Cluster Rolling Upgrade Using Multiple Version Support

Ellard Roush

9 October 2001

Problem Statement

Upgrade the Cluster Operating System Software

While Maintaining Service Availability

Motivation

- **Organizations depend on computer services**
- **All down times should be avoided**
- **Software changes regularly**
- **Rolling Upgrade provides software change w/o down time**
- **Standard Product Feature**
  - **Microsoft Windows 2000**
  - **Compaq TruCluster**
  - **HP MC/ServiceGuard**
  - **Data General UX Cluster**

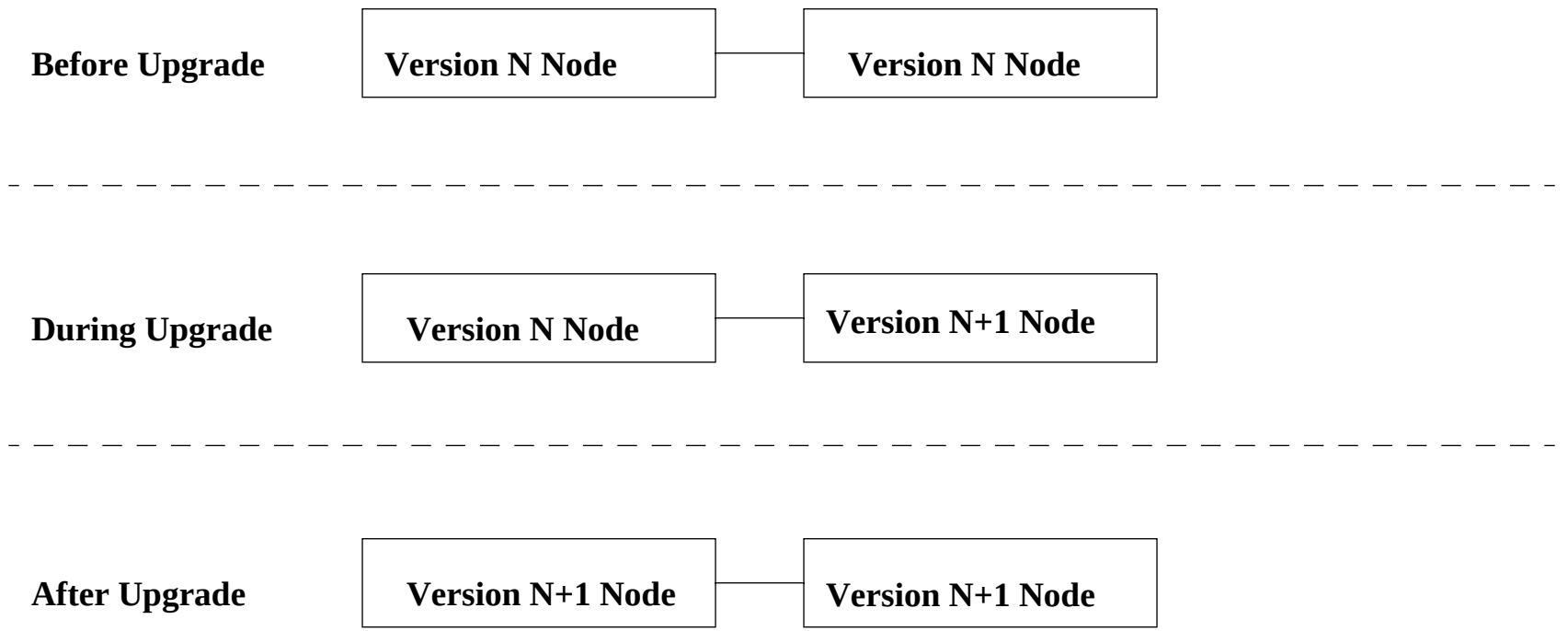
Contributions

- **Object level versioning in the operating system**
- **Wider range of change supported**
  - **"1-1" object replacement**
  - **"Many-1" object consolidation**
  - **"1-Many" object splitting**
- **Versioning is distinct from Inheritance**
- **Minimal overhead**

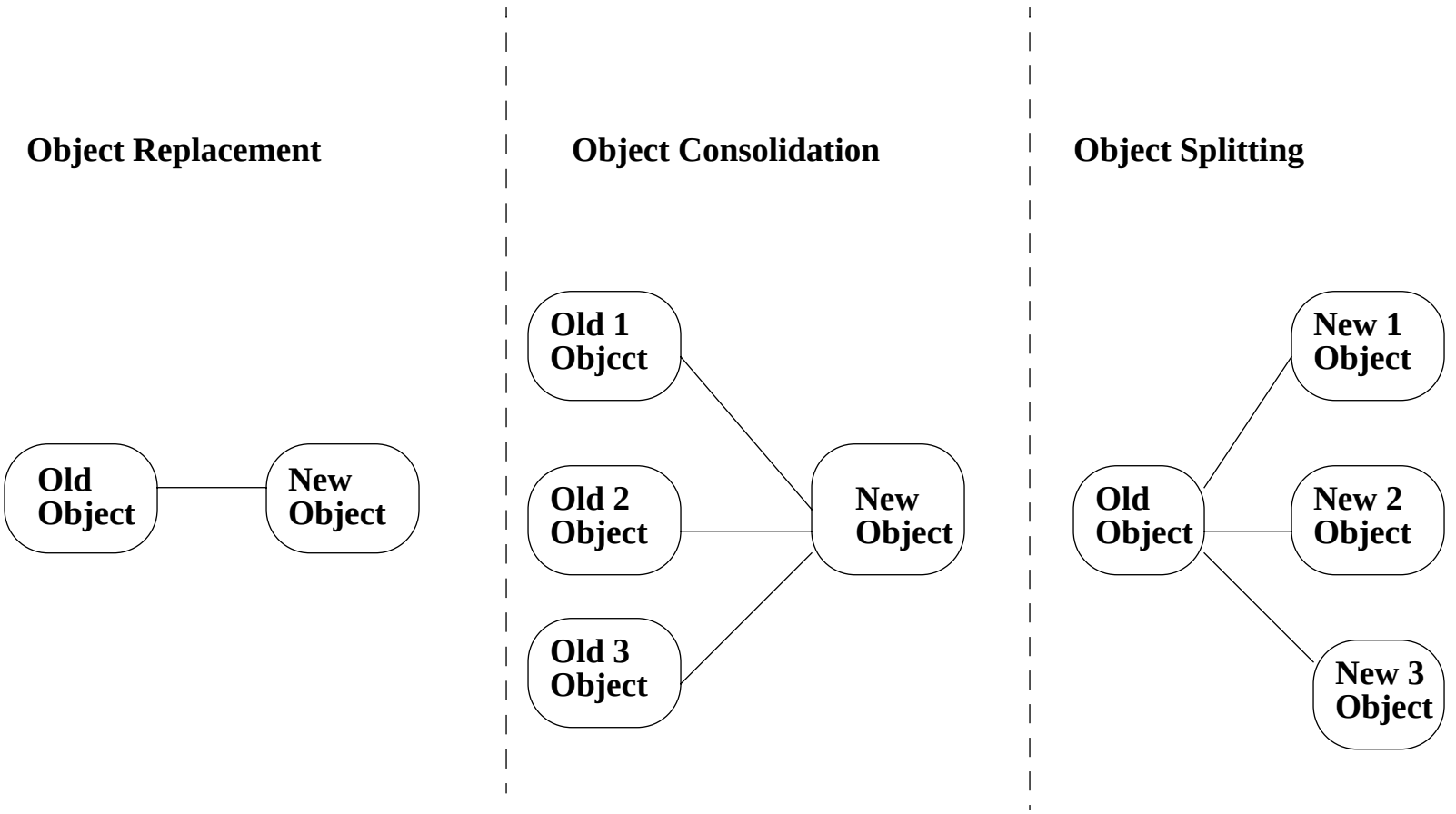
Rolling Upgrade vs Component Upgrade

- **Rolling Upgrade provides simple local O.S. upgrade**
- **Rolling Upgrade limits version concerns to just inter-node interactions**

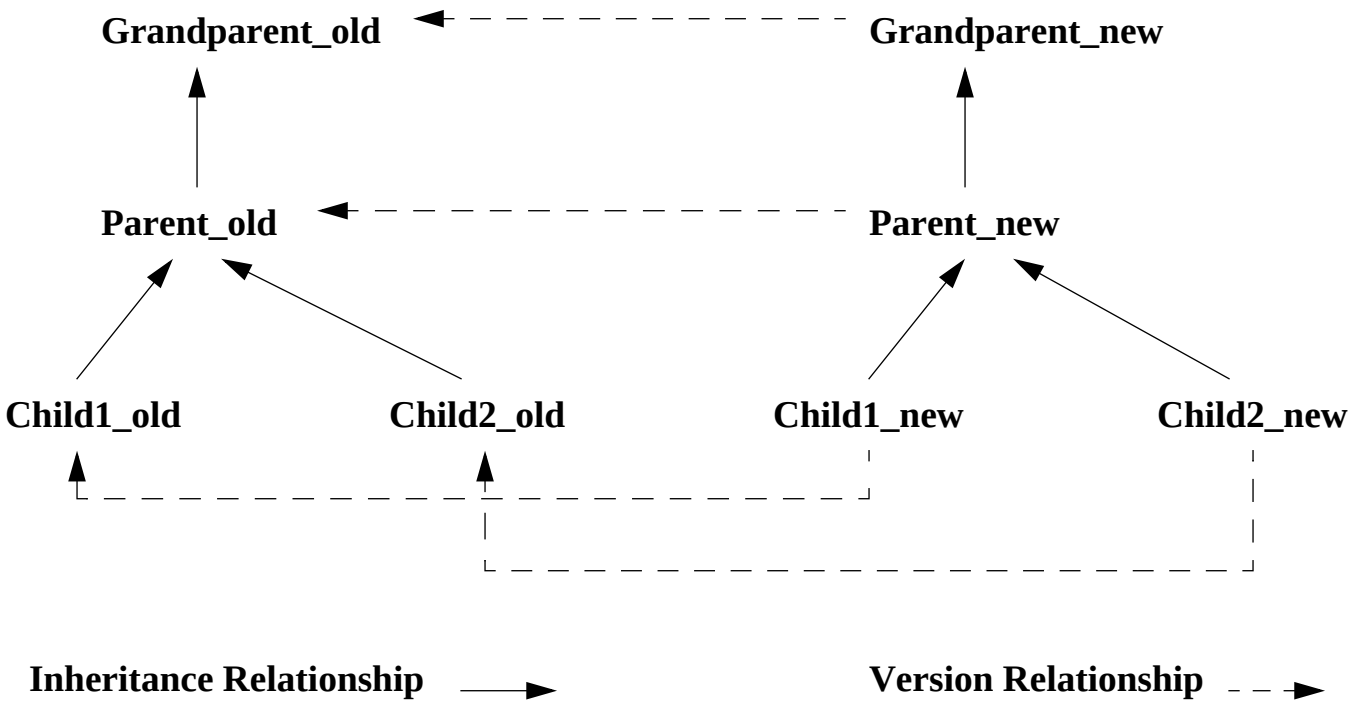
Cluster Rolling Upgrade



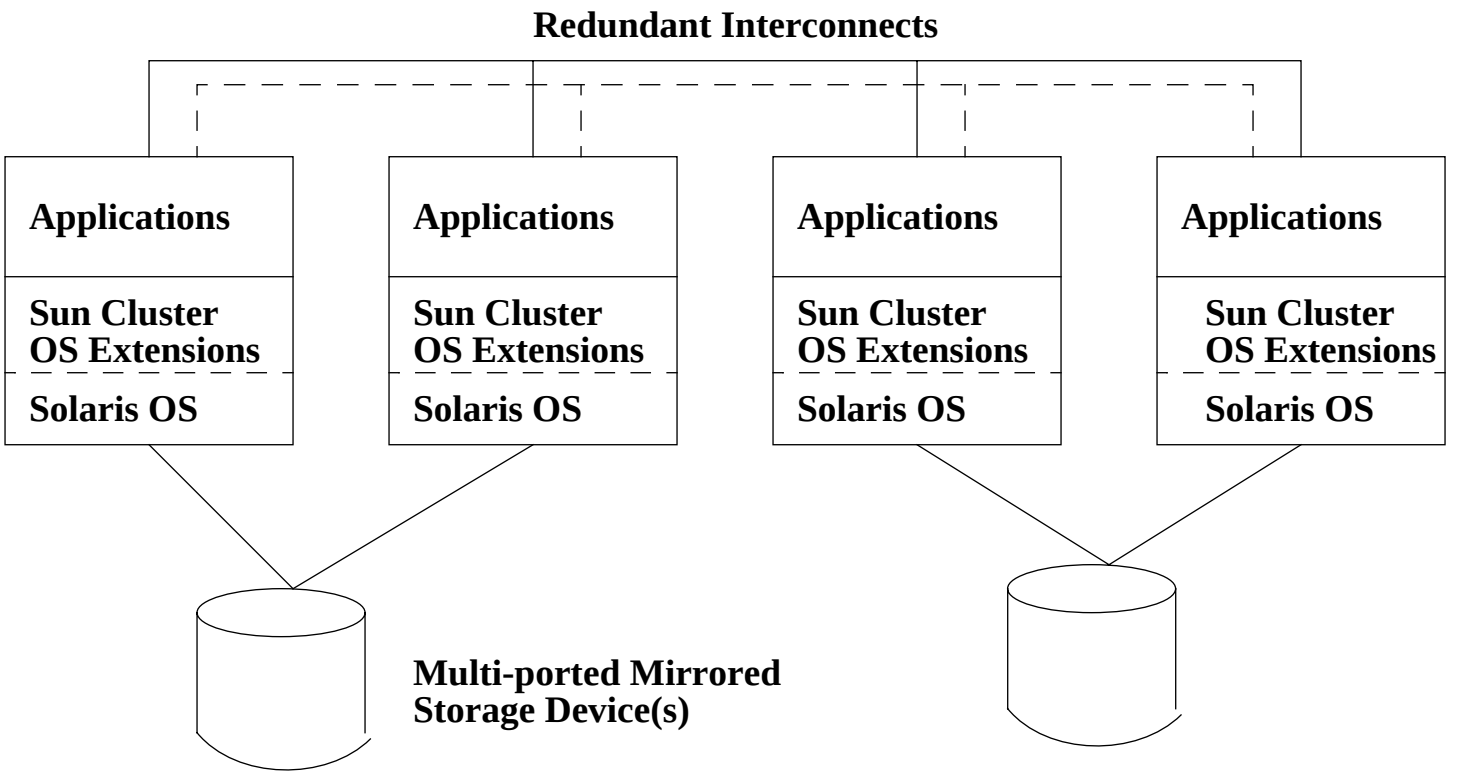
Types of Change



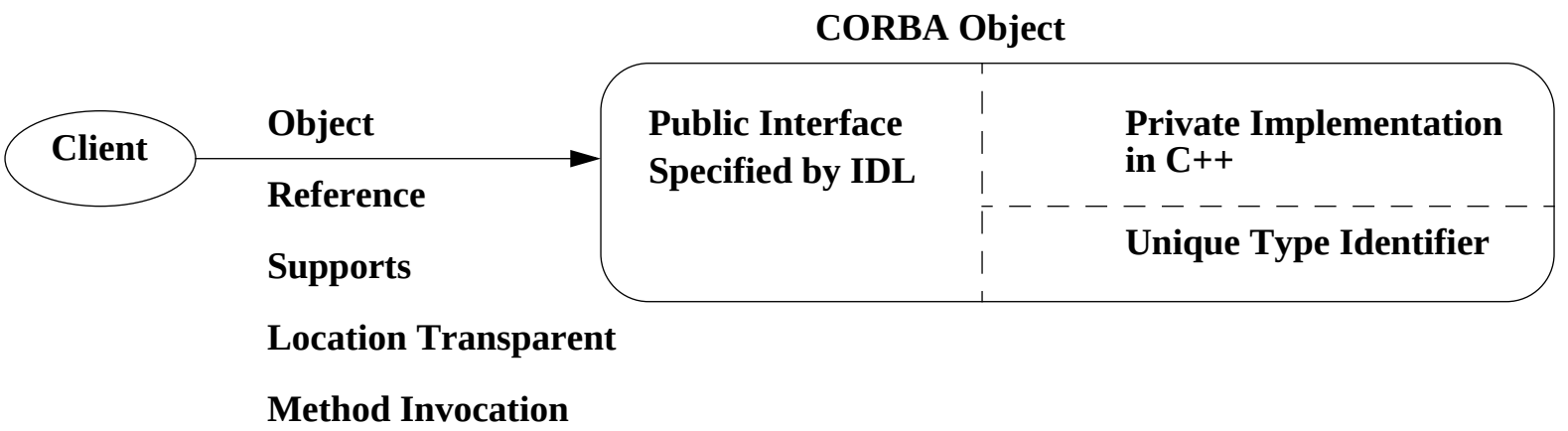
Why Inheritance Doesn’t Work



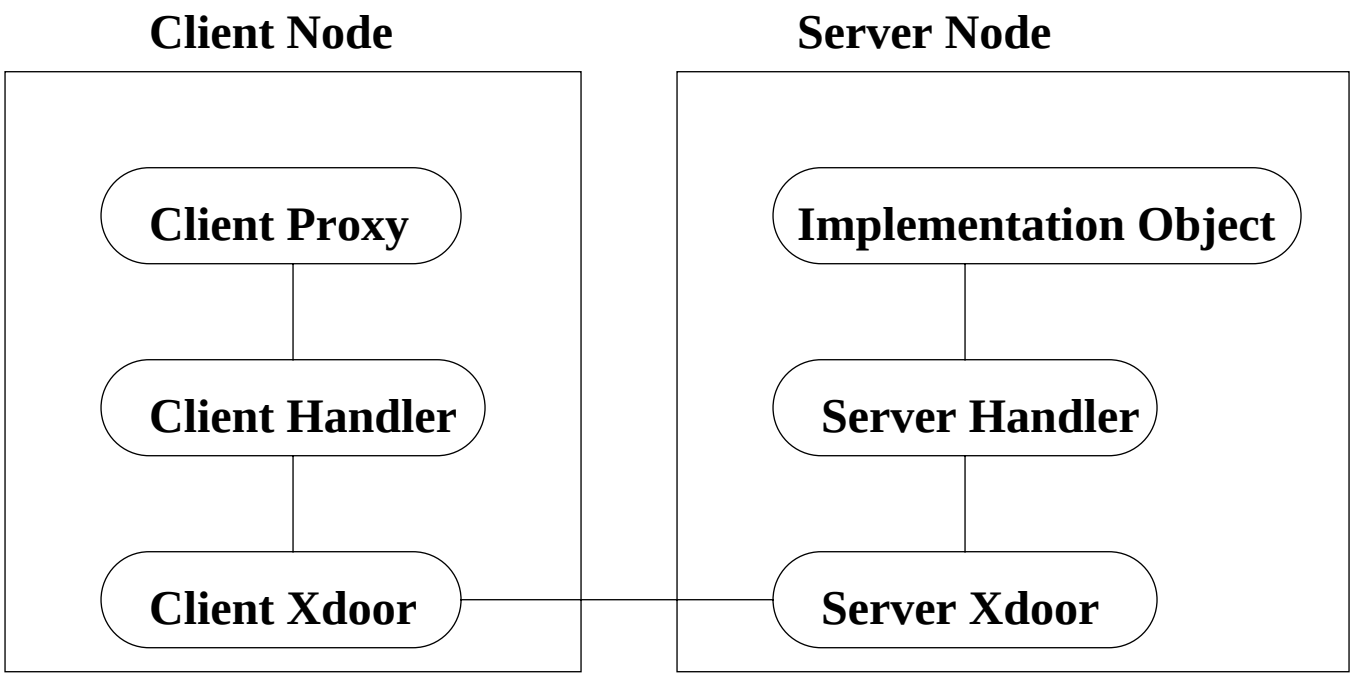
Sun Cluster


Redundant Interconnects
Applications
Sun Cluster
OS Extensions
Solaris OS
Applications
Sun Cluster
OS Extensions
Solaris OS
Applications
Sun Cluster
OS Extensions
Solaris OS
Applications
Sun Cluster
OS Extensions
Solaris OS
Multi-ported Mirrored
Storage Device(s)

Software Model



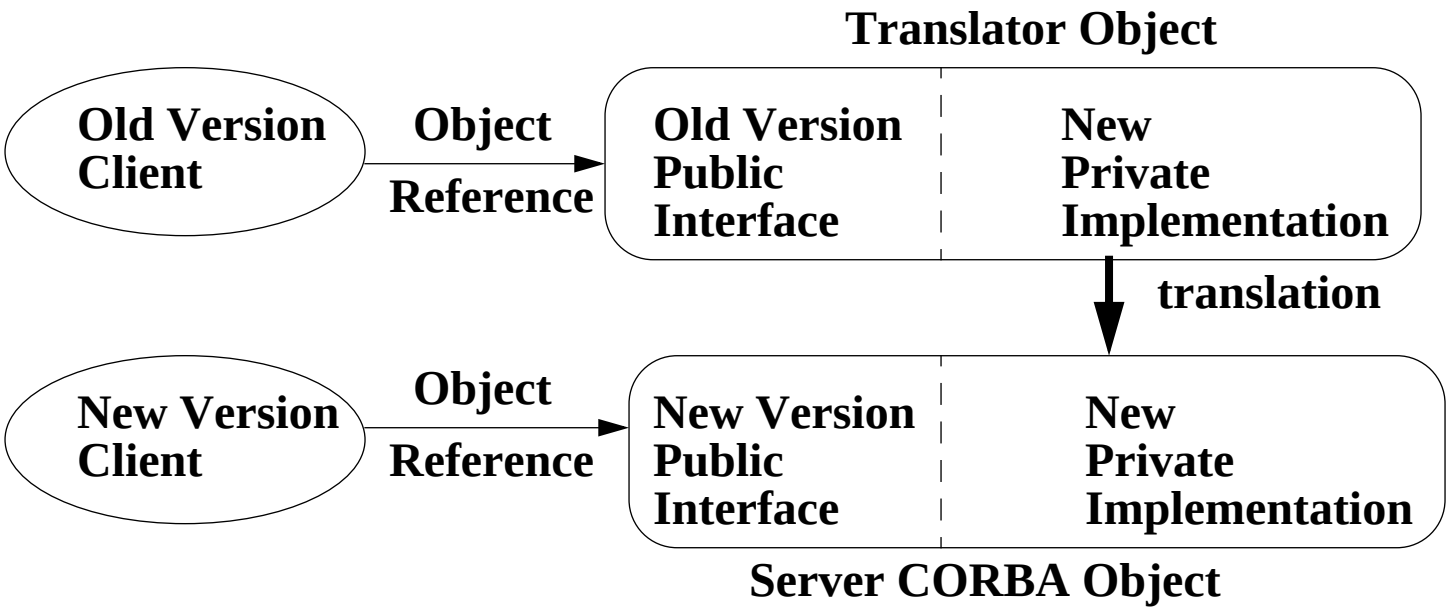
Current Client - Server Infrastructure



Solution Key Features

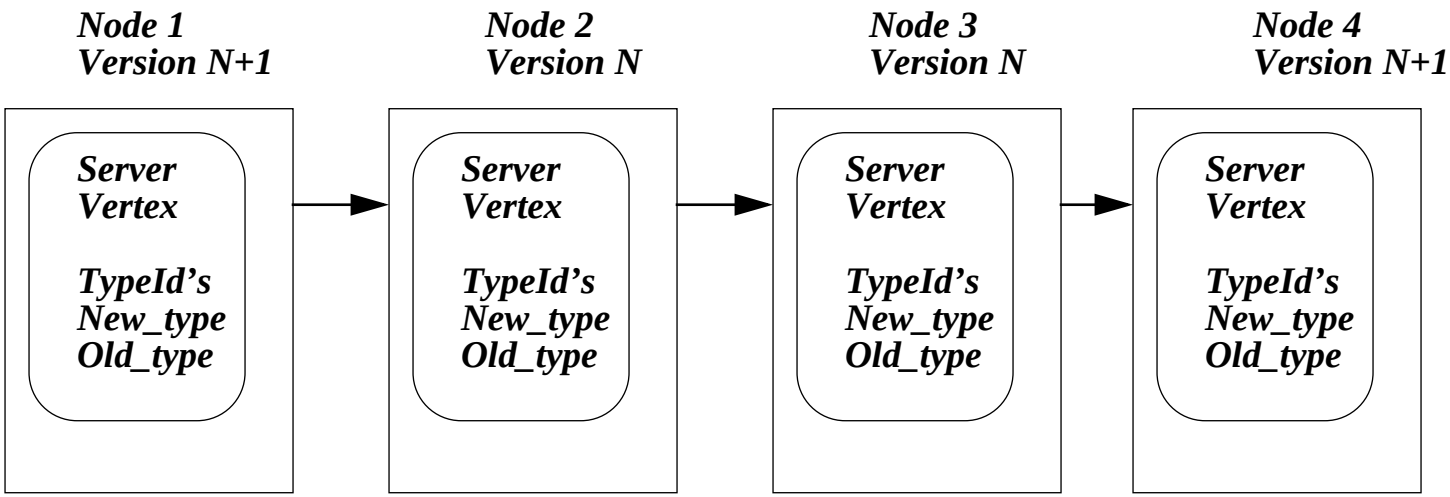
- **IDL specifications**
  - **immutable**
  - **IDL language expanded for version information**
- **Translator**
  - **Fully supports old IDL specification**
  - **Implementation uses new object**
  - **Programmer provides implementation**
- **Vertex**
  - **One object reference can act as old or new object**
  - **Client selects object version**

Solution Model

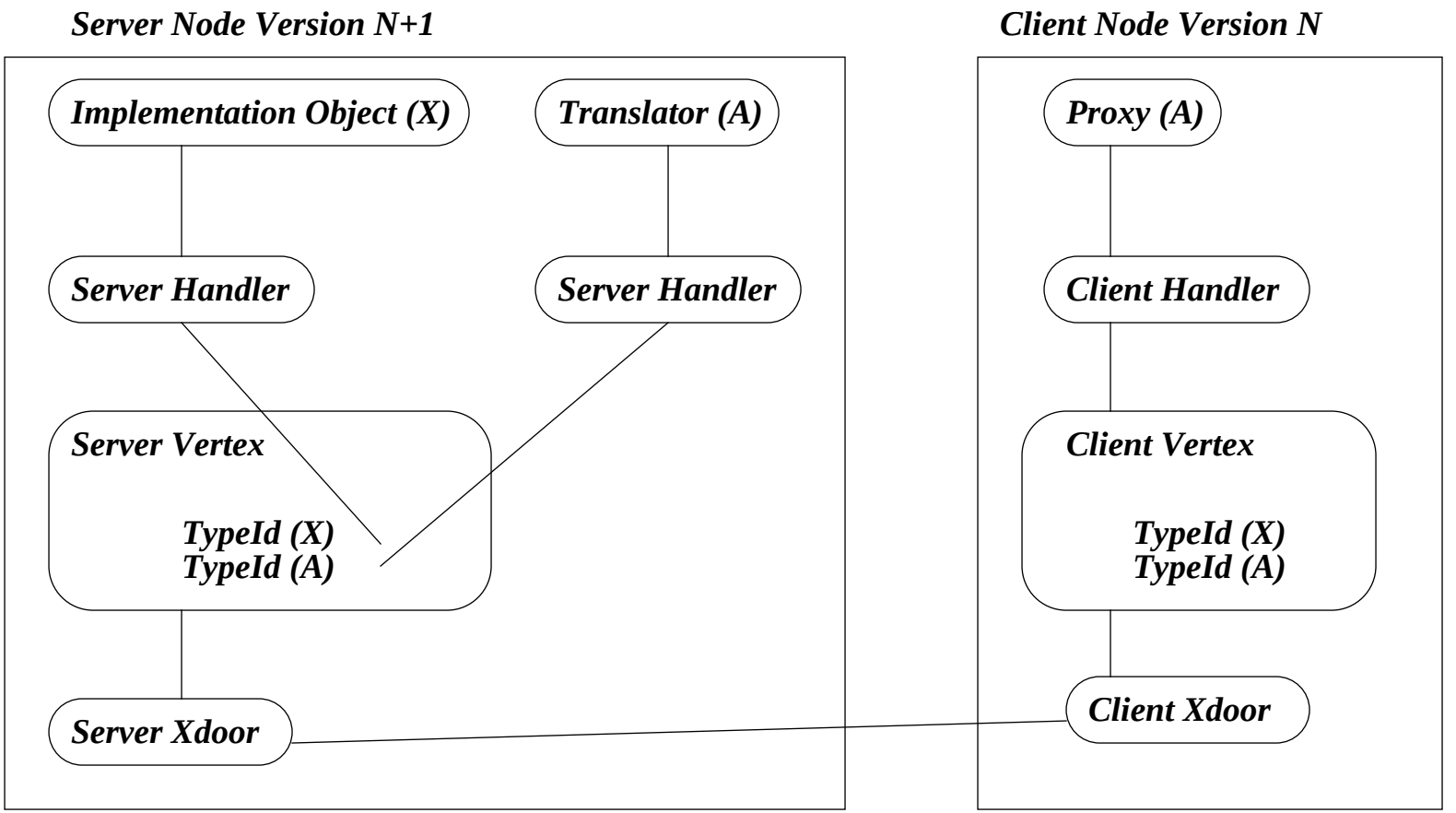


Reference Passing

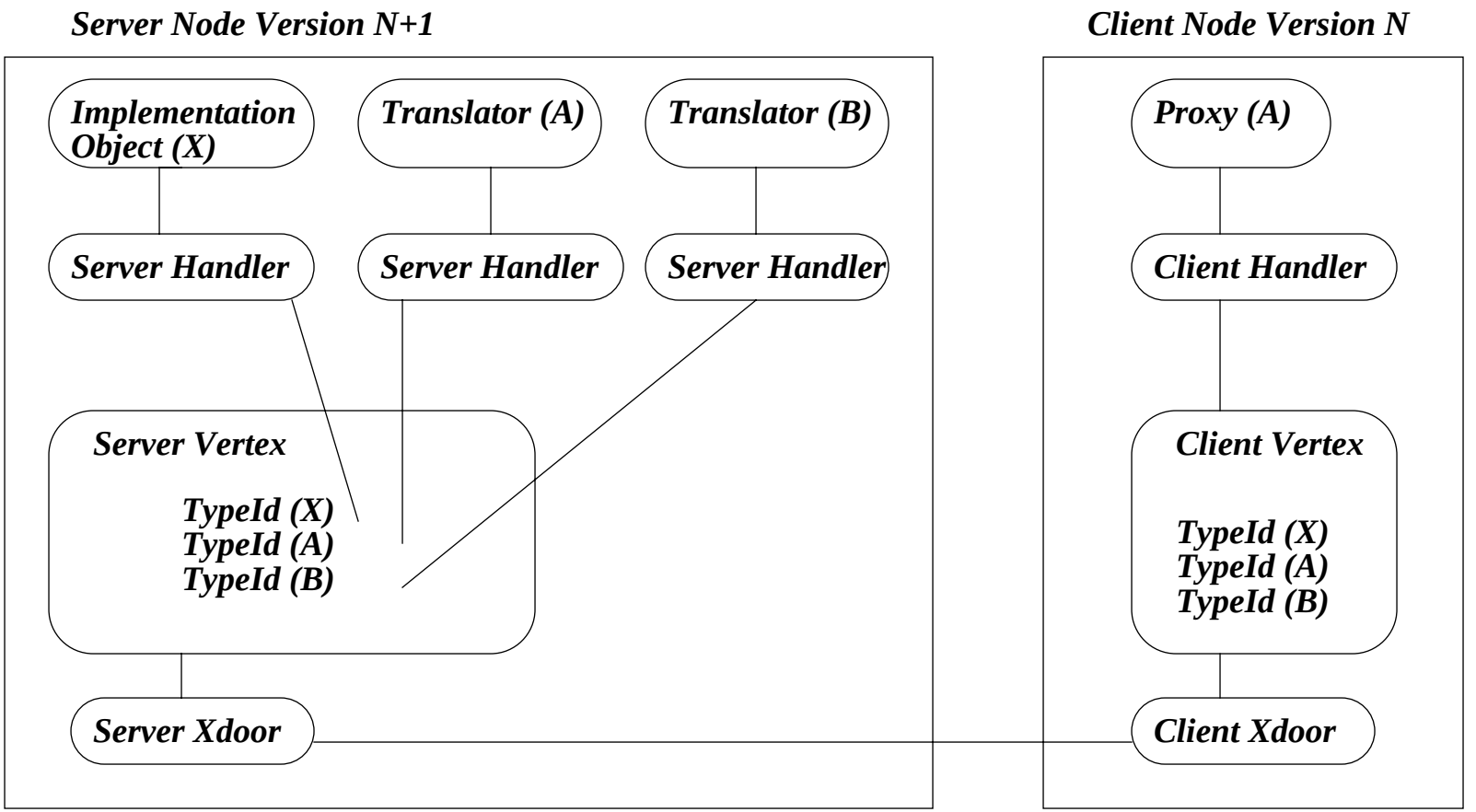
One Object Reference Works Everywhere



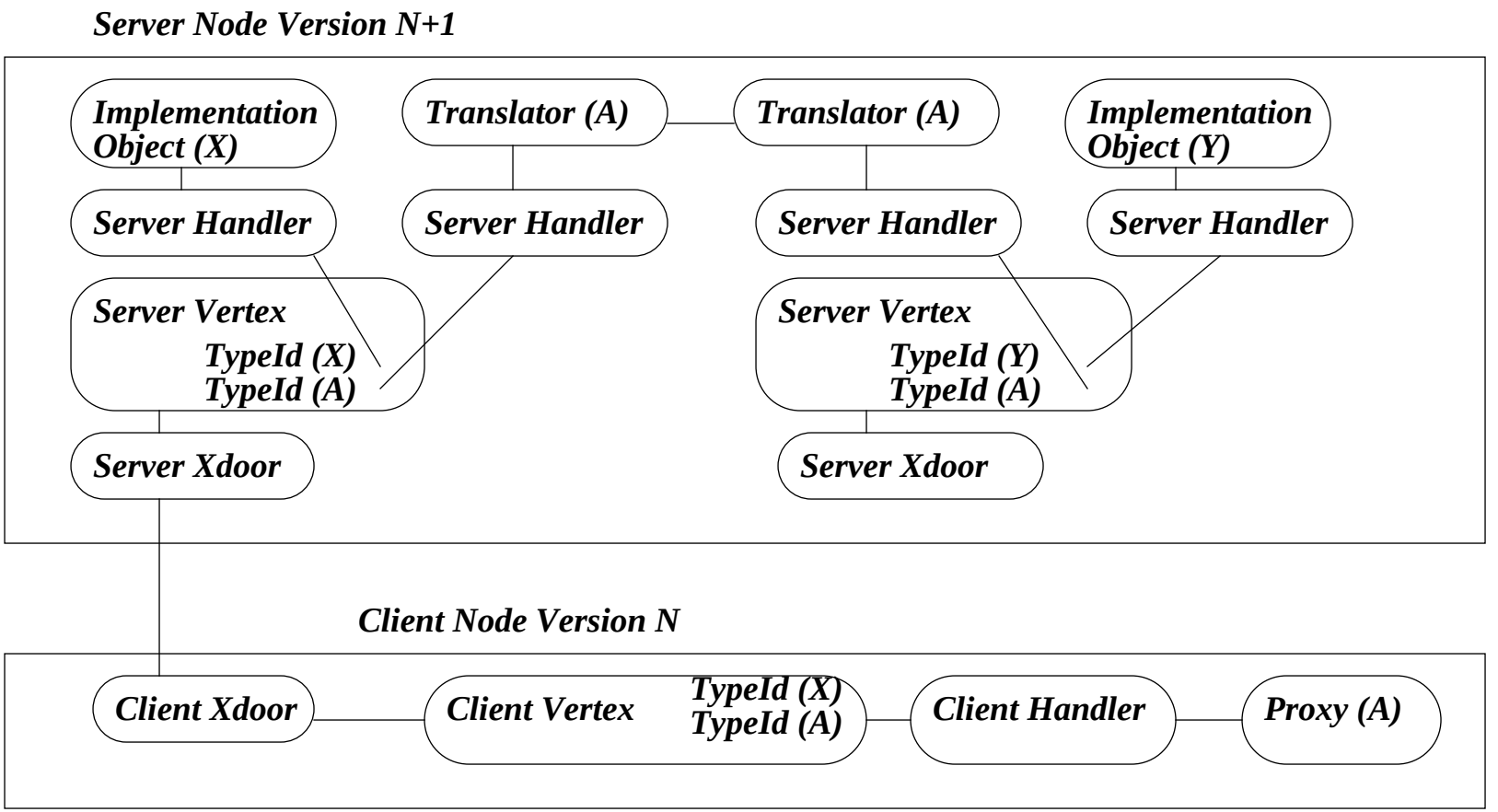
"1-1" Relationship Infrastructure



"Many-1" Relationship Infrastructure



"1-Many" Relationship Infrastructure



Desirable Properties

- **New version node creates only new version objects**
- **Old version node creates only old version objects**
- **Older version never knows anything about newer versions**
- **No time limit on period cluster supports multiple versions**
- **Version change works in either direction**
- **No naming limitations**
- **Supports version retirement**

Conclusions

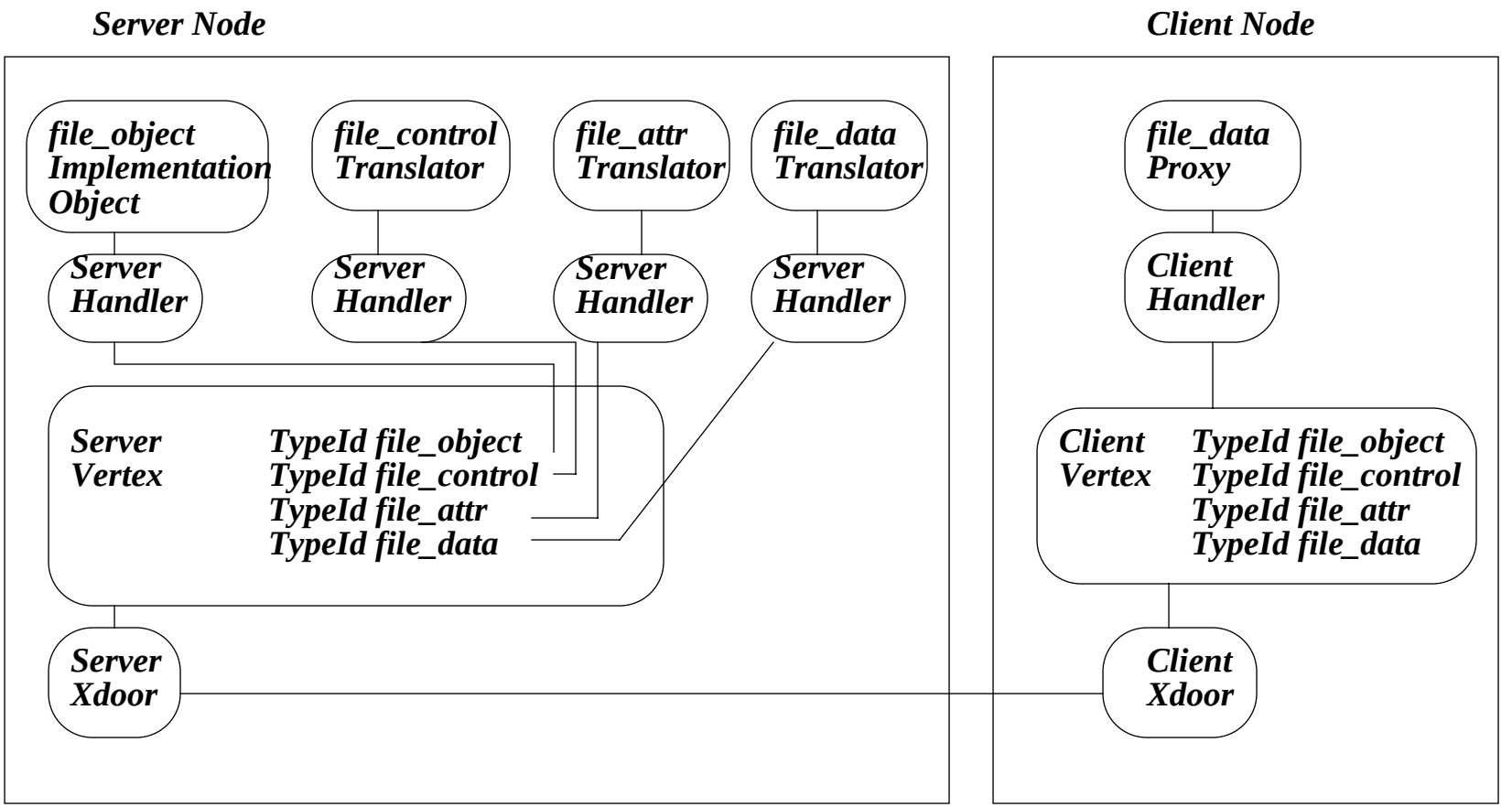
- **Enables different versions to interoperate**
- **Supports Rolling Upgrade**
- **Allows arbitrarily complex relationships**
- **Solves general version interoperability problem**
- **Minimal overhead**

```
//
// Multiple objects supporting one file
//
module file {  // Version N

  interface file_control {
    void    lock();
    void    unlock();
  };

  interface file_attr {
    void    set_accesstime(in time_t time);
    time_t  get_accesstime();
  };

  interface file_data {
    void    put_page(in page_t page, in page_number_t number);
    page_t  get_page(in page_number_t number);
  };
};
```

```
//
// One object supporting one file
//
module file_new {  // Version N+1

  interface file_object
    [ translator            // Old supported interfaces
      file::file_control,
      file::file_attr,
      file::file_data ]
  {
    void    lock();
    void    unlock();

    void    set_accesstime(in time_t time);
    time_t  get_accesstime();

    void    put_page(in page_t page, in page_number_t number);
    page_t  get_page(in page_number_t number);
  };
};
```

```
//
// "Translator" Implementation
//
class file_control_impl : server_object_template<file::file_control>
{
public:

  file_control_impl(file_object_impl *filep) :
    file_objectp(filep)
      {}

  void    lock()     { file_objectp->lock();   }
  void    unlock()   { file_objectp->unlock(); }

private:

   //
   // Pointer to new implementation object
   //
  file_object_impl  *file_objectp;
};
```

Example Infrastructure



Reference Passing

One object reference works everywhere