# ---Binder là gì ????----------------------------------------

* A kernel driver to facilitate inter-process communication.
* Lightweight RPC (Remote Procedure C**o**mmunication) mechanism.
* Synchronous communication b/w processes
* Not yet another object-oriented kernel.
* Instead, an object-oriented operating system environment that works on traditional kernels, like Linux!
* Focused on scalability, stability, flexibility, low-latency/overhead, easy programming model
* Essential to Android!
* Comes from OpenBinder:  allows [processes](https://en.wikipedia.org/wiki/Process_(computing)) to present [interfaces](https://en.wikipedia.org/wiki/Interface_(computing)) which may be called by other [threads](https://en.wikipedia.org/wiki/Thread_(computing))

# ---Tại sao dung binder -0--------------------------------------------------

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 . When a binder service is no longer referenced by any clients, its owner is automatically notified that it can dispose of it

Per-process thread pool for processing requests.

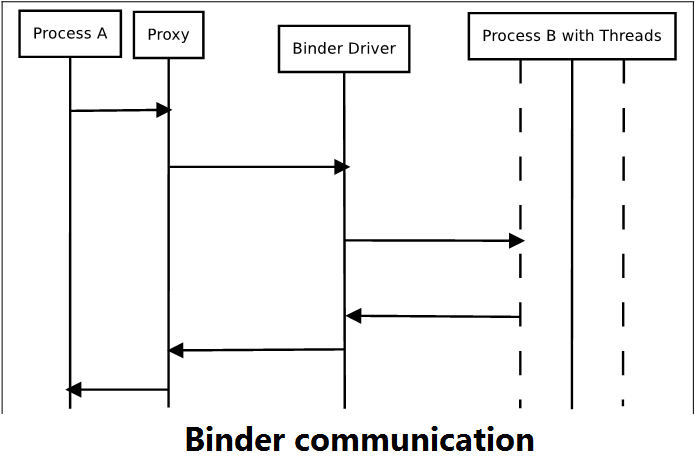
Unique object-mapping across process boundaries A reference to a remote object can be passed to yet another process and can be used as an identifying token.

Methods on remote objects can be invoked as if they were local  (Mục tiêu chính của phương pháp này là làm cho lời gọi từ xa RPC tương tự như thể lời gọi thủ tục thông thường cục bộ và ẩn đi việc truyền dữ liệu đi về qua mạng.)

* Trong cơ chế gọi hàm trong nội bộ một process, lập trình viên thường không quan tâm đến thời gian chuyển thực thi từ đối tượng gọi (caller) vào đối tượng bị gọi (receiver). Thời gian này rất ngắn, chương trình nạp biến cục bộ của caller vào stack. Ngược lại trong mô hình RPC thực tế, khoảng thời gian truyền tham số đối tượng gọi (caller) đến đối tượng bị gọi (receiver) ở xa, rồi kết quả trả về sẽ đi từ receiver về tới caller là không nhỏ. Lập trình viên đành phải chấp nhận hoặc lờ đi hoặc đặt cơ chế hết hạn (time out). Cách lập trình viên tối ưu chương trình cục bộ là chia nhỏ chương trình thành nhiều đối tượng gọi nhau, hoặc các hàm tái sử dụng được gọi lại. Tuy nhiên với RPC, cách chia nhiều hàm để gọi này chưa chắc hợp lý khi thời gian trễ mỗi lần gọi RPC là khó có thể bỏ quả, càng nhiều lần gọi, tổng thời gian trễ sẽ tăng, khả năng nghẽn cổ chai do kiểu hỏi đáp liên tục sẽ tăng. Nhiều nơi gọi là chatty ~ gọi vặt liên tục

# ---Cach dung 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 pool of worker threads.The server shall spawn a new thread for each new Binder request from the client. The bridge between the client and the server process is the binder driver. The **Binder driver** is a character device that is part of kernel space. This module  ensures the client reaches the appropriate destination  across process boundaries.



# ---AIDL là gì ------------------------------------------------------------

 Binder is not comparable with AIDL. They are totally different things. Binder is a IPC mechanism used in Android; while AIDL, as its name -- Android Interface Definition Language, is similar to other IDLs (IDLs describe an interface in a language-independent way, enabling communication between software components that do not share one language, for example, between those written in C++ and those written in Java.) you might have worked with. AIDL allows you to define the programming interface that both the client and service agree upon in order to communicate with each other using Binder. App developers, however, do not use the Binder directly. Instead, they must define and interact with interfaces using AIDL.

* Possible to pass custom object,  any primitive data across apps.

# ---Tại sao dung AIDL ---------------------------------------------------------

AIDL does nothing but lets the system to generate the boilerplate code that hides the binder IPC detail.

1. you don't need IPC (i.e., your client and server stay in the same process), you don't need AIDL;
2. If you want to write the boilerplate code yourself for IPC, you don't need AIDL;

# ---Dùng AIDL ntn ----------------------------------------------------------------

# Ref:

<https://android.jlelse.eu/android-binder-framework-8a28fb38699a>

<http://rts.lab.asu.edu/web_438/project_final/CSE_598_Android_Architecture_Binder.pdf>

<http://strayinsights.blogspot.com/2018/04/android-binder-tutorial_25.html>

<https://sujaiantony.wordpress.com/2011/12/28/an-android-101-an-overview-on-binder-framework/>

<https://events.static.linuxfound.org/images/stories/slides/abs2013_gargentas.pdf>

<https://techmaster.vn/posts/33649/so-sanh-giua-rpc-va-messaging>

<http://www.inspirel.com/articles/RPC_vs_Messaging.html>

<https://daynhauhoc.com/t/thac-mac-posix-la-gi/41170>

<https://en.wikipedia.org/wiki/Interface_description_language>

<https://stackoverflow.com/questions/40994800/should-i-use-aidl-or-not>

<https://android.jlelse.eu/android-aidl-937daf89e685>

<https://viblo.asia/p/android-interface-definition-language-aidl-eW65GEMRZDO>

<https://developer.android.com/guide/components/aidl>

<https://www.synacktiv.com/posts/systems/binder-transactions-in-the-bowels-of-the-linux-kernel.html>

<https://www.androiddesignpatterns.com/2013/07/binders-window-tokens.html>