BlueChi

A multi-nodes systemd service controller

https://github.com/containers/bluechi

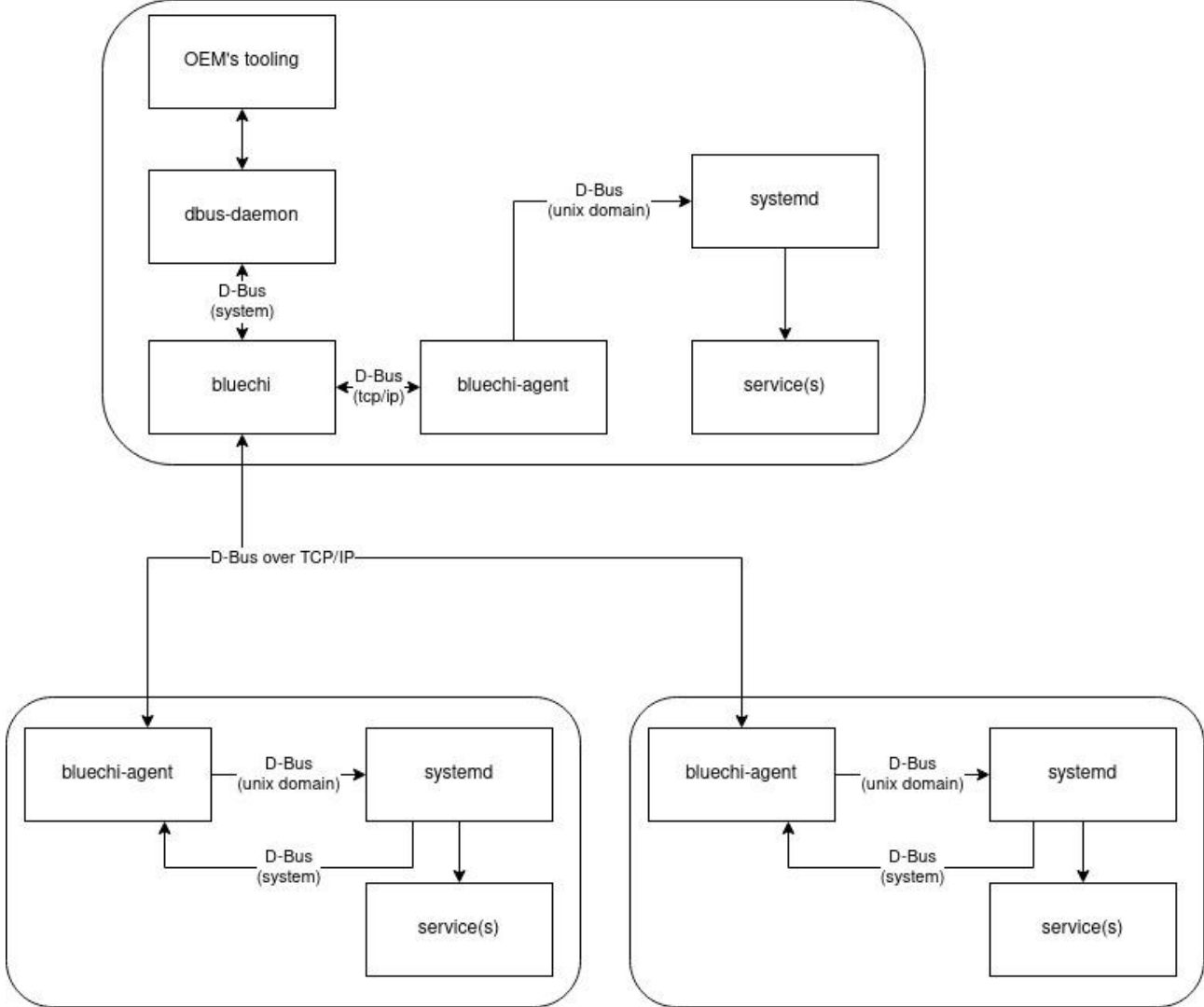
Goals

- Systemd service controller
  - Application can be containerized - doesn’t have to be
  - Integrates well with quadlet (and thus with podman)
- Multi-nodes support
- Dbus-based API
  - Use dbus to make BlueChi do something
  - Use dbus to hear back from the services
- Cross-node service dependency
- Part of the monitoring stack

Architecture

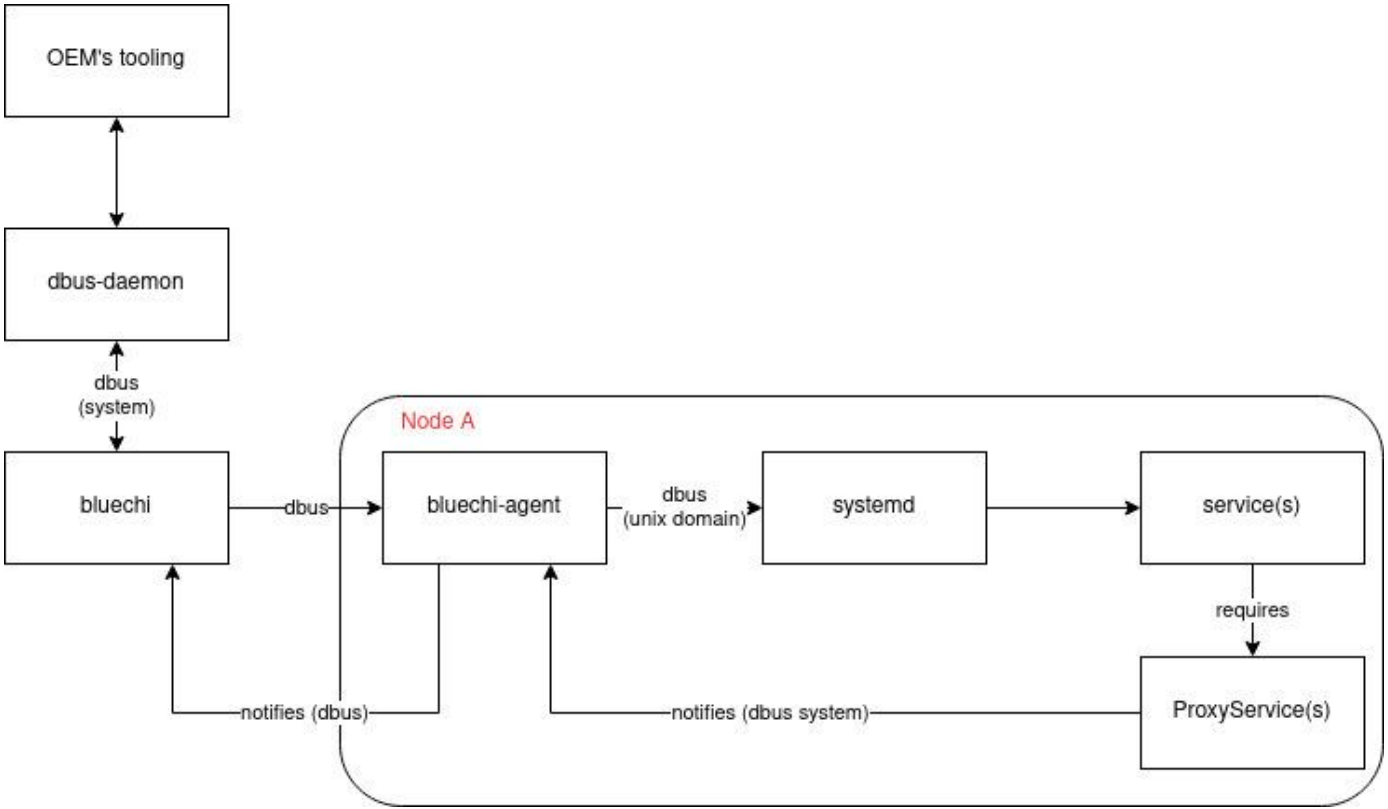
- bluechi
  - The application controlling systemd services via the agent running on different nodes
  - This can be run next to a bluechi-agent
- bluechi-agent
  - The agent running on each node
  - Receives notification from bluechi
  - Talks to systemd via its dbus unix domain socket
- bluechictl
  - A CLI that interacts with bluechi
  - More a convenience tool than something that would be integrated into another program
    - The proper way being to use the dbus API directly

Global overview

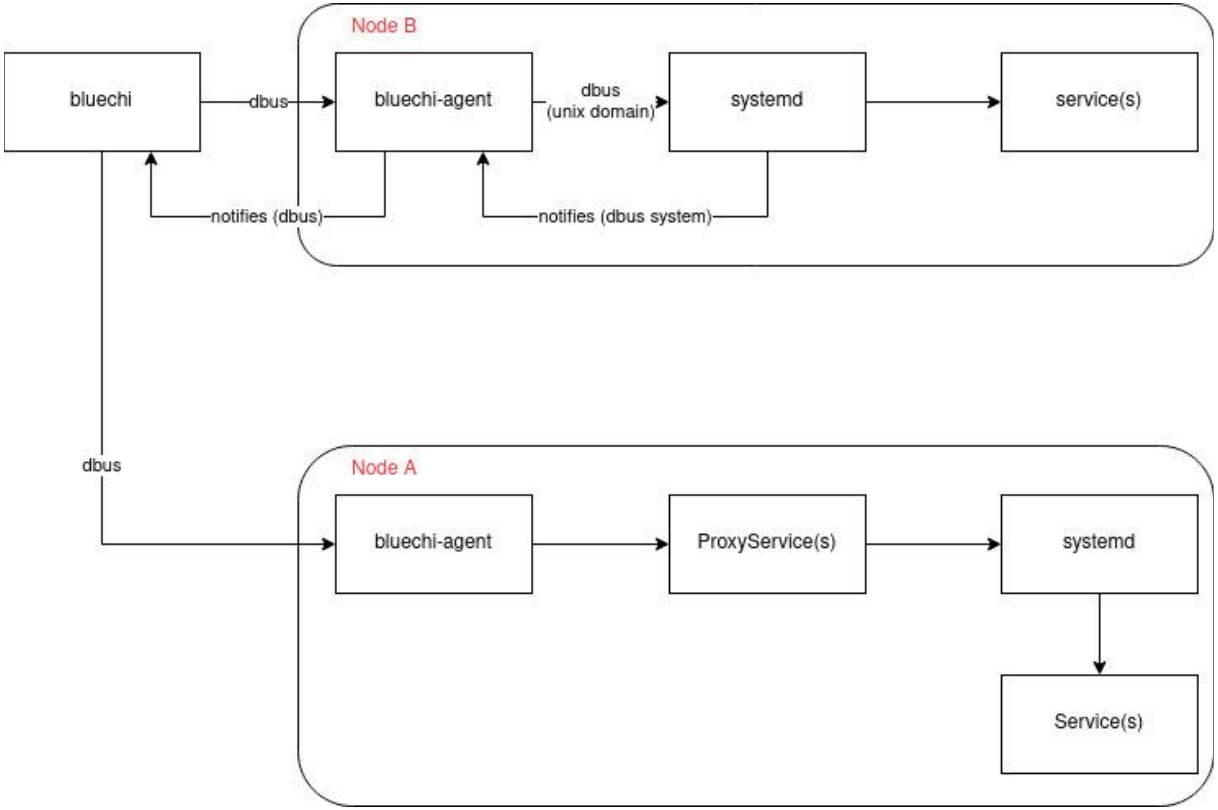


Cross-device service dependencies

Cross-node dependency support (Part 1)



Cross-node dependency support (Part 2)



Using BlueChi - examples

These examples are in python, but anything interacting with dbus will work

List active services (1) - raw dbus call

```python
#!/usr/bin/python3
# SPDX-License-Identifier: CC0-1.0

from collections import namedtuple
import dasbus.connection
bus = dasbus.connection.SystemMessageBus()

NodeUnitInfo = namedtuple("NodeUnitInfo", ["node", "name",
                                           "description", "load_state", "active_state", "sub_state", "follower",
                                           "object_path", "job_id", "job_type", "job_object_path"])

manager = bus.get_proxy("org.eclipse.bluechi",  "/org/eclipse/bluechi")
units = manager.ListUnits()
for u in units:
    info = NodeUnitInfo(*u)
    if info.active_state == "active" and info.name.endswith(".service"):
        print(f"Node: {info.node}, Unit: {info.name}")
```

List active services (2) - python bluechi library

```python
#!/usr/bin/env python
# SPDX-License-Identifier: CC0-1.0
#
# vim:sw=4:ts=4:et
from bluechi.api import Manager

for unit in Manager().list_units():
    # unit[node, name, description, load_state, active_state, ...]
    if unit[4] == "active" and unit[1].endswith(".service"):
        print(f"Node: {unit[0]}, Unit: {unit[1]}")
```

List active services (2)

```
$ sudo python doc/api-examples/list-active-services.py
Node: laptop, Unit: systemd-user-sessions.service
Node: laptop, Unit: rpc-statd-notify.service
Node: laptop, Unit: power-profiles-daemon.service
Node: laptop, Unit: systemd-update-utmp.service
Node: laptop, Unit: rngd.service
Node: laptop, Unit: rtkit-daemon.service
Node: laptop, Unit: NetworkManager-wait-online.service
Node: laptop, Unit: netcf-transaction.service
Node: laptop, Unit: bolt.service
Node: laptop, Unit: systemd-journald.service
```

In this example, bluechi talks to a bluechi-agent running on a full Fedora desktop system

Start service

```
#!/usr/bin/python3
# SPDX-License-Identifier: CC0-1.0

from datetime import datetime
import sys
from dasbus.connection import SystemMessageBus
from dasbus.loop import EventLoop

bus = SystemMessageBus()

if len(sys.argv) != 3:
    print("No node name and unit supplied")
    sys.exit(1)

node_name = sys.argv[1]
unit_name = sys.argv[2]

manager = bus.get_proxy("org.eclipse.bluechi", "/org/eclipse/bluechi")
node_path = manager.GetNode(node_name)
node = bus.get_proxy("org.eclipse.bluechi", node_path)

loop = EventLoop()


def job_removed(id, job_path, node_name, unit, result):
    if job_path == my_job_path:
        run_time = (datetime.utcnow() - start_time).total_seconds()
        print(f"Started '{unit}' on node '{node_name}' with result '{result}' in {run_time*1000:.1f} msec")
        loop.quit()


start_time = datetime.utcnow()

manager.JobRemoved.connect(job_removed)
my_job_path = node.StartUnit(unit_name, "replace")
loop.run()
```

Start service

```
$ systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: inactive (dead)
       Docs: man:httpd.service(8)

Jul 10 15:27:40 flame.pingoured.fr systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jul 10 15:27:40 flame.pingoured.fr systemd[1]: Started httpd.service - The Apache HTTP Server.
Jul 10 15:27:40 flame.pingoured.fr httpd[2658984]: Server configured, listening on: port 80
Jul 10 15:28:00 flame.pingoured.fr systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Jul 10 15:28:02 flame.pingoured.fr systemd[1]: httpd.service: Deactivated successfully.
Jul 10 15:28:02 flame.pingoured.fr systemd[1]: Stopped httpd.service - The Apache HTTP Server.
$ sudo python doc/api-examples/start-unit.py laptop httpd.service
Started 'httpd.service' on node 'laptop' with result 'done' in 139.1 msec
$ systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: active (running) since Mon 2023-07-10 15:28:29 CEST; 1s ago
       Docs: man:httpd.service(8)
   Main PID: 2661114 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 23233)
     Memory: 15.4M
        CPU: 117ms
     CGroup: /system.slice/httpd.service
             ├─2661114 /usr/sbin/httpd -DFOREGROUND
             ├─2661115 /usr/sbin/httpd -DFOREGROUND
             ├─2661117 /usr/sbin/httpd -DFOREGROUND
             ├─2661118 /usr/sbin/httpd -DFOREGROUND
             └─2661119 /usr/sbin/httpd -DFOREGROUND

Jul 10 15:28:29 flame.pingoured.fr systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jul 10 15:28:29 flame.pingoured.fr systemd[1]: Started httpd.service - The Apache HTTP Server.
Jul 10 15:28:29 flame.pingoured.fr httpd[2661114]: Server configured, listening on: port 80
```

Some monitoring

```
# hirtectl monitor
Monitor path: /org/containers/hirte/monitor/3
Subscribing to node '*' and unit '*'
[laptop] *
        Unit created (reason: virtual)
[rpi4] *
        Unit created (reason: virtual)
[rpi4] hirte-agent.service
        Unit properties changed (Interface: org.freedesktop.systemd1.Service)
        StatusText:
[rpi4] hirte-agent.service
        Unit properties changed (Interface: org.freedesktop.systemd1.Unit)
        ActiveState: active
[rpi4] *
        Unit state changed (reason: virtual)
        Active: inactive (agent-offline)
[rpi4] *
        Unit removed (reason: virtual)
```

Metrics

```
# hirtectl metrics enable
Done
# hirtectl metrics listen
Waiting for metrics signals...
[laptop] Agent systemd StartUnit job on httpd.service net measured time: 167.0ms
[laptop] Job /org/containers/hirte/job/7 to start unit httpd.service:
        Hirte job gross measured time: 0.0ms
        Unit net start time (from properties): 0.0ms
[laptop] Agent systemd StopUnit job on httpd.service net measured time: 1030.0ms
```

```
# hirtectl start laptop httpd.service
Unit httpd.service start operation result: done
# hirtectl stop laptop httpd.service
Unit httpd.service stop operation result: done
# time systemctl start httpd

real    0m0.168s
user    0m0.009s
sys     0m0.009s
# time systemctl stop httpd

real    0m1.052s
user    0m0.009s
sys     0m0.008s
#
```

Ballpark estimate

Metrics does not support the cross-node service dependency at the moment - it is being worked on

Assumptions

Assumptions

- BlueChi does not handle the “initial setup” of the system
  - Ie: systems boot to it target/default state - BlueChi handles the transitions
- BlueChi does not know service dependencies
  - This is handled at the systemd level

Testing BlueChi

Testing BlueChi

- On Fedora, CentOS/RHEL:
  - Soon to be available: `dnf install bluechi`
    - hirte is still available there
  - For CentOS/AutoSD/RHEL:
    - From the EPEL project
    - From the AutoSD or Automotive SIG repository
- Using Containers:
  - AutoSD Development Container: https://gitlab.com/CentOS/automotive/container-images/-/tree/main/images/autosd

Questions?

Some references:

- https://www.redhat.com/en/blog/running-containers-cars
- https://www.redhat.com/en/blog/introducing-hirte-deterministic-multi-node-service-controller
- https://www.youtube.com/watch?v=8RiRiviSVqM - Hirte - Multi-node service orchestration for edge - DevConf.CZ 2023

- https://github.com/containers/bluechi/
- https://bluechi.readthedocs.io/en/latest/