Efficient Design with XPC

Session 702

Damien Sorresso
Senior Software Engineer

Agenda

- Focus on performance
    - Architectural design
    - Implementation
- New features
- Efficient usage patterns

Last Time at WWDC…

What Is XPC?

- Service bootstrapping and IPC
- Easily factor app into services
- Services deployed within app bundle

Why Use XPC?

- Separate address space
  - Fault isolation
  - Different privileges/entitlements
  - Least-required privilege
- Completely managed lifecycle
  - Less boilerplate

Bundled Services

- ✓ Ship in app
- ✓ Stateless
- ✓ Fully managed lifecycle
- ✓ App Store-rific

launchd Services

- ✗ Require installation
- ✗ App Store no me gusta
- ✗ More boilerplate
- ✓ Run as root
- ✓ Independent of app

APIs

APIs

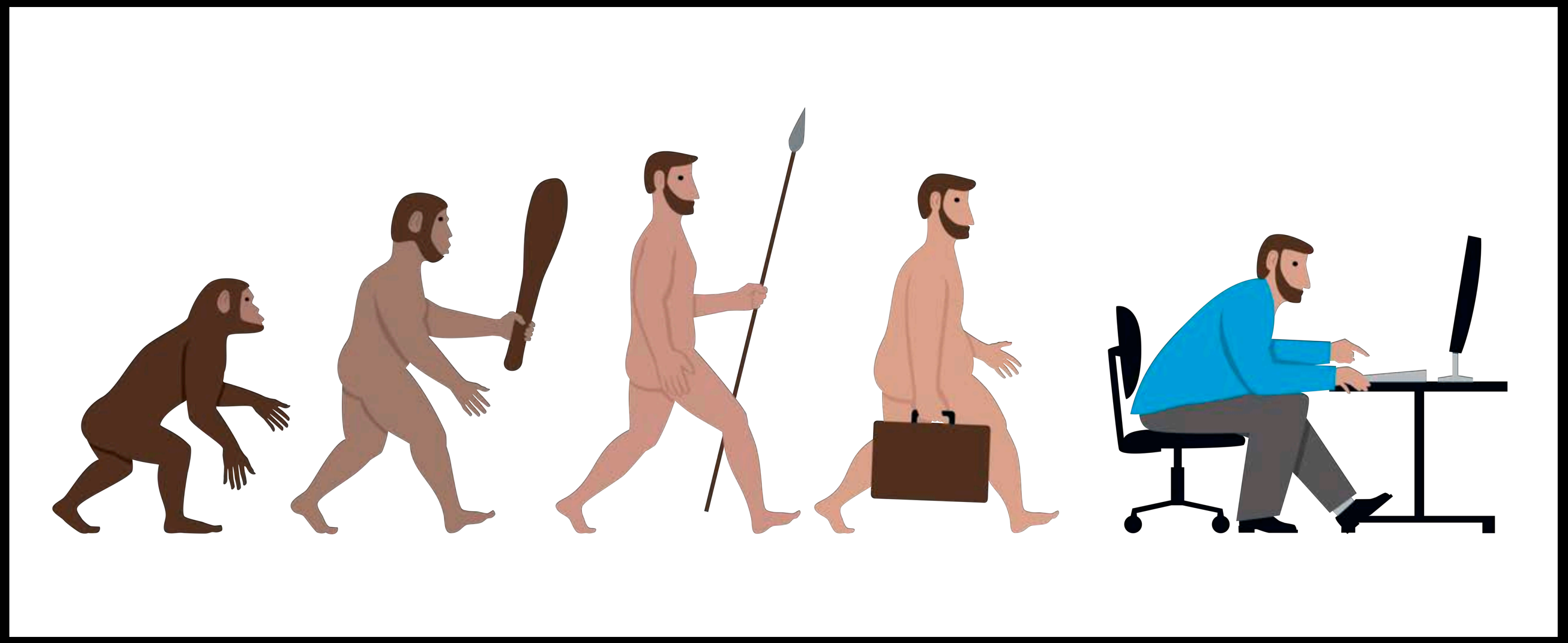
NSXPCConnection
libxpc
libdispatch
libobjc / ARC

Architecture

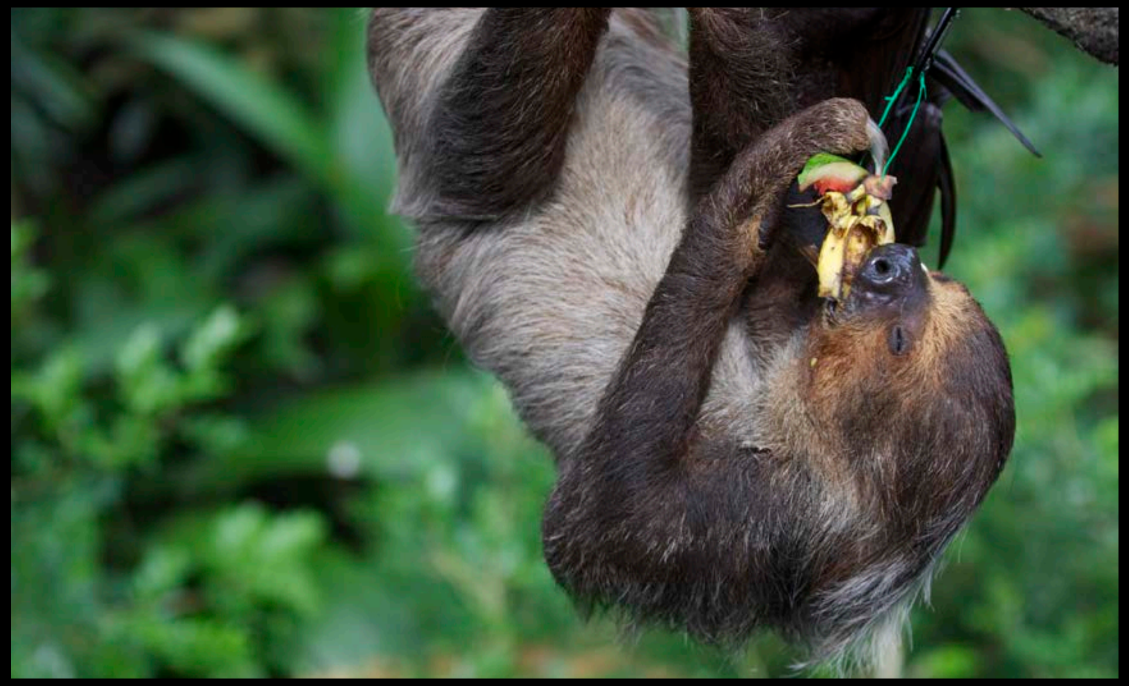
Architectural Goals

Architectural Goals

Avoid long-running processes



Adapt to resource availability



Lazy initialization

XPC Events

- System events demand-launch

XPC Events

- System events demand-launch
  - IOKit matching

XPC Events

- System events demand-launch
  - IOKit matching
  - BSD notifications—notify(3)

XPC Events

NEW

- System events demand-launch
  - IOKit matching
  - BSD notifications—notify(3)
  - CFDistributedNotifications

XPC Events

- System events demand-launch
    - IOKit matching
    - BSD notifications—notify(3)
    - CFDistributedNotifications
- Only available to launchd services

XPC Events

In the launchd.plist

```
<key>LaunchEvents</key>
<dict>
    <key>com.apple.iokit.matching</key>
    <dict>
        <key>com.mycompany.device-attach</key>
        <dict>
            <key>idProduct</key>
            <integer>2794</integer>
            <key>idVendor</key>
            <integer>725</integer>
            <key>IOProviderClass</key>
            <string>IOUSBDevice</string>
            <key>IOMatchLaunchStream</key>
            <true/>
        </dict>
    </dict></dict>
```

XPC Events

In the launchd.plist

```
<key>LaunchEvents</key>
<dict>
    <key>com.apple.iokit.matching</key>
    <dict>
        <key>com.mycompany.device-attach</key>
        <dict>
            <key>idProduct</key>
            <integer>2794</integer>
            <key>idVendor</key>
            <integer>725</integer>
            <key>IOProviderClass</key>
            <string>IOUSBDevice</string>
            <key>IOMatchLaunchStream</key>
            <true/>
        </dict>
    </dict></dict>
```

XPC Events

In the launchd.plist

```
<key>LaunchEvents</key>
<dict>
    <key>com.apple.iokit.matching</key>
    <dict>
        <key>com.mycompany.device-attach</key>
        <dict>
            <key>idProduct</key>
            <integer>2794</integer>
            <key>idVendor</key>
            <integer>725</integer>
            <key>IOProviderClass</key>
            <string>IOUSBDevice</string>
            <key>IOMatchLaunchStream</key>
            <true/>
        </dict>
    </dict></dict>
```

XPC Events

In the launchd.plist

```
<key>LaunchEvents</key>
<dict>
    <key>com.apple.iokit.matching</key>
    <dict>
        <key>com.mycompany.device-attach</key>
        <dict>
            <key>idProduct</key>
            <integer>2794</integer>
            <key>idVendor</key>
            <integer>725</integer>
            <key>IOProviderClass</key>
            <string>IOUSBDevice</string>
            <key>IOMatchLaunchStream</key>
            <true/>
        </dict>
    </dict></dict>
```

XPC Events
Consumption

```
xpc_set_event_stream_handler(“com.apple.iokit.matching”, q,
        ^(xpc_object_t event) {
    // Every event has the key XPC_EVENT_KEY_NAME set to a string that
    // is the name you gave the event in your launchd.plist.
    const char *name = xpc_dictionary_get_string(event, XPC_EVENT_KEY_NAME);

    // IOKit events have the IORegistryEntryNumber as a payload.
    uint64_t id = xpc_dictionary_get_uint64(event, “IOMatchLaunchServiceID”);

    // Reconstruct the node you were interested in here using the IOKit
    // APIs.
});
```

XPC Events

Consumption

```
xpc_set_event_stream_handler(“com.apple.iokit.matching”, q,
        ^(xpc_object_t event) {
    // Every event has the key XPC_EVENT_KEY_NAME set to a string that
    // is the name you gave the event in your launchd.plist.
    const char *name = xpc_dictionary_get_string(event, XPC_EVENT_KEY_NAME);

    // IOKit events have the IORegistryEntryNumber as a payload.
    uint64_t id = xpc_dictionary_get_uint64(event, “IOMatchLaunchServiceID”);

    // Reconstruct the node you were interested in here using the IOKit
    // APIs.
});
```

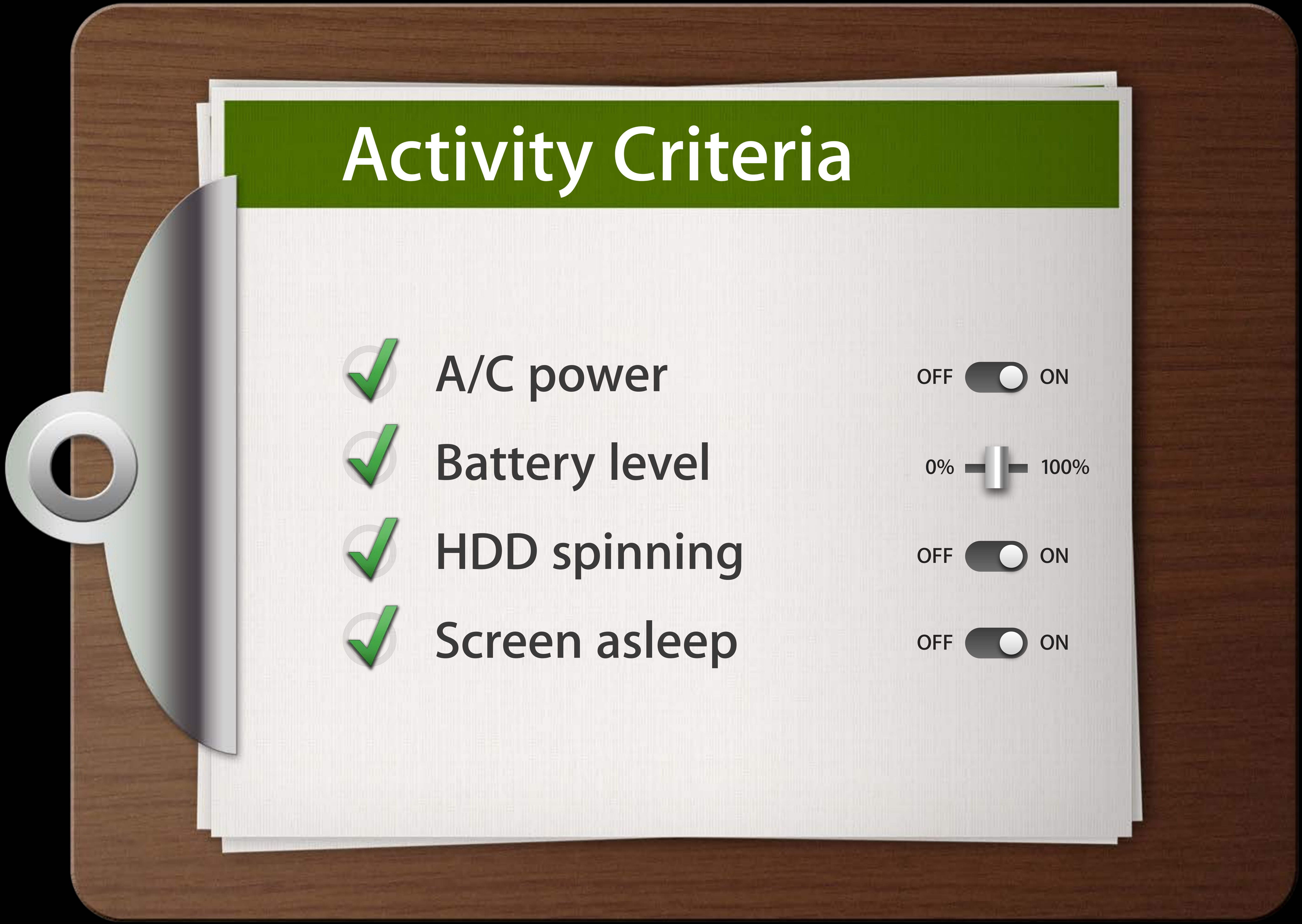
Centralized Task Scheduling

XPC activity

- Opportunistic task scheduling
- Defer work until a “good time”
- Minimize disruption to user experience
- Maximize efficient use of battery

Activity Types

Type	Example	Interrupted when…
Maintenance	Garbage collection	User begins using machine
Utility	Fetch network data	Resources become scarce



Activity Criteria

✓	A/C power	OFF [✓ ON]
✓	Battery level	0% — 100%
✓	HDD spinning	OFF [✓ ON]
✓	Screen asleep	OFF [✓ ON]

XPC Activity

launchd and XPC services

- Persist across launches
- System state change launches on-demand
- Activity picks up where it left off

XPC Activity

```
xpc_object_t criteria = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_INTERVAL, 5 * 60);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_GRACE_PERIOD, 10 * 60);

// Activity handler runs on background queue.
xpc_activity_register(“com.mycompany.myapp.myactivity”, criteria,
        ^(xpc_activity_t activity) {
    id data = createDataFromPeriodicRefresh();

    // Continue the activity asynchronously to update the UI.
    xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_CONTINUE);
    dispatch_async(dispatch_get_main_queue(), ^{
        updateViewWithData(data);
        xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_DONE);
    });
});
```

XPC Activity

```
xpc_object_t criteria = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_INTERVAL, 5 * 60);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_GRACE_PERIOD, 10 * 60);

// Activity handler runs on background queue.
xpc_activity_register(“com.mycompany.myapp.myactivity”, criteria,
        ^(xpc_activity_t activity) {
    id data = createDataFromPeriodicRefresh();

    // Continue the activity asynchronously to update the UI.
    xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_CONTINUE);
    dispatch_async(dispatch_get_main_queue(), ^{
        updateViewWithData(data);
        xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_DONE);
    });
});
```

XPC Activity

```
xpc_object_t criteria = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_INTERVAL, 5 * 60);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_GRACE_PERIOD, 10 * 60);

// Activity handler runs on background queue.
xpc_activity_register(“com.mycompany.myapp.myactivity”, criteria,
        ^(xpc_activity_t activity) {
    id data = createDataFromPeriodicRefresh();

    // Continue the activity asynchronously to update the UI.
    xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_CONTINUE);
    dispatch_async(dispatch_get_main_queue(), ^{
        updateViewWithData(data);
        xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_DONE);
    });
});
```

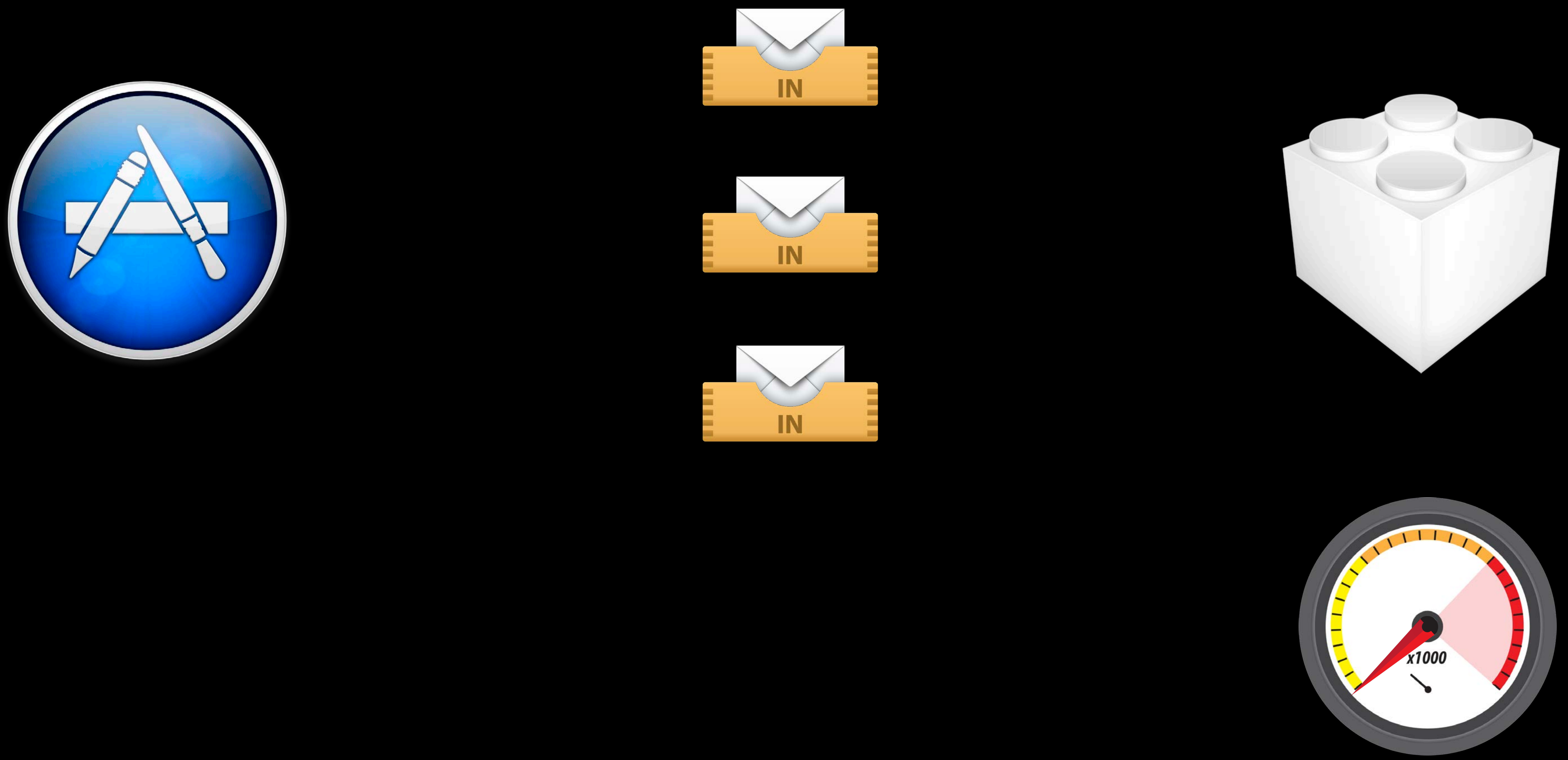
XPC Activity

```
xpc_object_t criteria = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_INTERVAL, 5 * 60);
xpc_dictionary_set_int64(criteria, XPC_ACTIVITY_GRACE_PERIOD, 10 * 60);

// Activity handler runs on background queue.
xpc_activity_register(“com.mycompany.myapp.myactivity”, criteria,
        ^(xpc_activity_t activity) {
    id data = createDataFromPeriodicRefresh();

    // Continue the activity asynchronously to update the UI.
    xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_CONTINUE);
    dispatch_async(dispatch_get_main_queue(), ^{
        updateViewWithData(data);
        xpc_activity_set_state(activity, XPC_ACTIVITY_STATE_DONE);
    });
});
```

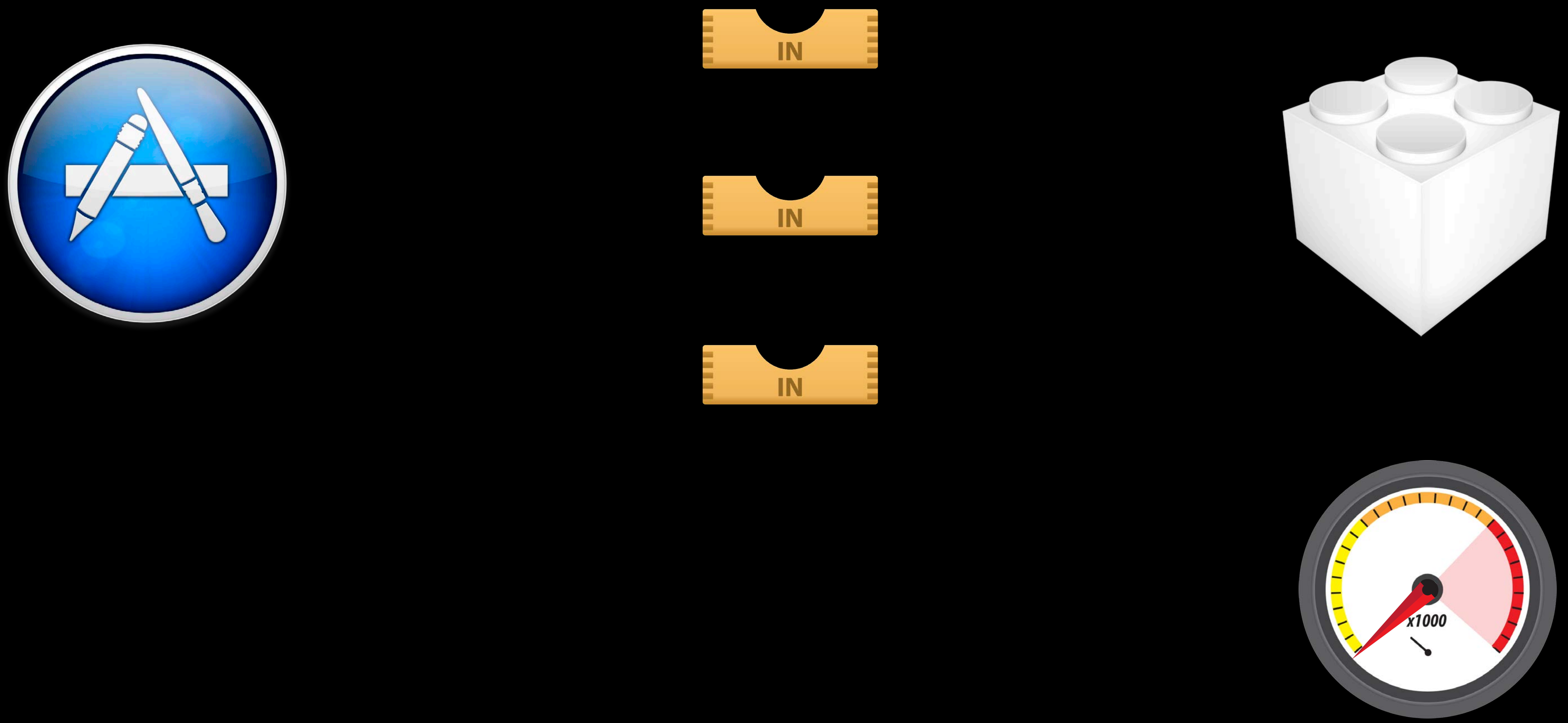
Runtime

Service Lifecycle

Service Lifecycle



Service Lifecycle



Service Lifecycle

- Service launches on-demand
- System stops service as needed
    - App quits
    - Memory pressure
    - Idle/lack of use

XPC Runtime

Sudden termination

- Sudden termination management
- Disabled when request is live
- Enabled when reply is sent

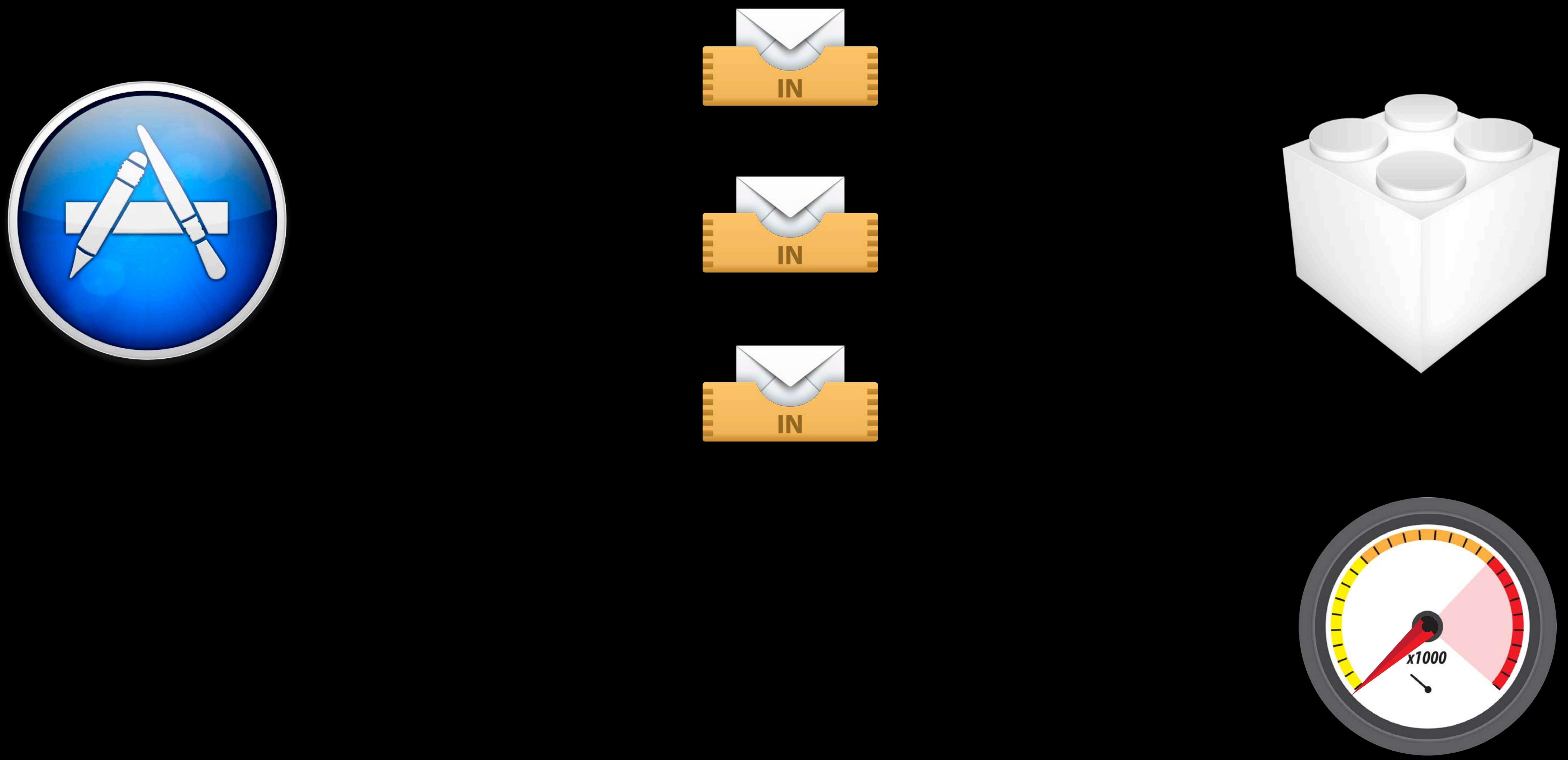
Importance Boosting

NEW

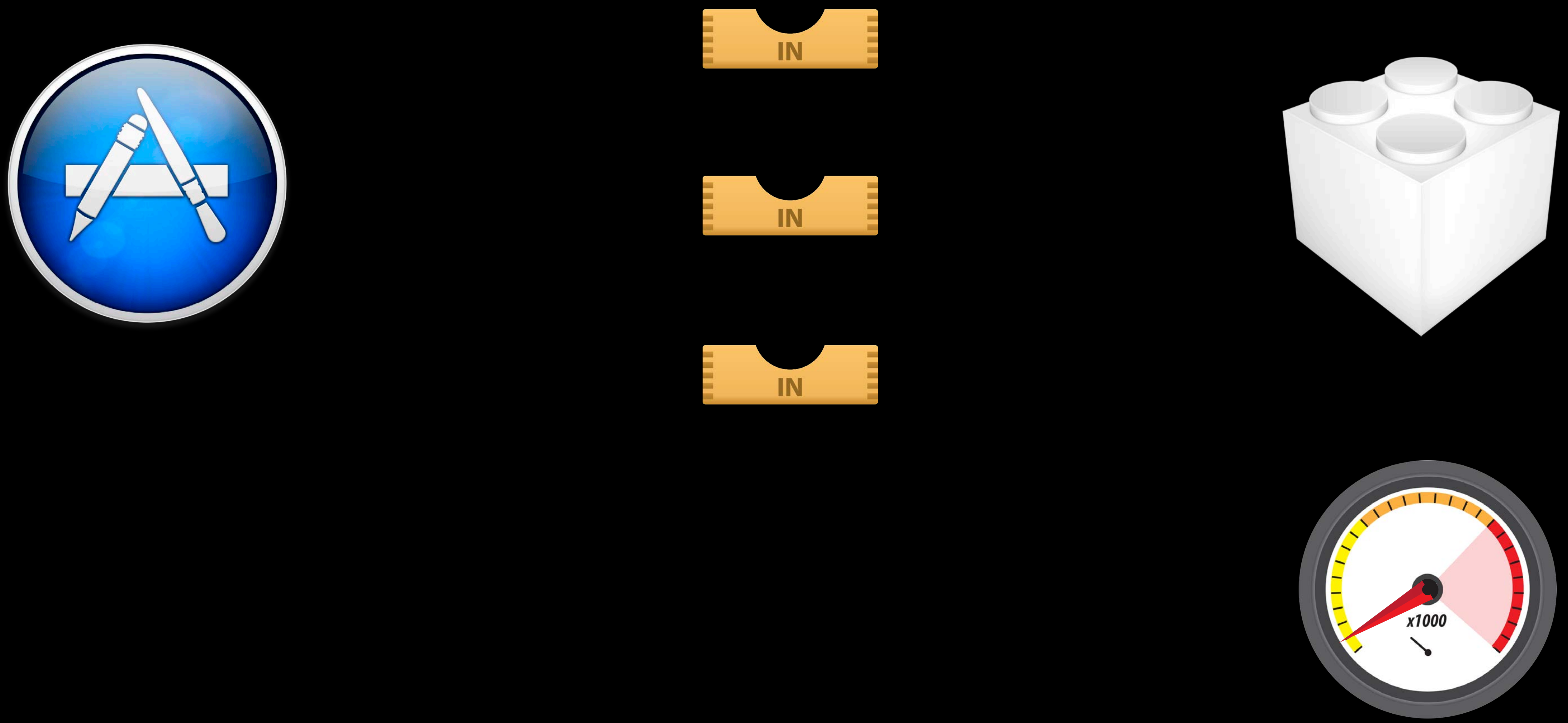
- Bundled services
  - Background priority by default
  - UI app requests boost priority
- Minimize disruption of user experience

Boost Lifetime

Boost Lifetime



Boost Lifetime



Importance Boosting

NEW

- launchd services can opt in
- ProcessType plist key

ProcessType Values

Value	Contention Behavior	Use when…
Adaptive	Contend with apps when doing work on their behalf	app uses XPC to communicate with launchd job
Background	Never contend with apps	app has no dependency on launchd job’s work
Interactive	Always contend with apps	the sun explodes
Standard	Default value Similar behavior as previous releases	you just want to purty-up your plist

Persisting a Boost

```
// Transfers boost from ‘message’ to ‘reply’.
xpc_object_t reply = xpc_dictionary_create_reply(message);

// Set up work using information in ‘message’ and do it
// asynchronously.
setup_work_context_with_message(message);
dispatch_async(queue, ^{
    // After work is done, populate ‘reply‘ with result.
    xpc_connection_send_message(conn, reply);

    // Boost drops after runtime has sent ‘reply’.
});
```

Persisting a Boost

```
// Transfers boost from ‘message’ to ‘reply’.
xpc_object_t reply = xpc_dictionary_create_reply(message);

// Set up work using information in ‘message’ and do it
// asynchronously.
setup_work_context_with_message(message);
dispatch_async(queue, ^{
    // After work is done, populate ‘reply‘ with result.
    xpc_connection_send_message(conn, reply);

    // Boost drops after runtime has sent ‘reply’.
});
```

Persisting a Boost

```
// Transfers boost from ‘message’ to ‘reply’.
xpc_object_t reply = xpc_dictionary_create_reply(message);

// Set up work using information in ‘message’ and do it
// asynchronously.
setup_work_context_with_message(message);
dispatch_async(queue, ^{
    // After work is done, populate ‘reply‘ with result.
    xpc_connection_send_message(conn, reply);

    // Boost drops after runtime has sent ‘reply’.
});
```

Multiple Replies

With anonymous connections

Multiple Replies

With anonymous connections

```
xpc_connection_t ac = xpc_connection_create(NULL, NULL);
```

Sending Connections

```
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_connection(message, “backchannel”, ac);
xpc_connection_send_message(connection, message);
```

Sending Connections

```
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_connection(message, “backchannel”, ac);
xpc_connection_send_message(connection, message);
```

Multiple Replies

With anonymous connections

```
xpc_connection_set_event_handler(backchannel, ^(xpc_object_t conn) {
    __block size_t seqno = 0;
    // Accept remote connection.
    xpc_connection_set_event_handler(conn, ^(xpc_object_t message) {
        // Receive the reply.
        bool done = do_stuff_with_message(message, ++seqno);
        if (done) {
            xpc_connection_cancel(conn);
        }
    });

    xpc_connection_resume(conn);
});

xpc_connection_resume(backchannel);
```

Multiple Replies

With anonymous connections

```
xpc_connection_set_event_handler(backchannel, ^(xpc_object_t conn) {
    __block size_t seqno = 0;
    // Accept remote connection.
    xpc_connection_set_event_handler(conn, ^(xpc_object_t message) {
        // Receive the reply.
        bool done = do_stuff_with_message(message, ++seqno);
        if (done) {
            xpc_connection_cancel(conn);
        }
    });

    xpc_connection_resume(conn);
});

xpc_connection_resume(backchannel);
```

Multiple Replies

Server side

```
xpc_connection_t backchannel =
        xpc_dictionary_create_connection(message, “backchannel”);
xpc_connection_set_event_handler(backchannel, ^(xpc_object_t event) {
    // This is only used for sending.
    xpc_connection_cancel(backchannel);
});
xpc_connection_resume(backchannel);
```

Multiple Replies

Server side

```
xpc_connection_t backchannel =
        xpc_dictionary_create_connection(message, “backchannel”);
xpc_connection_set_event_handler(backchannel, ^(xpc_object_t event) {
    // This is only used for sending.
    xpc_connection_cancel(backchannel);
});
xpc_connection_resume(backchannel);
```

Multiple Replies

Server side

```
xpc_connection_set_event_handler(conn, ^(xpc_object_t message) {
    // Boost dropped when last block is done and ‘message’ is
    // released.
    dispatch_apply(5, queue, ^{
        // Do stuff with ‘message’ and populate ‘replyi’.
        do_stuff_and_populate(message, replyi);
        xpc_object_t replyi = xpc_dictionary_create(NULL, NULL, 0);
        xpc_connection_send_message(backchannel, replyi);
    });
});
```

Multiple Replies

Server side

```
xpc_connection_set_event_handler(conn, ^(xpc_object_t message) {
    // Boost dropped when last block is done and ‘message’ is
    // released.
    dispatch_apply(5, queue, ^{
        // Do stuff with ‘message’ and populate ‘replyi’.
        do_stuff_and_populate(message, replyi);
        xpc_object_t replyi = xpc_dictionary_create(NULL, NULL, 0);
        xpc_connection_send_message(backchannel, replyi);
    });
});
```

Big Data

NEW

- Runtime recognizes large data objects
- Object is sent with minimal copies
- Copy-less fast path available

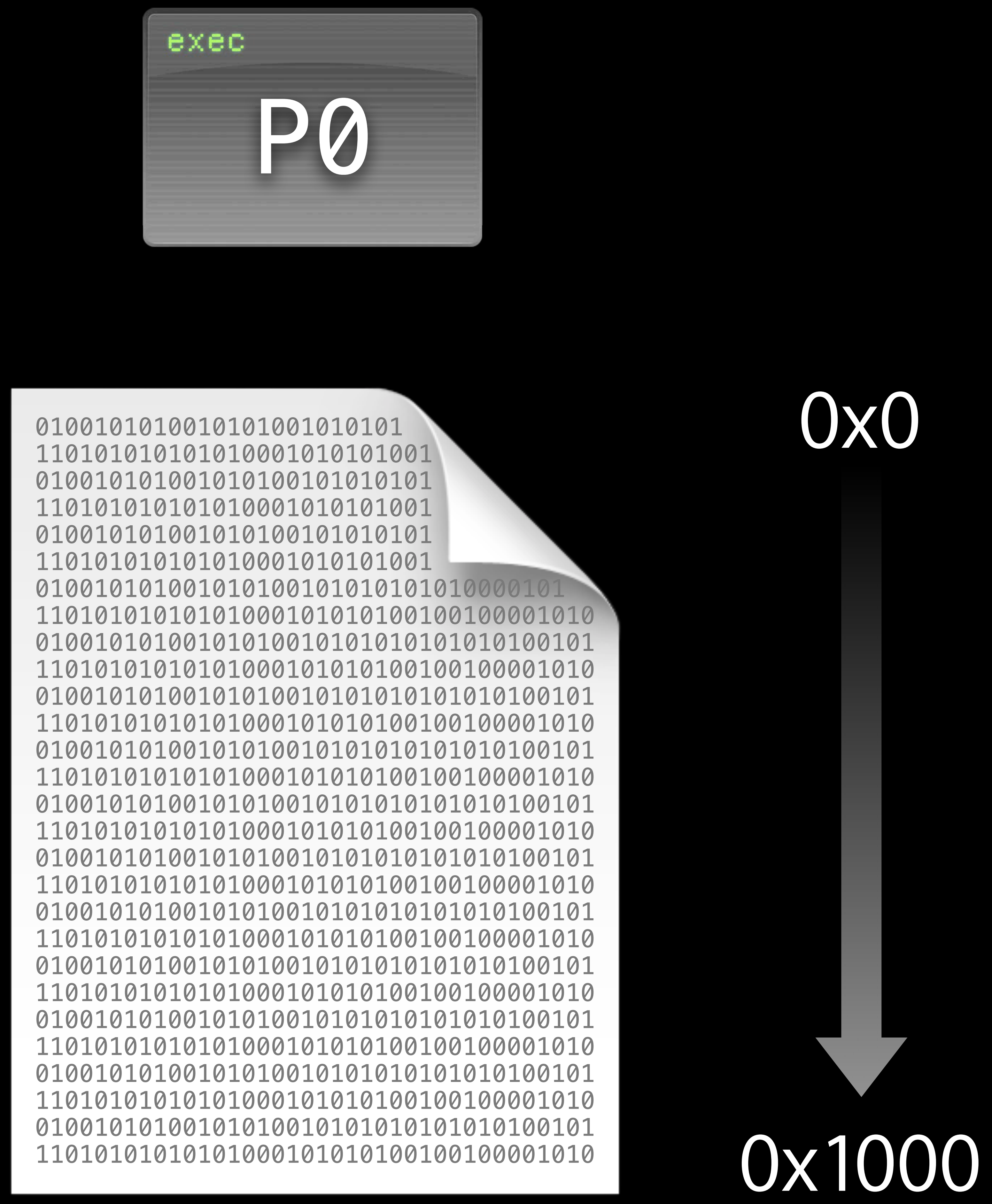
Big Data

How it works

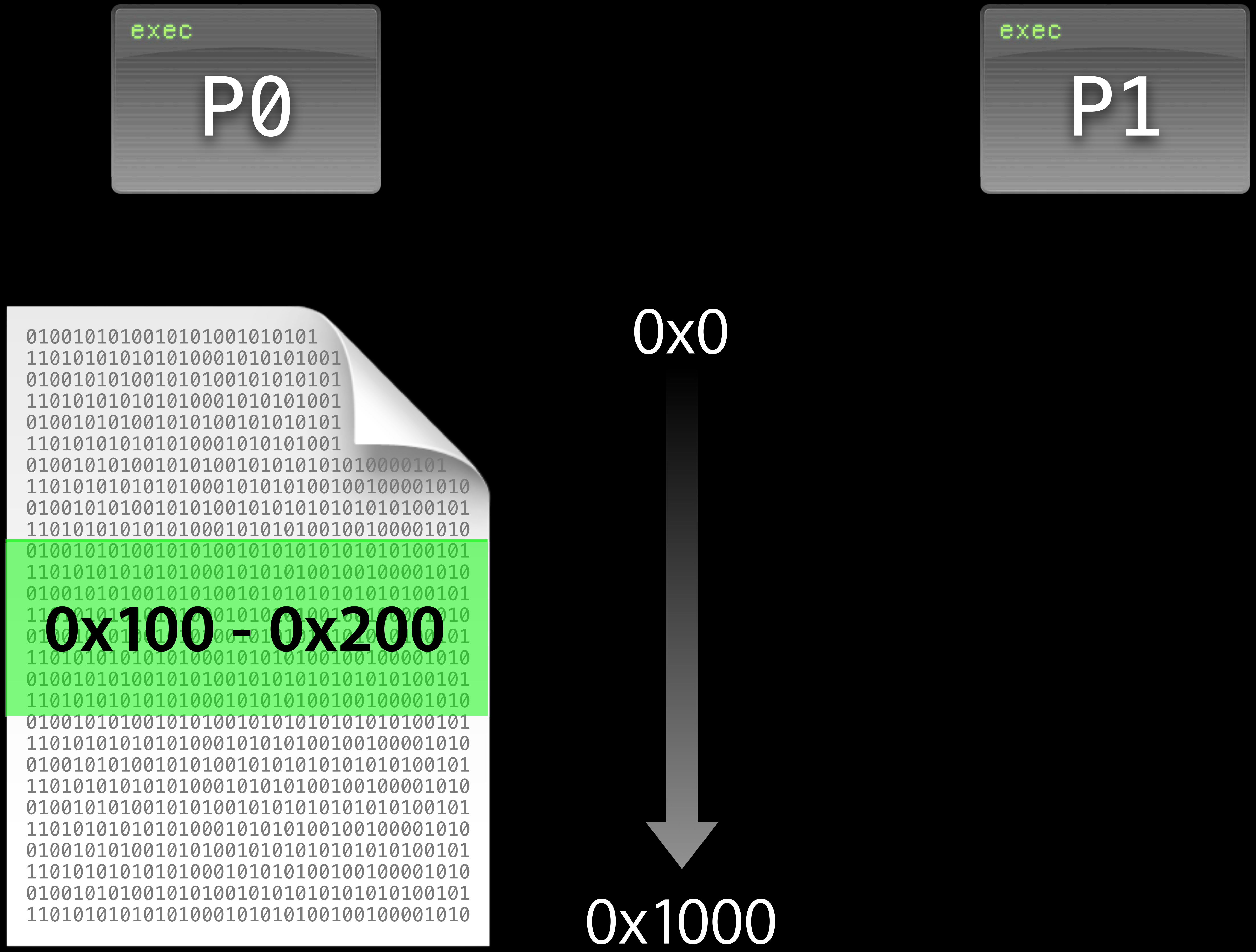
- Deals with VM object backing data
- Shares copy-on-write
- Page-granular

Safely Sharing Memory

Safely Sharing Memory



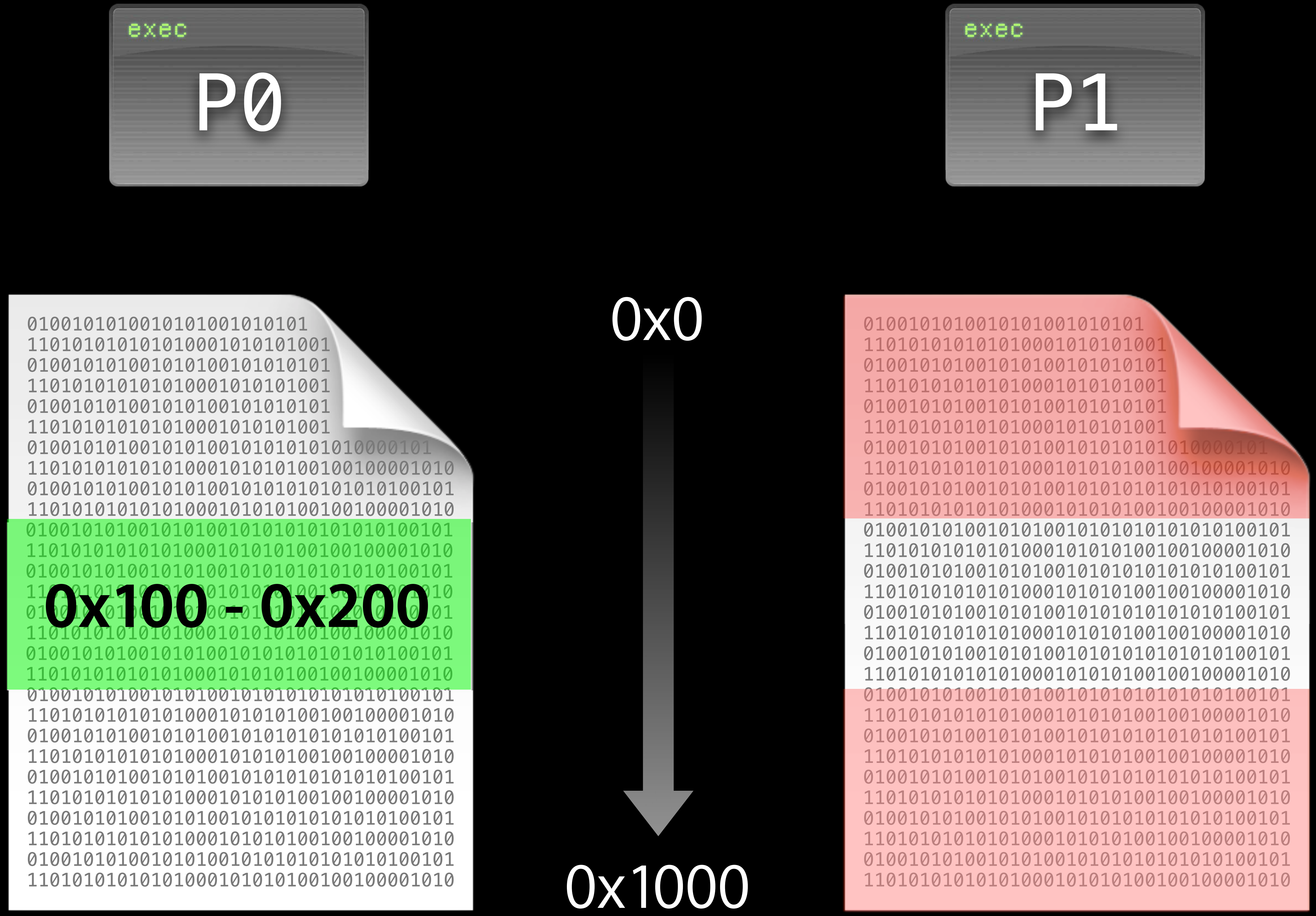
Safely Sharing Memory



🟩 malloc(3)ed

🟥 Random Data

Safely Sharing Memory



🟩 malloc(3)ed

🟥 Random Data

Big Data

Minimizing copies

- Create dispatch data object
- DISPATCH_DATA_DESTRUCTOR_MUNMAP
- Wrap dispatch data in XPC data

Sending Big Data

```
// Points to large mmap(2)ed region of memory.
void *buff;
// Size of region pointed to by ‘buff’.
size_t sz;

dispatch_data_t ddata = dispatch_data_create(buff, sz,
        DISPATCH_TARGET_QUEUE_DEFAULT, DISPATCH_DATA_DESTRUCTOR_MUNMAP);

// ‘buff’ is now owned by ‘ddata’.
xpc_object_t xdata = xpc_data_create_with_dispatch_data(ddata);

// Send ‘xdata’ in a message.
// Look Ma! No copies!
```

Sending Big Data

```
// Points to large mmap(2)ed region of memory.
void *buff;
// Size of region pointed to by ‘buff’.
size_t sz;

dispatch_data_t ddata = dispatch_data_create(buff, sz,
        DISPATCH_TARGET_QUEUE_DEFAULT, DISPATCH_DATA_DESTRUCTOR_MUNMAP);

// ‘buff’ is now owned by ‘ddata’.
xpc_object_t xdata = xpc_data_create_with_dispatch_data(ddata);

// Send ‘xdata’ in a message.
// Look Ma! No copies!
```

Sending Big Data

```
// Points to large mmap(2)ed region of memory.
void *buff;
// Size of region pointed to by ‘buff’.
size_t sz;

dispatch_data_t ddata = dispatch_data_create(buff, sz,
        DISPATCH_TARGET_QUEUE_DEFAULT, DISPATCH_DATA_DESTRUCTOR_MUNMAP);

// ‘buff’ is now owned by ‘ddata’.
xpc_object_t xdata = xpc_data_create_with_dispatch_data(ddata);

// Send ‘xdata’ in a message.
// Look Ma! No copies!
```

Big Data

Using NSXPCConnection

- Dispatch data toll-free bridged with NSData
- Not subclasses
- Not no-copy
- Any of the following
    - Mapped
    - NSDataDeallocatorVM
    - NSDataDeallocatorMunmap

Receive Fast Path

- Message-receive much faster
- Substantially reduced allocations and copies
- Drain messages more quickly

Receive Fast Path

Eligible messages

- No out-of-line types
  - File descriptors
  - Shared memory

Receive Fast Path

Forcing the slow path

- Some actions force full unpack
  - Copying
  - xpc_dictionary_apply(3)
- xpc_dictionary_get_value(3)
- Accessing nested containers
- Modifying dictionary

Timeouts

Timeouts

- XPC APIs have no support for timeouts
- Unneeded in most cases

Timeouts

- Local-machine and network IPC different
- Kernel is not unreliable medium
- Server should always be responsive

Timeouts

Masking bugs

- Can confuse behavior expectations
  - Operation may have timed out
  - Server may have a bug
- Expressed to client identically

Server-Side Timeout

- If service does network operations
  - Have it manage timeout
  - Return ETIMEDOUT (or similar) to client
- Lack of response indicates server bug

Timeouts

When they are needed

- Hard deadline for response
- When transit time makes a difference
  - “Real time”
  - Derived from desired throughput rate
  - fps, Hz, etc.
- Not arbitrary

Debugging Tips

Debugging

- Xcode enhancements
  - Transparent breakpoints
  - Debugging services just works
  - Copy Files destination
- imptrace(1) for debugging boosts

Debugging

- Connection-invalid indicates configuration error
- Make sure service target…
  - is dependency of app target
  - is in Copy Files build phase
- Make sure CFBundleIdentifier matches service name

API Misuse

- XPC API is defensive
- Aborts on detectable corruption/misuse
  - Retain count underflow
  - Obvious API misuse
  - xpc_abort(3)
- Issues illegal instruction—SIGILL

API Misuse

- Look in Application-Specific Information
- Under lldb
    - xpc_debugger_api_misuse_info(3)
    - Returns (const char *)

Crash Report

```
Exception Type:  EXC_BAD_INSTRUCTION (SIGILL)
Exception Codes: 0x0000000000000001, 0x0000000000000000

Application Specific Information:
API MISUSE: Over-release of an object
```

Crash Report

```
Exception Type:  EXC_BAD_INSTRUCTION (SIGILL)
Exception Codes: 0x0000000000000001, 0x0000000000000000

Application Specific Information:
API MISUSE: Over-release of an object
```

lldb

```
Program received signal EXC_BAD_INSTRUCTION, Illegal instruction/operand.
0x000000010012b25e in _xpc_api_misuse ()
(lldb) p (char *)xpc_debugger_api_misuse_info()
$1 = 0x7fff5fbff908 "XPC API Misuse: Over-release of object."
(lldb)
```

lldb

```
Program received signal EXC_BAD_INSTRUCTION, Illegal instruction/operand.
0x000000010012b25e in _xpc_api_misuse ()
(lldb) p (char *)xpc_debugger_api_misuse_info()
$1 = 0x7fff5fbff908 "XPC API Misuse: Over-release of object."
(lldb)
```

libxpc Assertions

- Might see messages in system log

```
assertion failed: 13A476z: libxpc.dylib + 2794 [0B05C709-16BA-3C31-
ACC6-1234774ED777]: 0x11
```

- Indicates unexpected but non-fatal error
- File a bug

More Information

Paul Danbold
Core OS Technology Evangelist
danbold@apple.com

Documentation
xpc(3)
/usr/include/xpc
Daemons and Services Programming Guide

Apple Developer Forums
devforums.apple.com

Related Sessions

Debugging with Xcode	Pacific Heights Wednesday 2:00PM	🟧
Building Efficient OS X Apps	Nob Hill Tuesday 4:30PM	🟦

Labs

XPC and GCD Lab	CoreOS Lab B Tuesday 3:15PM
CoreOS Lab Open Hours	CoreOS Labs A/B Friday 2:00PM

WWDC2013