D-Bus in the Kernel

FOSDEM 2014, Brussels, Belgium

January 2014

Who?

Greg Kroah-Hartman,
Daniel Mack,
Lennart Poettering,
Kay Sievers

with help from Tejun Heo

Most newer OS designs started around powerful IPC

Mach, QNX, Hurd, ...

Linux only had IPC primitives (sockets, fifos, shared memory)

D-Bus is powerful IPC

Method Call Transactions,

D-Bus is powerful IPC

Method Call Transactions, Signals,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials, File Descriptor Passing,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials, File Descriptor Passing, Language agnostic,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials, File Descriptor Passing, Language agnostic, Network transparency,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials, File Descriptor Passing, Language agnostic, Network transparency, no trust required,

D-Bus is powerful IPC

Method Call Transactions, Signals, Properties, OO, Broadcasting, Discovery, Introspection, Policy, Activation, Synchronization, Type-safe Marshalling, Security, Monitoring, exposes APIs/not streams, Passing of Credentials, File Descriptor Passing, Language agnostic, Network transparency, no trust required, High-level error concept...

D-Bus has limitations

D-Bus has limitations

Suitable only for control, not payload

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

Hookup with security frameworks happens in userspace

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

Hookup with security frameworks happens in userspace

Activatable bus services are independent from other system services

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

Hookup with security frameworks happens in userspace

Activatable bus services are independent from other system services

Codebase is a bit too baroque, XML, ...

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

Hookup with security frameworks happens in userspace

Activatable bus services are independent from other system services

Codebase is a bit too baroque, XML, ...

No race-free exit-on-idle bus activated services

D-Bus has limitations

Suitable only for control, not payload

It's inefficient (10 copies, 4 complete validations, 4 context switches per duplex method call transaction)

Credentials one can send/recv are limited

No implicit timestamping

Not available in early boot, initrd, late boot

Hookup with security frameworks happens in userspace

Activatable bus services are independent from other system services

Codebase is a bit too baroque, XML, ...

No race-free exit-on-idle bus activated services

...

D-Bus is fantastic, solves real problems

D-Bus is fantastic, solves real problems

Right approach: good concepts, generic, comprehensive, covers all areas

D-Bus is fantastic, solves real problems

Right approach: good concepts, generic, comprehensive, covers all areas

Established, it's the single most used local, high-level IPC system on Linux, bindings for most languages

D-Bus is fantastic, solves real problems

Right approach: good concepts, generic, comprehensive, covers all areas

Established, it's the single most used local, high-level IPC system on Linux, bindings for most languages

Used in init system (regardless if systemd or Upstart), the desktops, embedded, ...

kdbus

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

Activation is identical to activation of other services

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

Activation is identical to activation of other services

Userspace is much simpler, no XML, . . .

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

Activation is identical to activation of other services

Userspace is much simpler, no XML, . . .

Priority queues, . . .

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

Activation is identical to activation of other services

Userspace is much simpler, no XML, . . .

Priority queues, . . .

Race-free exit-on-idle for bus activated services

kdbus

Suitable for large data (GiB!), zero-copy, optionally reusable

It's efficient (2 or fewer copies, 2 validations, 2 context switches per duplex methd call transaction)

Credentials sent along are comprehensive (uid, pid, gid, selinux label, pid starttime, tid, comm, tid comm, argv, exe, cgroup, caps, audit, . . .)

Implicit timestamping

Always available, from earliest boot to latest shutdown

Open for LSMs to hook into from the kernel side

Activation is identical to activation of other services

Userspace is much simpler, no XML, . . .

Priority queues, . . .

Race-free exit-on-idle for bus activated services

. . .

Overview

Overview

Receiver buffers

Overview

Receiver buffers

Single copy to destination(s)

Overview

Receiver buffers

Single copy to destination(s)

Method call windows

Overview

Receiver buffers

Single copy to destination(s)

Method call windows

Name registry

memfds

memfds

File descriptors for memory regions

memfds

File descriptors for memory regions

Zero Copy!

memfds

File descriptors for memory regions

Zero Copy!

Sealing

memfds

File descriptors for memory regions

Zero Copy!

Sealing

At 512K zero copy is faster than single copy

memfds

File descriptors for memory regions

Zero Copy!

Sealing

At 512K zero copy is faster than single copy

(a bit like Android ashmem)

Signal Broadcasting

Signal Broadcasting

Bloom Filters

Signal Broadcasting

Bloom Filters

Every broadcast message includes bloom filter (calculated by sender) that contains all supported matches, kernel will then simply check receiver bloom filter mask (calculated by receiver) against it.

Signal Broadcasting

Bloom Filters

Every broadcast message includes bloom filter (calculated by sender) that contains all supported matches, kernel will then simply check receiver bloom filter mask (calculated by receiver) against it.

Bloom filter uses SipHash, but kernel doesn't care

Policy:

Policy:

No XML, only simple ACL policy attached to service names

Policy:

No XML, only simple ACL policy attached to service names

More fine-grained access control needs to be done in userspace, but it's much easier

Policy:

No XML, only simple ACL policy attached to service names

More fine-grained access control needs to be done in userspace, but it's much easier

Use capability checks!

Policy:

No XML, only simple ACL policy attached to service names

More fine-grained access control needs to be done in userspace, but it's much easier

Use capability checks!

PolicyKit

Differences in Userspace:

Differences in Userspace:

GVariant used for marshalling (O(1) random access to struct and array fields)

Differences in Userspace:

GVariant used for marshalling (O(1) random access to struct and array fields)

Setup, activation, policy management, driver, proxy lives in systemd

Differences in Userspace:

GVariant used for marshalling (O(1) random access to struct and array fields)

Setup, activation, policy management, driver, proxy lives in systemd

New libsystemd-bus client library: waaaaay nicer to use – but not portable to non-Linux

Proxy: provides compatibility with dbus1 sockets

Proxy: provides compatibility with dbus1 sockets

Synthesizes obsolete AcquiredName, LostName, Hello messages

Proxy: provides compatibility with dbus1 sockets

Synthesizes obsolete AcquiredName, LostName, Hello messages

Implements XML policy

Proxy: provides compatibility with dbus1 sockets

Synthesizes obsolete AcquiredName, LostName, Hello messages

Implements XML policy

Activated on demand, exits on idle

Proxy: provides compatibility with dbus1 sockets

Synthesizes obsolete AcquiredName, LostName, Hello messages

Implements XML policy

Activated on demand, exits on idle

Remarshals gvariant/dbus1

Driver: translates driver method calls into ioctl calls

Driver: translates driver method calls into ioctl calls

org.freedesktop.DBus pseudo-service is a real service on kdbus

Driver: translates driver method calls into ioctl calls

org.freedesktop.DBus pseudo-service is a real service on kdbus

Note that driver signals are synthesized on client side, so the driver only handles method calls

Driver: translates driver method calls into ioctl calls

org.freedesktop.DBus pseudo-service is a real service on kdbus

Note that driver signals are synthesized on client side, so the driver only handles method calls

Activated on demand, exits on idle

Activation: new .busname unit type in systemd

Activation: new .busname unit type in systemd

Identical to .socket unit types for socket activation

Activation: new .busname unit type in systemd

Identical to .socket unit types for socket activation

dbus1 bus activation files still supported, but only for clients connecting via the proxy

libsystemd-bus

libsystemd-bus

New client library, designed to be easy to use

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

Lots of convenience functions

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

Lots of convenience functions

Focus on converting errno from/to bus errors

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

Lots of convenience functions

Focus on converting errno from/to bus errors

Connect to container, connect to remote

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

Lots of convenience functions

Focus on converting errno from/to bus errors

Connect to container, connect to remote

Credentials include units, slices, sessions, ...

libsystemd-bus

New client library, designed to be easy to use

Not portable to non-Linux

Assemble and parse messages with format strings

Handles introspection, signal dispatching, method vtables, properties, object manager

Lots of convenience functions

Focus on converting errno from/to bus errors

Connect to container, connect to remote

Credentials include units, slices, sessions, ...

It's probably what you want to use when you hack on system level software, and up

Android binder

Android binder

Some similar technical concepts, different semantics

Android binder

Some similar technical concepts, different semantics

No name registry, no broadcasts, no ordering

When?

When?

It's all in kdbus git, and systemd git, now!

When?

It's all in kdbus git, and systemd git, now!

Compile-time switch in systemd

When?

It's all in kdbus git, and systemd git, now!

Compile-time switch in systemd

We hope to get kdbus reviewed and accepted into the kernel in 2014

When?

It's all in kdbus git, and systemd git, now!

Compile-time switch in systemd

We hope to get kdbus reviewed and accepted into the kernel in 2014

gdbus support coming soon, also libdbus1 support

When?

It's all in kdbus git, and systemd git, now!

Compile-time switch in systemd

We hope to get kdbus reviewed and accepted into the kernel in 2014

gdbus support coming soon, also libdbus1 support

Google for git repos!

Outlook

Outlook

Sandboxing

Outlook

Sandboxing

Yielding CPU time to destination

Outlook

Sandboxing

Yielding CPU time to destination

Priority inheritance

Outlook

Sandboxing

Yielding CPU time to destination

Priority inheritance

Priority queues

...

That's all, folks!