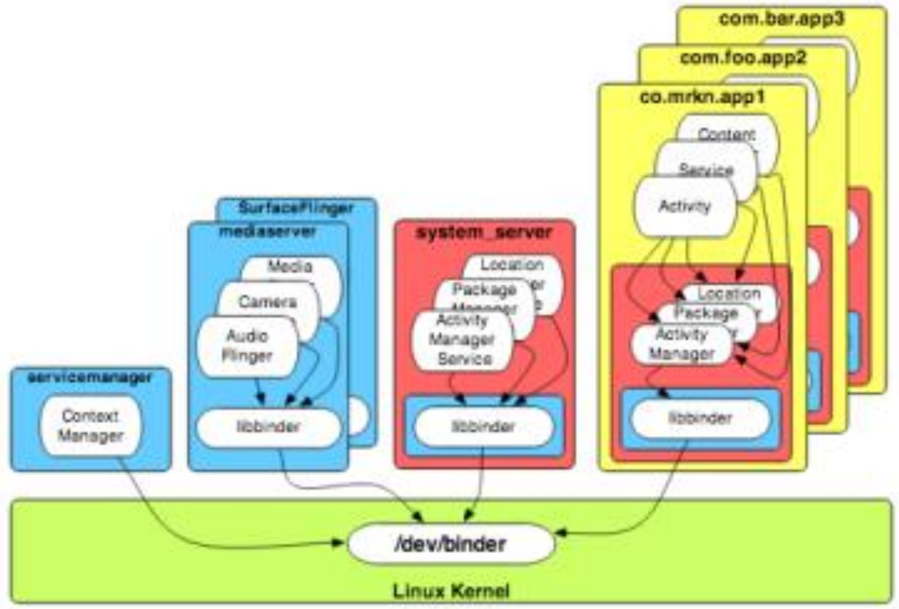
What is Binder

- Comes from OpenBinder : allows processes to present interfaces which may be called by other threads.
- A kernel driver to facilitate inter-process communication
- Lightweight RPC (Remote Procedure Communication) mechanism.
- Focused on scalability, stability, flexibility, low-latency/overhead, easy programming model
- Essential to Android!



Why Binder?

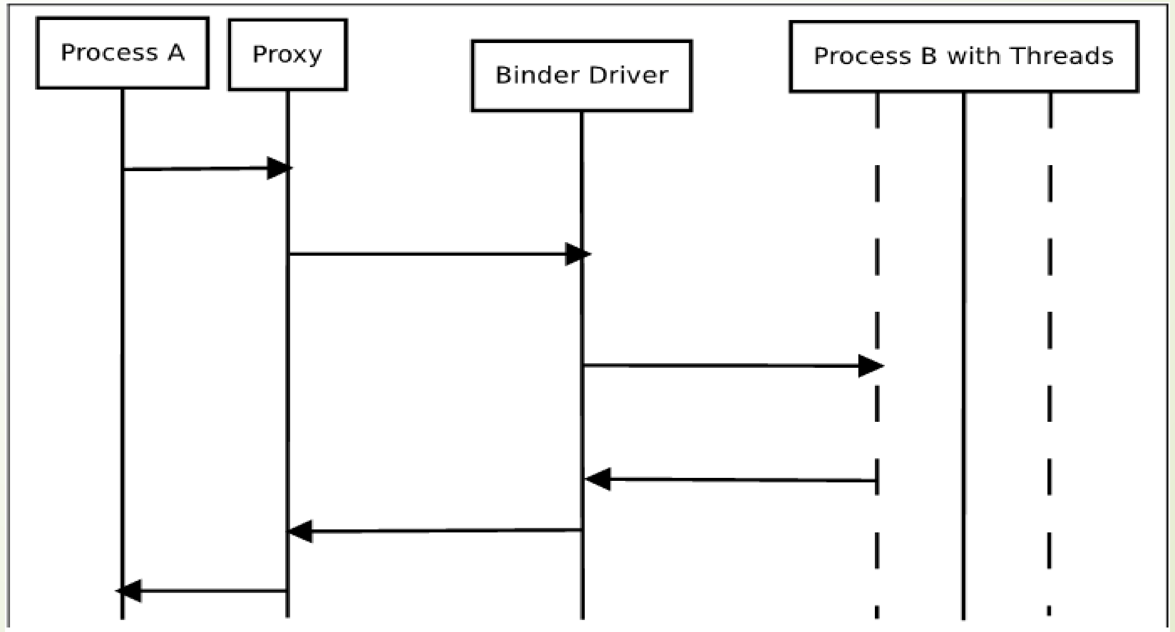
- Memory management: “unneeded” processes are removed to free resources (mainly memory) for new ones
  - Binder to the rescue! Its built-in reference-counting of “object” references plus death-notification mechanism.
- Per-process thread pool for processing requests
- Synchronous and asynchronous (oneway) invocation model
- Unique object-mapping across process boundaries
  - Each instance Binder objects maintains a unique identity across all processes in the system. This facility is provided by the Binder kernel driver, which analyzes the contents of each Binder transaction and assigns a unique 32-bit integer value to each Binder object it see.
  - How to mapping:
    - 1: A virtual memory address pointing to a Binder object in the same process
    - 2: A unique 32-bit handle (as assigned by the Binder kernel driver) pointing to the Binder’s virtual memory address in a different process.

Binder Terminology

- **Binder Object:** is an instance of a class that implements the Binder interface. A Binder object can implement multiple Binders.
- **Binder Protocol:** The Binder middleware uses a very low-level protocol (ioctl-based) to communicate with the driver.
- **IBinder Interface:** A Binder interface is a well-defined set of methods, properties, and events that a Binder can implement. It is usually described by the AIDL1 language.
- **Binder Token:** A numeric value that uniquely identifies a Binder.
- **Binder Driver**: The kernel-level driver that fascinates the communication across process boundaries

How to use Binder

- The Binder framework communication is a client server model
  - Each client initiates communication and waits for response from the server.
- Each client would have a proxy object for the client side communication
- The server side constitutes a thread pool, shall spawn a new thread for each new Binder request from the client



Binder communication

What is AIDL

- AIDL: Android Interface Definition Language , is similar to other IDLs , you might have worked with.
- Allow to define the programming interface that both the client and service agree upon in order to communicate with each other using Binder
- Possible to pass custom object, any primitive data across processes.

Why use AIDL?

- AIDL does nothing but lets the system to generate the boilerplate code that hides the binder IPC detail
- Don’t need AIDL when:
  - Don't need IPC (i.e., your client and server stay in the same process)
  - If you want to write the boilerplate code yourself for IPC

How to use AIDL