# Introducción a las Redes de Computadores

# Obligatorio 3 – 2011

# Servicio de Transporte Confiable RDT (Reliable Data Transfer sobre IP)

**Facultad de Ingeniería**
**Instituto de Computación**
**Departamento de Arquitectura de Sistemas**

*En este obligatorio se implementará el servicio de transferencia de datos confiable similar al especificado en el libro del curso como rdt.3.0. El mismo debe implementarse sobre el protocolo IP, que provee un servicio de datagramas no confiable (best effort).*

### Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en `http://www.fing.edu.uy/inco/pm/uploads/Ense%f1anza/NoIndividualidad.pdf`

En particular está prohibido utilizar documentación de otros grupos, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (news, correo, papeles sobre la mesa, etc.).

### Forma de entrega

La entrega debe realizarse mediante el moodle del curso via la "actividad" que se habilitara oportunamente. Se recomienda realizar una entrega con tiempo, a los efectos de verificar que su sistema le permite entregar correctamente. Se considerará como válida la última entrega dentro del plazo asignado.

Se debe entregar un solo archivo 'ob3.tar.gz' que contenga los ítems descritos en el apartado **Entregable**. Dicho archivo deberá ser generado utilizando la herramienta GNU TAR y compresión gzip. Otros formatos (bz2, rar, zip, cab, jar, etc.) no son válidos y serán rechazados.

### Fecha de entrega

Los trabajos deberán ser entregados siguiendo el procedimiento descrito anteriormente antes del domingo 29 de mayo de 2011 a las 23:55 horas, sin excepciones. No se aceptará ningún trabajo pasada la citada fecha. En particular, no se aceptarán trabajos enviados por mail a los docentes del curso, ni entregados en medios magnéticos en el instituto.

### Observaciones

La corrección se hará en la máquina virtual del curso tal como se distribuyó, sin cambios o agregados, por lo que debe probarse su funcionamiento en ese ambiente previo a la entrega.

### Descripción del problema a resolver

En el presente obligatorio, se implementará un servicio de capa transporte confiable y orientado a conexión (*RDT*), sobre el protocolo de capa red *Internet Protocol (IP)*. El término confiable indica que los datos recibidos en el extremo origen, serán entregados a

la aplicación destino sin pérdidas, y en el mismo orden en el que fueron entregadas, independientemente de los problemas que pueda presentar la transferencia de datos por la red.

Con relación a la especificación rdt 3.0 presentada en el libro del curso, el protocolo a desarrollar presenta las siguientes diferencias:

- No realiza chequeo de correctitud de datos recibidos, suponiendo que la red presenta pérdidas y duplicación pero no modificará los datos transmitidos.
- Se debe implementar buffers para el almacenamiento auxiliar, donde se guardarán los datos hasta efectivizar el envío y entrega de los mismos.

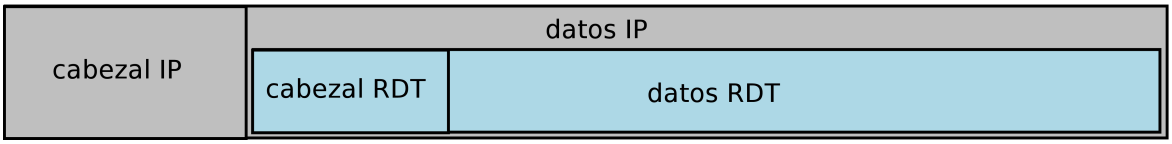
El protocolo debe permitir la comunicación unidireccional desde una aplicación en el equipo origen hacia una en el destino, utilizando *Stop & Wait* como mecanismo de *ARQ (Automatic Repeat reQuest)*. Toda comunicación entre aplicaciones debe ser establecida (y finalizada) mediante los métodos (de conexión y desconexión) que define el protocolo.

Las características generales de una conexión *RDT* son las siguientes:

1. Debe establecerse entre un extremo activo que inicia una conexión utilizando el método `conectarRDT` y un extremo pasivo que espera conexiones utilizando el método `aceptarRDT`.
2. Se enviarán datos en forma unidireccional desde el extremo que inició la conexión hacia el otro extremo.
3. Debe permitir el cierre de conexiones, que debe ser solicitado siempre por el extremo activo.
4. Debe permitir la trasmisión de bytes entre extremos de la conexión (tanto texto como de datos binarios).
5. Debe respetar el orden de entrega. Los datos se deben entregar a la capa de aplicación como un flujo continuo de bytes recibidos, independientemente de los mecanismos utilizados para la transmisión.
6. Es responsabilidad del protocolo garantizar la entrega de los datos que se hayan perdido por cualquier motivo, ya sean fallas en la red o saturación de los buffers de envío y/o recepción del *RDT*.
7. Debe descartar los datos recibidos en forma duplicada.

Para la implementación, se utilizarán sockets del tipo *Raw Sockets* de la capa red, que permiten el envío y recepción de datagramas, sin utilizar los mecanismos aplicados por el sistema operativo. De ésta forma se tendrá acceso a todos los datagramas intercambiados en la interfaz de *loopback (lo)*, siendo responsabilidad de la API diseñada la selección de los datagramas que tengan como destino la aplicación conectada por ella.

Se utilizará el protocolo IP para el intercambio de datos y control, configurando los datagramas intercambiados con el identificador *0xFF* (255) en el campo *Protocol*, correspondiente al protocolo *RDT.*




Es responsabilidad de la API implementada el armado de los datagramas IP a enviar, para lo que se deberá colocar los datos exigidos por el protocolo IP en los campos de control y datos.

Los datagramas IP intercambiados, contarán en su carga útil con datos en formato *RDT*, con los campos de control especificados en la letra.

Solo existirá una sesión *RDT* por aplicación. Las sesiones *RDT* se identificarán con los

siguientes 4 elementos,{*IP origen*, *puerto RDT origen*, *IP destino, puerto RDT destino*}. Para cada sesión se utilizarán buffers de almacenamiento auxiliar, que guardarán datos a enviar o recibidos por *RDT*, hasta que la aplicación lo solicite, devolviendo el control a la aplicación mientras ésta no solicite o no entregue más datos.

La gestión de la conexión *RDT* debe implementarse de acuerdo a las máquinas de estados presentada en la Fig. 1.

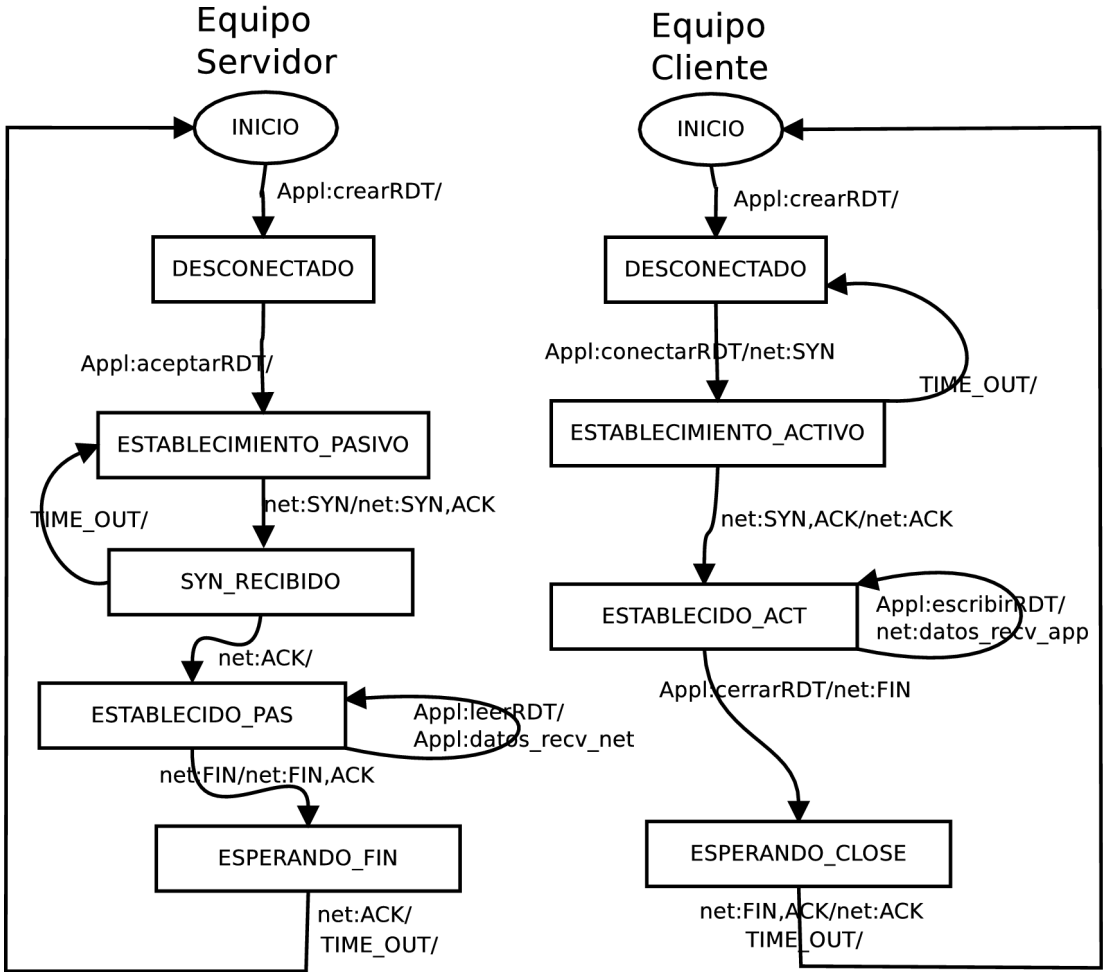



*Fig. 1: Máquina de estados del RDT*

Se deben implementar los siguientes métodos:

- `int crearRDT(struct in_addr localIPaddr)` Crea las estructuras de datos, e inicia los procedimientos necesarios para el control del *RDT*. Devuelve un valor mayor o igual a `0`. En caso de error devuelve -1.
- `int aceptarRDT(unsigned char localRDTport)` Queda esperando en forma pasiva la conexión de otro equipo a través del puerto RDT especificado (`localRDTport`). Esta función bloquea la aplicación hasta recibir una conexión. En caso de realizar en forma satisfactoria el procedimiento de conexión devuelve `0`, en caso contrario devuelve `-1`.
- `int conectarRDT(unsigned char localRDTport,unsigned char peerRDTPport, struct in_addr peerIPaddr)` Inicia en forma activa la conexión con otro equipo (`peerIPaddr`) a través del puerto RDT especificado (`peerRDTport`). Esta función bloquea la aplicación hasta establecer la conexión. En caso de realizar en forma satisfactoria el procedimiento de conexión devuelve `0`, en caso contrario devuelve `-1`.
- `int escribirRDT(const void *buf, size_t len)` Solicita el envío al otro

extremo de la conexión, de los datos pasados en `buf` de largo `len`. Esta función, dependiendo del estado y capacidad de los buffers disponibles, aceptará para envío hasta `len` bytes pasados en la variable `buf`, los que se almacenarán hasta tener la certeza de su recepción por el extremo opuesto. La función devuelve la cantidad de bytes aceptados para su transmisión; en caso de no ser posible por falta de buffers devuelve `0` y en otros casos el procedimiento devuelve `-1`. En caso de no haberse establecido conexión previamente devuelve `-1`.

- `int leerRDT(void *buf, size_t len)` Entrega como máximo `len` bytes recibidos por la conexión. En caso de no contar con datos devolverá `0`. Los datos se devuelven a través del parámetro `buf`. El procedimiento devuelve cantidad de bytes devueltos en `buf`, y en caso de error devuelve `-1`. En caso de no haberse establecido conexión devuelve `-1`.
- `int cerrarRDT()` Cierra la conexión en forma ordenada y destruye las estructuras de datos, y los procedimientos necesarios para el protocolo *RDT*. Previo al cierre deben enviarse los datos que se encuentran en los buffers de *RDT*. En caso de error devuelve `-1`.

El intercambio de información entre los dos extremos se debe realizar a través de *TPDUs (Transport Protocol Data Unit)* con formato *RDT*. Las *TPDUs* contienen un cabezal cuya estructura es presentada a continuación y un campo de datos con largo entre `0` y `MAX_IP_SIZE` bytes.

```
struct rdt_header {
  unsigned char        srcPort;
  unsigned char        destPort;
  unsigned char        flags_rdt;    /* Flags de control  */
  unsigned char        nro_SEC_rdt;  /* nro de secuencia de rdt*/
  unsigned char        nro_ACK_rdt;  /* nro de ACK de rdt*/
};
```

El campo `flags_rdt` se utiliza considerando los diferentes bits del octeto, según el siguiente esquema:

- `DAT`, indica si el segmento *RDT* contiene datos (valor en 1). Los segmentos que solo tengan información de control (DAT en 0) no se numerarán para el *ARQ*. Se accede con la máscara 0x80.
- `SYN`, para establecimiento de conexión (similar a TCP). Se accede con la máscara 0x01.
- `ACK`, para indicar que el segmento contiene información de ACK (similar a TCP). Se accede con la máscara 0x02.
- `FIN`, para indicar fin de conexión (similar a TCP). Se accede con la máscara 0x04.

**Se pide:**

Implementar una API en C/C++, que contenga las funciones definidas anteriormente. Dicha API se linkeditará con programas que utilicen las mencionadas funciones. Deben resolverse los siguientes elementos del protocolo:

- Establecimiento de conexión del *RDT*, generando las estructuras sobre IP requeridas para la conexión real.
- Permitir el intercambio de datos en forma confiable, sin pérdida ni duplicación de datos.
- Desconexión del *RDT*.

**Entregable**
Se deberá entregar:

1.Documentación. La documentación debe contener:
a)Descripción exhaustiva en idioma Español del protocolo.
b)Propuesta de implementación del mecanismo de transporte confiable *RDT*.
c)Pseudocódigo detallado y comentado del módulo *RDT* entregado.

2.Implementación:
a)Código fuente.
b)*Makefile* para la compilación del mismo.
c)Instrucciones para su utilización (incluyendo forma de compilación de una aplicación con *RDT*).

**Insumos**
Se entregarán los siguientes insumos para el trabajo:

- Archivo `rdt.h` conteniendo el cabezal de las funciones a implementar, constantes pre-establecidas, y formato del cabezal de los segmentos *RDT*.
- Programas ejemplo que utilizan la API especificada para transmitir información utilizando la implementación de RDT realizada durante este obligatorio.
- Programa ejemplo de envío de datagramas IP sobre *raw socket*, y otro de recepción, con el envío de datos por IP.
- Shell script `netEmulator.sh` que permite la distorsión de la red, duplicando y perdiendo datagramas en la red.

## Documentación

La documentación del obligatorio debe incluirse dentro del archivo de la entrega. La misma debe entregarse como un único archivo tipo PDF de nombre **informeOb3grupoXX.pdf**. El mismo deberá respetar la numeración de secciones acorde a los requerimientos anteriores.

Es necesario que en el informe figuren el número de grupo y el nombre y la cédula de identidad de cada integrante. En caso que esto no se cumpla el obligatorio no será corregido, con la consecuente pérdida del curso de sus autores. NO se aceptarán otros formatos de informe. (ver `http://www.universidad.edu.uy/odfpdf/`).

**Notas adicionales:**

- Se sugiere la utilización del *wireshark*, para visualizar la información generada en la red por la aplicación. El mencionado software no decodifica *RDT*, pero permitirá analizar el envío de flags y valores del cabezal, así como el contenido de los datos.
- Se sugiere la utilización de threads, asegurando la ejecución simultanea de código así como la mutua exclusión.