cosylab
CONTROL SYSTEM LABORATORY

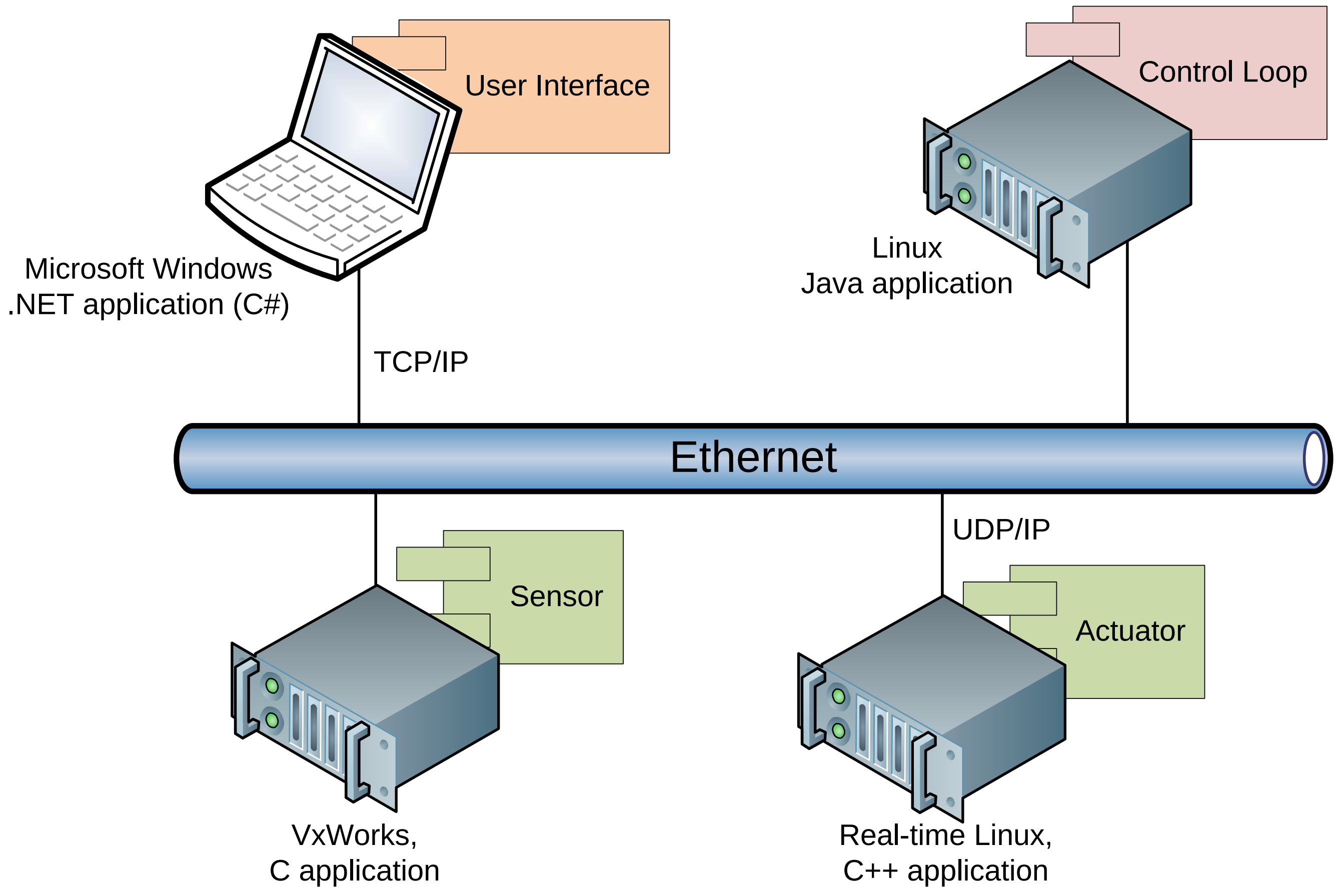
Longevity of Accelerator Control System Middleware

Klemen Žagar
klemen.zagar@cosylab.com

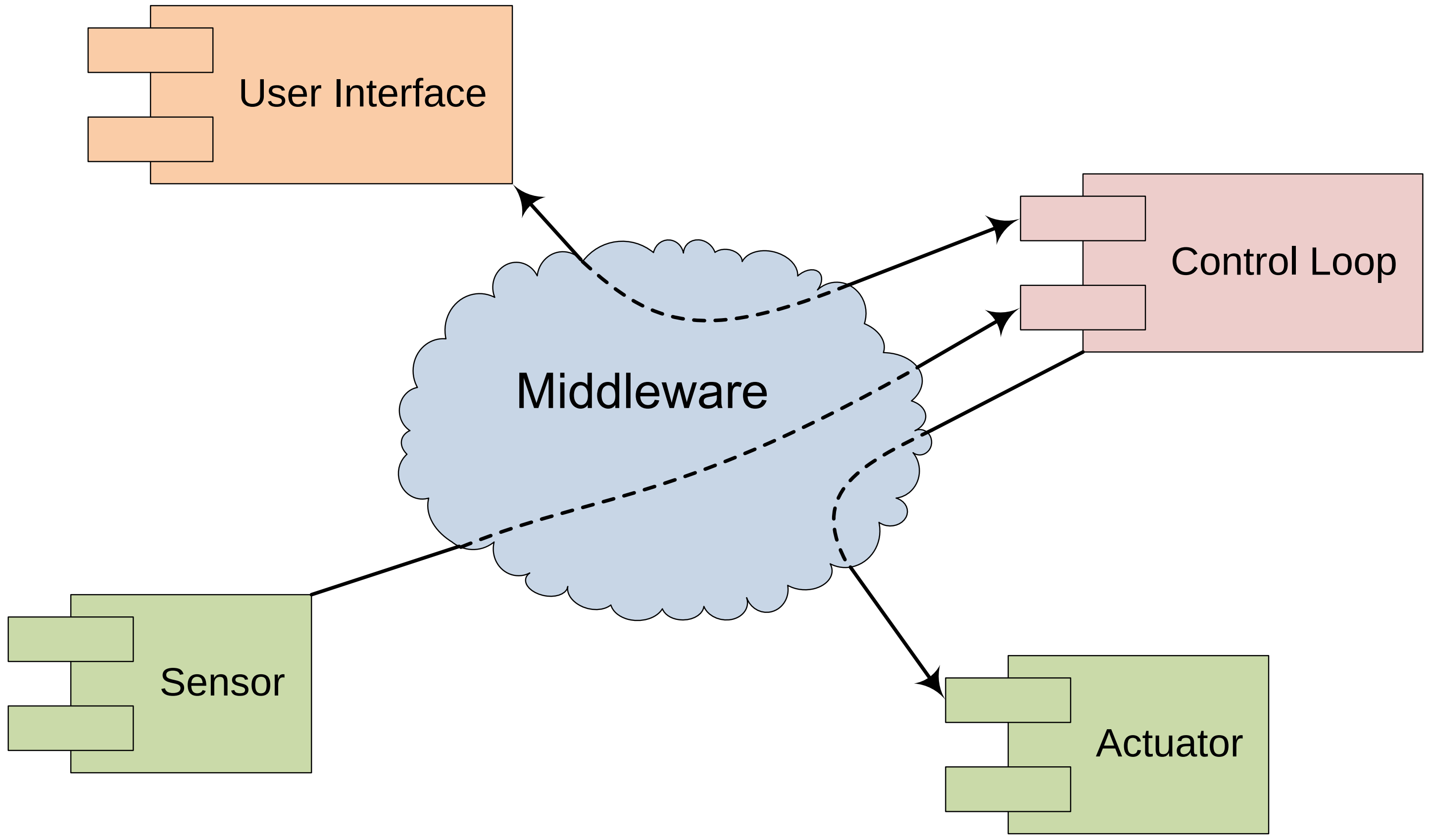
Overview

- What is *middleware*?
- Why is middleware particularly important for control system's longevity?
- What middleware is available today?
- What will be available in 30 years?
- What forces can render a middleware obsolete, and how to defend against them?

What is *middleware*?



What is *middleware*?



What does middleware do?

- Mapping of abstract interface definitions (e.g., CORBA IDL) to programming languages (e.g., Java, C++ etc. *bindings*).
- Marshaling/serialization of parameters and data structures (i.e., converting to/from binary form).
- Managing network connections (connect, disconnect, send, receive, reconnect).
- Concurrency: allowing ‘server’ processes (targets of invocations) to handle multiple requests from several network connections concurrently.
- Management of send/receive buffers.
- Implementation of higher-level protocols, such as *reliable multicast*.

Middleware and longevity

- If a single component becomes obsolete, it can be re-implemented and re-deployed.
  - Re-deployment without system-wide restart is usually supported by modern middleware.
- Software components don’t interact directly with each other, but via middleware.
  - Problems in middleware affect **the entire system**.
  - If middleware becomes obsolete, **all** software components need to be modified in some way!
- Gradual “upgrade” is difficult.
  - Requires “gateways” between the “old” and the “new” middleware.
  - Gateways can become performance/availability bottleneck, might be difficult to configure and maintain, etc.
- Some middlewares are more than just that – they are also control system frameworks.
  - Replacing them requires even more intervention in the code.

Today's middleware (2009)

- Message-centric:
  - JMS

- Data-centric:
  - EPICS
  - Data Distribution Service (DDS)

- Remote procedure call:
  - ICE
  - CORBA
  - Java RMI
  - Sun RPC
  - DCOM
  - Web Services
  - Microsoft Windows Communications Foundation (WCF)

Why does middleware “age”?

(Applies to other technologies as well.)

- No longer supported for a particular operating system, hardware platform or programming language.
- Significantly outperformed by a new middleware in any of these categories:
  - Convenience of development.
  - Performance (throughput, latency, scalability).
  - Resource consumption (memory, CPU, power).
  - User experience.
  - (Useful) functionality.
  - Maintenance costs.
- No longer “cool”/”hype”.
- No longer needed (e.g., radical shift in underlying technology).

Anti-aging measures

Anti-aging measures: open source

- Open source:
  - "*Use the source, Luke*"
    Can check the *ultimate manual* in case of unexpected behavior.
  - Can fix bugs or adopt to new platform without waiting for the supplier.
- Drawbacks?
  - Uncoordinated efforts lead to many half-solutions to a problem, which are usually incompatible. (A cultural thing.)
- When open source is not an option:
  - Ensure an escrow agreement (get source code if vendor ceases support).

Anti-aging measures: availability of support

- Support should be available:
  - Training (in the beginning and when recruiting new personnel).
  - Consulting (tackling difficult challenges in the right way).
  - Development (adding features, fixing bugs).
- Support offered by:
  - The community. Free, but no guarantees on availability or quality.
  - Commercial support. Not free, but with guarantees.
- Community support:
  - Size of the community.
  - Eagerness of users to help others (cultural thing).
  - Check mailing lists, forums, web sites, ... (state today and the trend).
- Commercial support:
  - Market structure (monopoly vs. free market).
  - Whom to choose? Check references in the domain.
- Example: EPICS
  - Community support (tech-talk mailing list)
  - Commercial support (Observatory Sciences, Cosylab, Alceli, etc.)

Anti-aging measures: economy of scale

- More users => more “budget” for development:
  - drives down “cost per user”,
  - increases the number of available features.

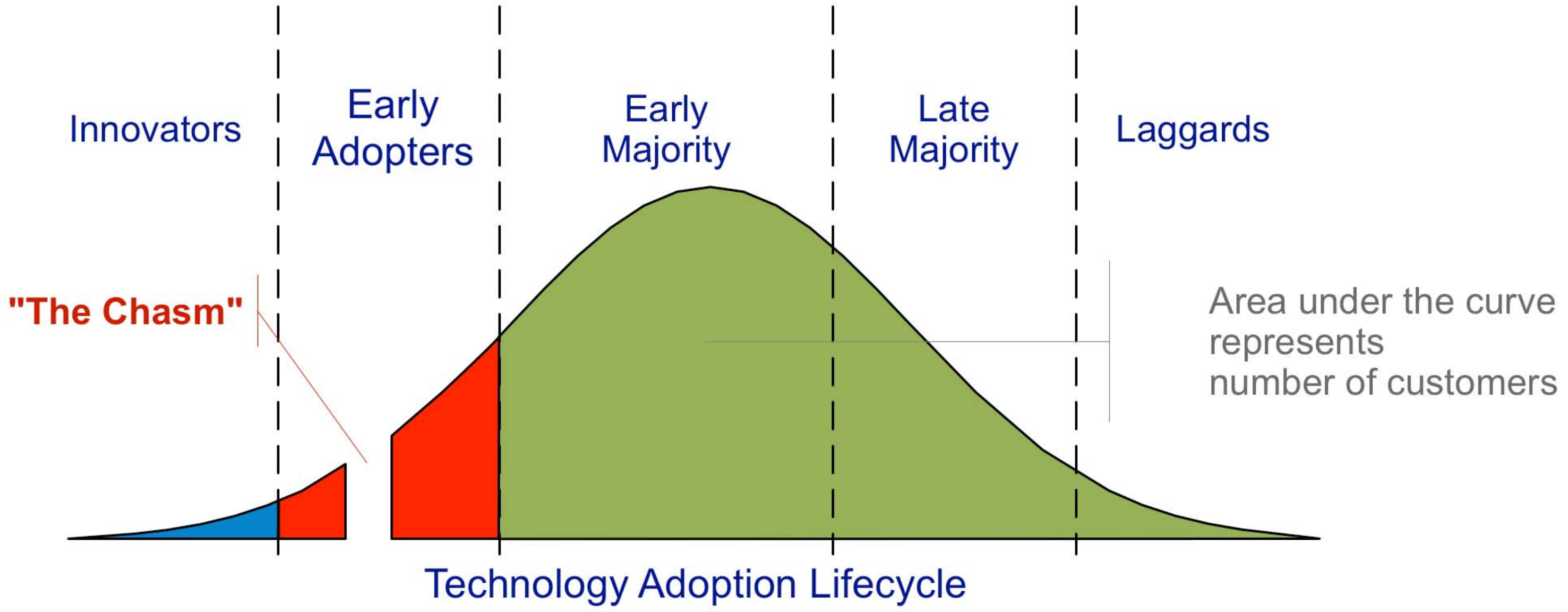
- Increased chance that:
  - a particular platform/device is supported,
  - someone else already fixed the bugs,
  - someone took the time to write documentation.

- Less likely to “go out of business”.

- Ideally: other users are in the same boat as you.
  - E.g., large experimental facilities with similar “time-to-live” requirements.

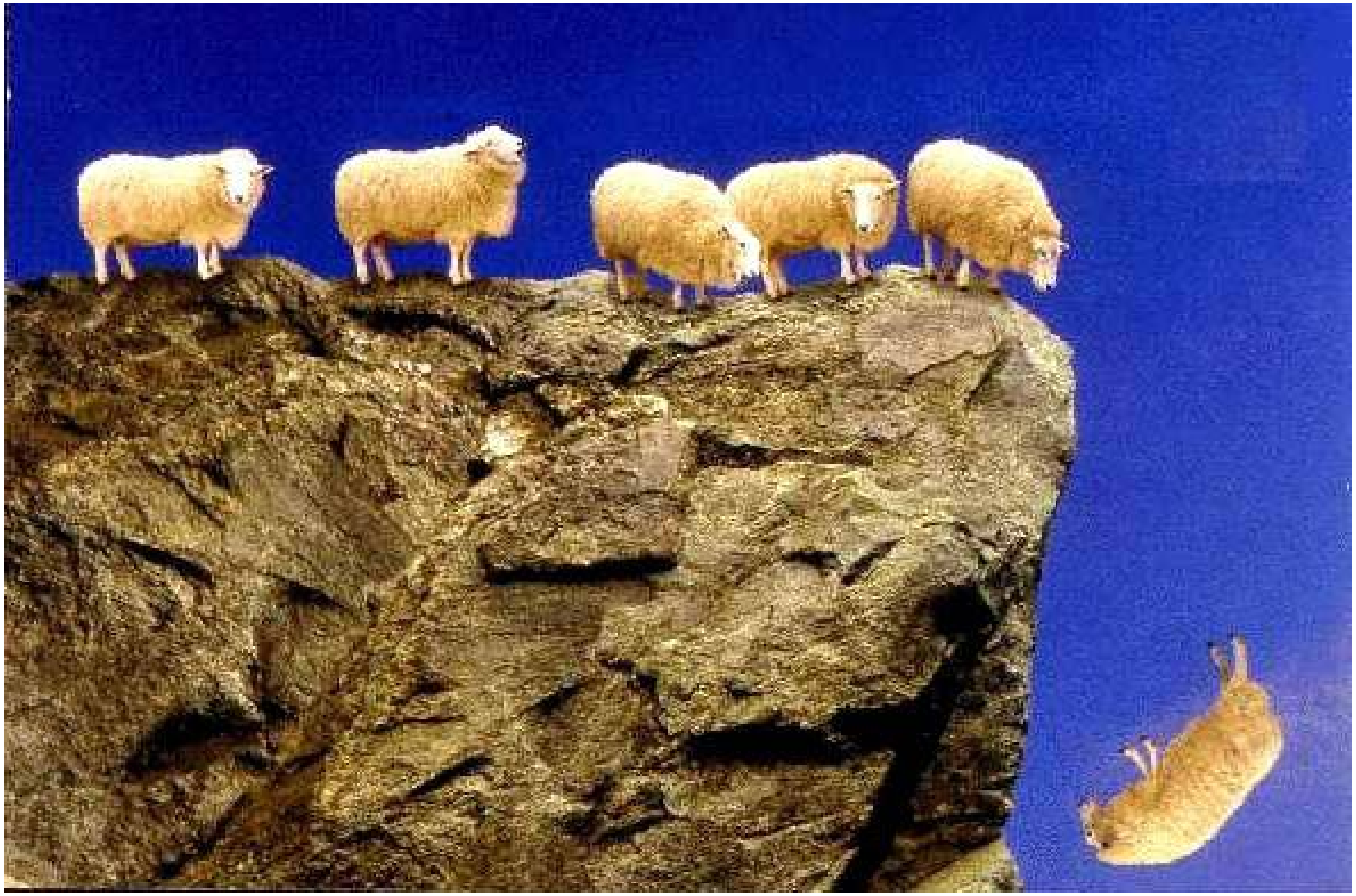
Anti-aging measures: maturity

- All technologies might eventually die-out.
  - Try to avoid one that is dying now.
- Some technologies might not make it at all.
  - The chasm.


Innovators
Early Adopters
Early Majority
Late Majority
Laggards
"The Chasm"
Area under the curve represents number of customers
Technology Adoption Lifecycle

- Safe bet: follow the sheep!

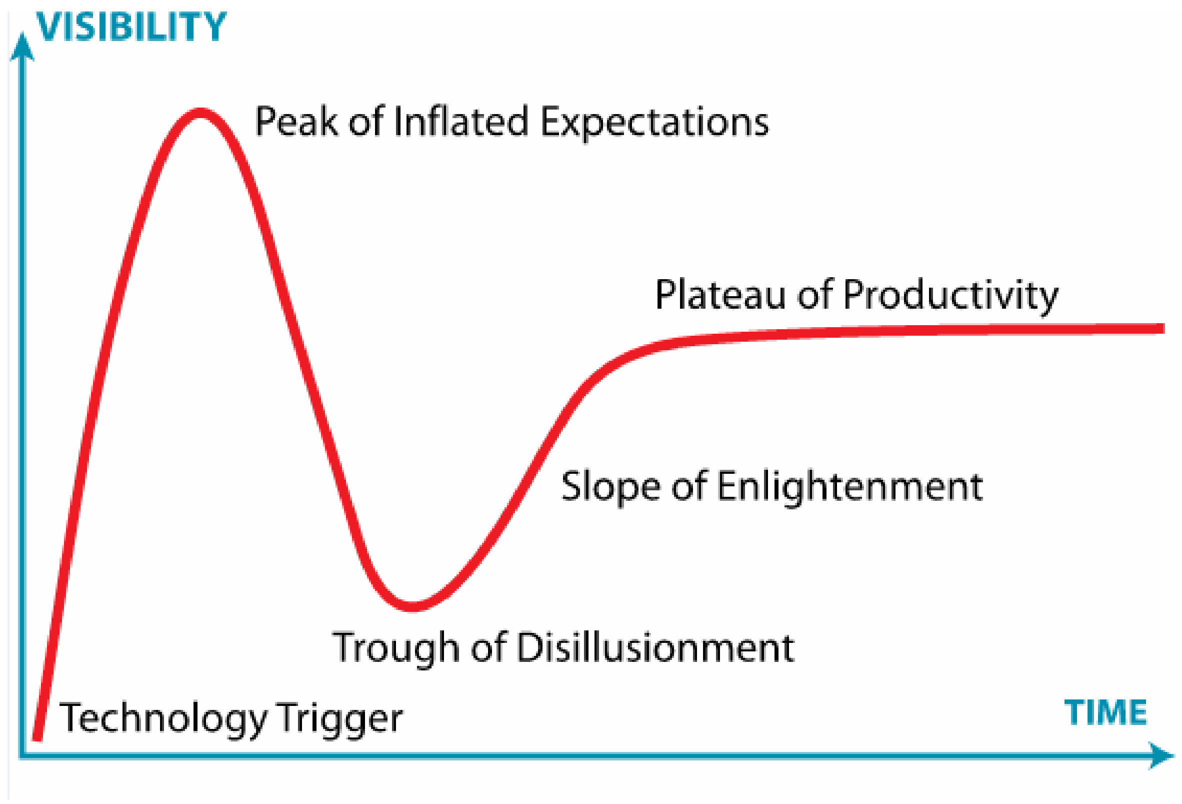
Anti-aging measures: maturity



- Safe bet: follow the sheep!
  (Unless they head off the cliff. ☺)

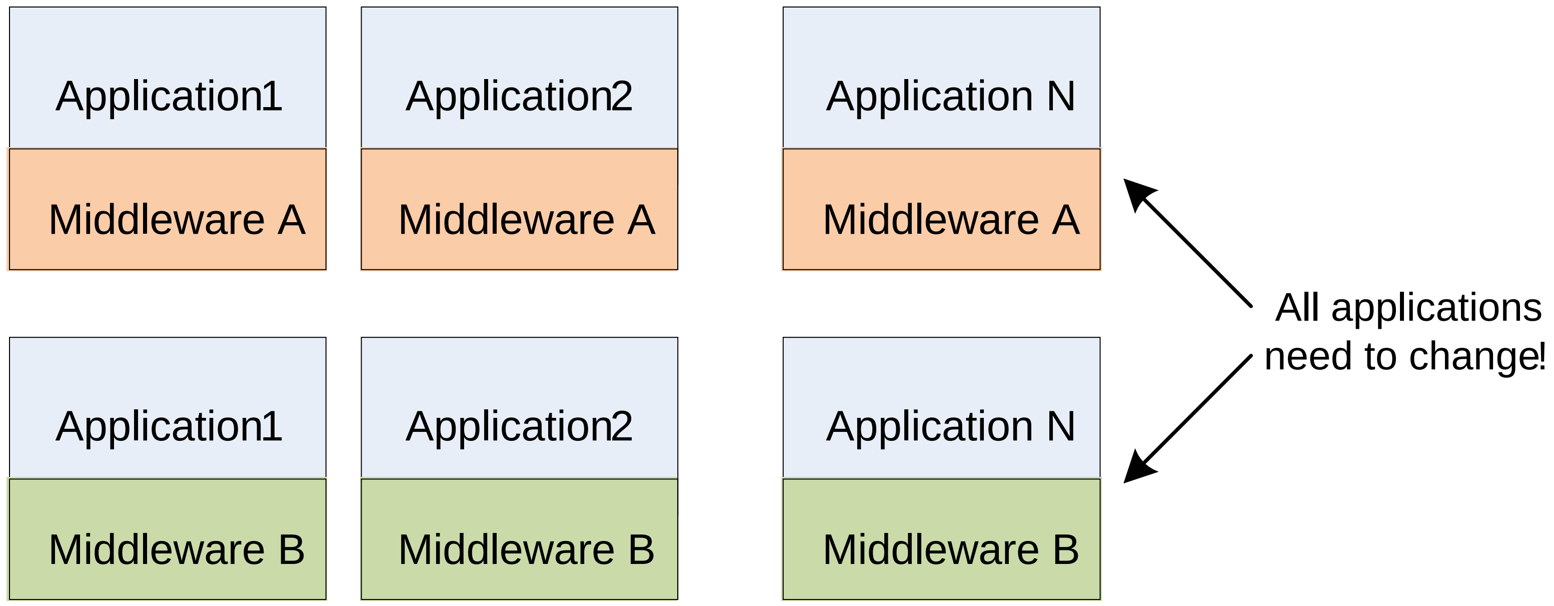
Anti-aging measures: maturity

- Some technologies are over-marketed (the *hype*).
  - Everyone wants to use them “to be cool”.
  - Might be a good choice, but can get over-used/abused (e.g., using XML as a programming language).



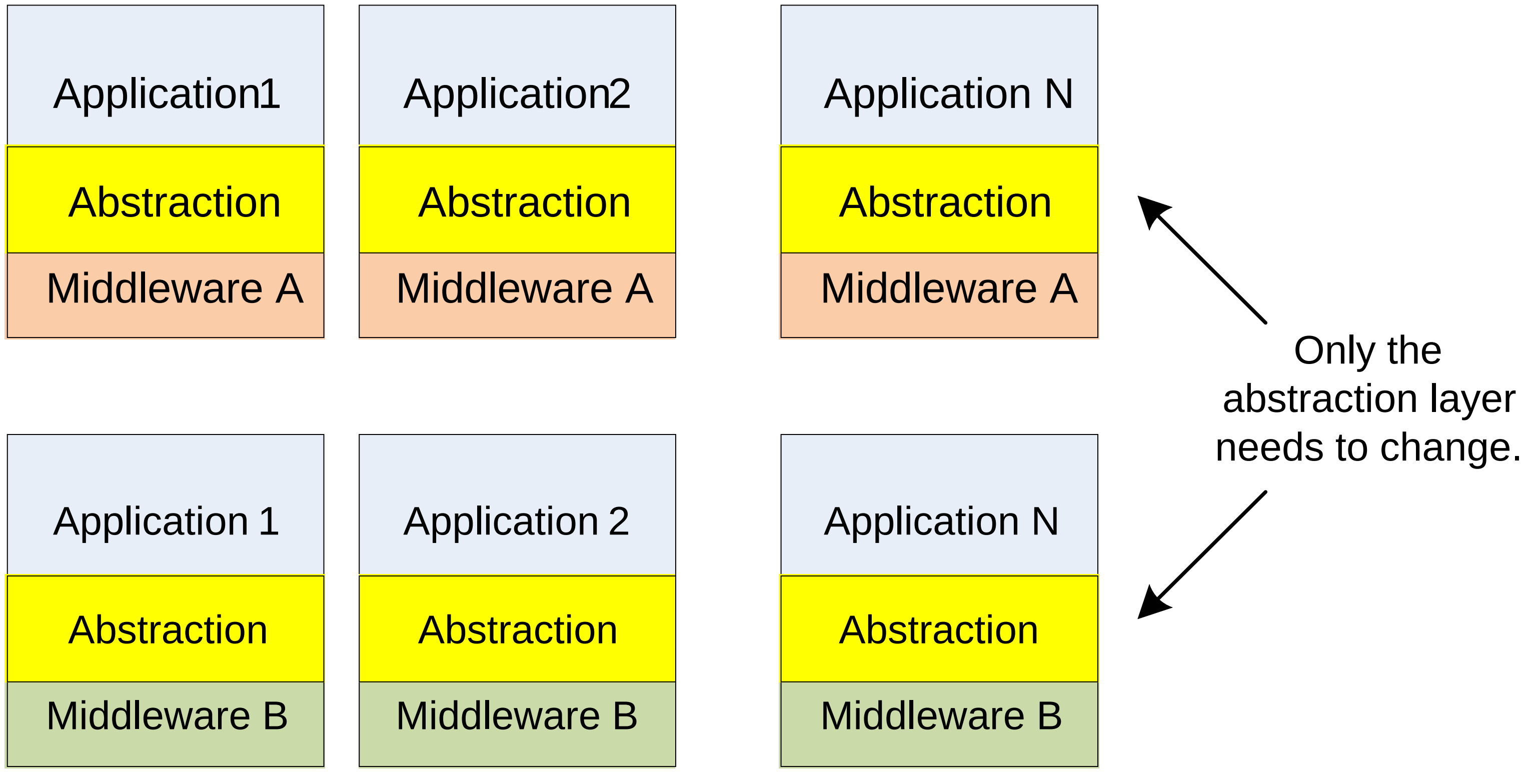
Anti-aging measures: abstraction layer

- If middleware becomes inadequate, it might need to be replaced.
- Most work would be adjusting all applications.



Anti-aging measures: abstraction layer

- Define a thin middleware abstraction layer.
  - A simple API.
  - Applications don’t access middleware directly, only through the abstraction layer.



Middleware in 30 years…

The proverbial crystal ball:

Foreseeable future

- Support for IP multicasting
  - Already supported by some data-centric middleware
  - Scalable monitors
- Hard real-time networking
  - Some work already done (e.g., real-time CORBA), but does not make use of hard real-time network stacks
  - Real-time network stacks not yet standardized (RTnet, NAPI for Linux)
- Convergence of approaches:
  - Some RPC-style middlewares already support message-oriented approach (e.g., CORBA Notification Service), though not optimally
  - RPC-style on top of data-centric approach

Conclusion

- Future is impossible to predict, but we can take some precautions.
- Some criteria for longevity consideration:
    - Proprietary vs. “public domain”
    - Open source
    - Open standard
    - Quality and cost of support
    - Economy of scale
    - Simplicity
    - Maturity

Thank You for Your Attention