, utilizando los servicios proporcionados por las capas inferiores

El servicio y el entorno (II)

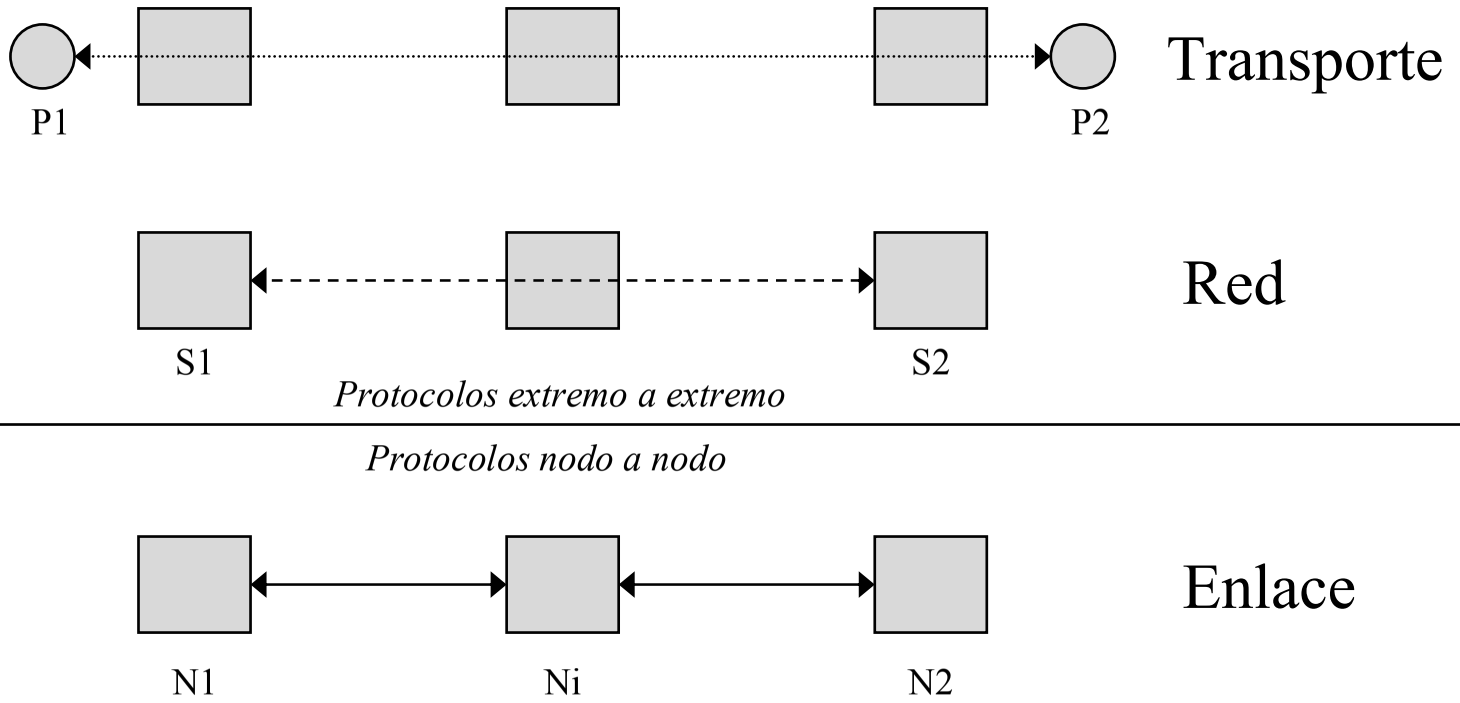


Ejemplos:

- Arquitectura OSI-ISO
- Protocolos de Internet



El servicio y el entorno (III)

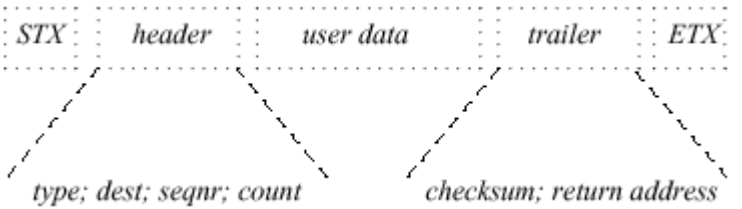


Vocabulario y formato

- Los formatos se corresponden con el nivel más bajo de estructura:
  - Orientados a bit
    - Marca de principio/final de trama
    - Relleno de bits (*bit stuffing*)
  - Orientados a carácter
    - Datos y caracteres de control
    - Relleno de caracteres
  - Orientados a cuenta de caracteres (*byte-count*)
    - No necesitan relleno
    - Incluyen campos con la longitud de la trama

Vocabulario y formato (II)

- Con los formatos anteriores se pueden definir estructuras de más alto nivel.
  - La información de control se colocar en estructuras que encapsulan los datos de usuario.
  - Los mensajes constan de cabecera, datos de protocolo y cola:
    - Cabecera (*header*): tipo de mensaje, direccionamiento, tamaño, número de secuencia, ...
    - Cola (*trailer*): Redundancia/*checksum*, direccionamiento, ...



Dinámica del Protocolo (reglas)

- En general, el desarrollo de protocolos es parte de la ingeniería de software. Pero:
  - La dinámica del protocolo (reglas) se implementa mediante programas ejecutados concurrentemente por un conjunto de procesos que interaccionan
    - El comportamiento de un protocolo depende de la ordenación temporal de eventos
    - El comportamiento de un protocolo no siempre es reproducible
    - No existe una metodología de diseño que garantice, a priori, que las reglas producidas son consistentes y completas

Dinámica del Protocolo (reglas, II)

- Para asegurarnos de que el diseño de un protocolo es correcto necesitamos herramientas de apoyo
  - Herramientas para la especificación del protocolo
    - Lenguajes formales, semiformales, máquinas de estados, etc.
  - Herramientas para la definición de criterios de corrección
  - Herramientas para la comprobación de la consistencia de las reglas definidas
  - **Metodologías de diseño adecuadas**

Dinámica del Protocolo (reglas, III)

- Las dos ideas básicas del diseño estructurado de protocolos:
  - **Simplicidad**: Un buen protocolo se construye con unos pocos módulos
    - Bien diseñados, bien comprendidos
    - Hacen una pequeña función, y la hacen bien
    - Permite hacer protocolos simples, robustos y eficientes y fáciles de comprobar.
  - **Modularidad**: Las piezas interaccionan de un modo simple y bien definido. Cada pieza es a su vez un pequeño protocolo

Dinámica del Protocolo (reglas, IV)

- Un protocolo debe estar
  - **Bien construido**:
    - No está subespecificado ni sobreespecificado
    - Está acotado: no desborda los recursos del sistema
    - Se autoestabiliza y se autoadapta
- y debe ser
  - **Robusto**
    - Preparado para todas las acciones y secuencias de acciones posibles (no solo las probables), en cualquier condición.
    - Los requisitos sobre el entorno deben ser los mínimos
  - **Consistente y completo**:
    - Libre de bloqueos, ciclos improductivos y terminaciones anormales

Recapitulación

- **Las 10 reglas del diseño**
  1. Asegúrate de que el protocolo está bien definido
  2. Define el servicio antes de decidir las estructuras para implementarlo (el qué antes que el cómo)
  3. Diseña la funcionalidad externa antes de la interna
  4. Mantén el diseño simple
  5. No conectes lo que es independiente. Separa los aspectos ortogonales
  6. Mantén el diseño abierto. Resuelve una clase de problemas, no problemas particulares
  7. Antes de implementar, verifica un prototipo

Recapitulación (II)

- **Las 10 reglas del diseño (continuación)**

  8. Implementa el diseño, mide sus prestaciones, y si es necesario, optimízalo.
  9. Comprueba que la implementación final optimizada es equivalente al prototipo que verificaste en el punto 7
  10. Nunca violes las reglas 1 a 7

  La regla menos cumplida es la regla 10