Federal University of Santa Catarina
Laboratory for Software/Hardware Integration

Resource Management in Deeply Embedded Systems using Static Metaprogramming Techniques

Roger K. Immich
Diego L. Kreutz
Antônio Augusto Fröhlich *

http://www.lisha.ufsc.br/

August 2006

Motivation

- Resource management mechanisms in ordinary operating systems are usually too “expensive” to be adopted in Deeply Embedded Systems (DES)
  - Complex data structures
  - Polymorphic objects
- Yet embedded systems do manipulate resources that must be managed!

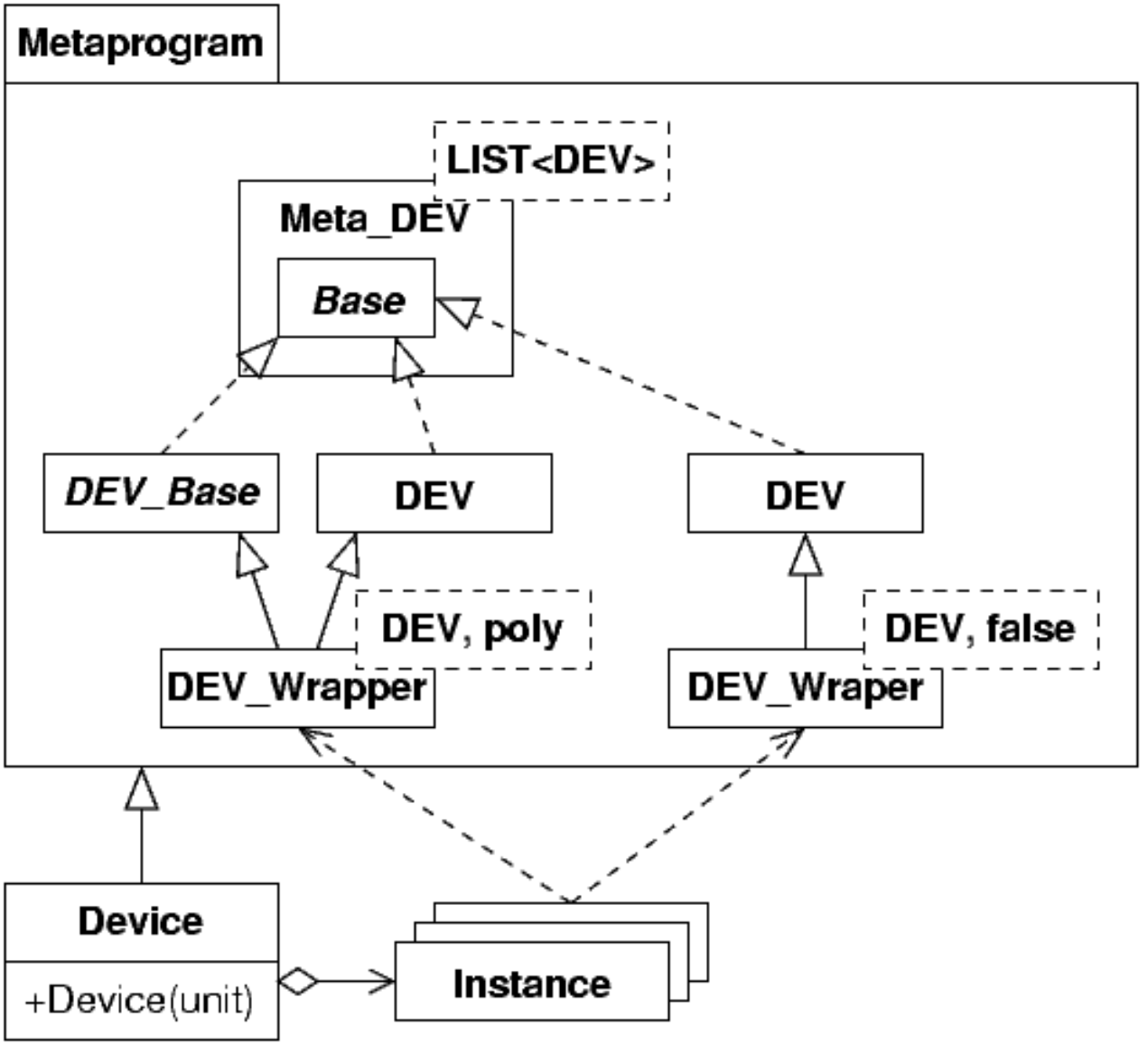
Resource Management in Deeply Embedded Systems

- Basic resources such as CPU time and memory are usually managed by the OS
- The management of platform-specific resources such as timers, ADCs, network interfaces, sensors and actuators is usually consider part of application programmer duties!
  - Application quality (portability, reusability, maintainability, etc) is compromised!

An Static Metaprogrammed Resource Manager

- DES resources rarely change at run-time
- Static metaprogramming (SMP) techniques
  - Flexibility at compile-time
  - No run-time overhead
- SMP resource manager
  - Absent if associated resource is absent
  - Direct access to resources that are unique
  - Direct access to resources of the same type
  - Indirect access to resources of different types (polymorphic)

SMP Resource Manager Overview



Metaprogram Basic Elements

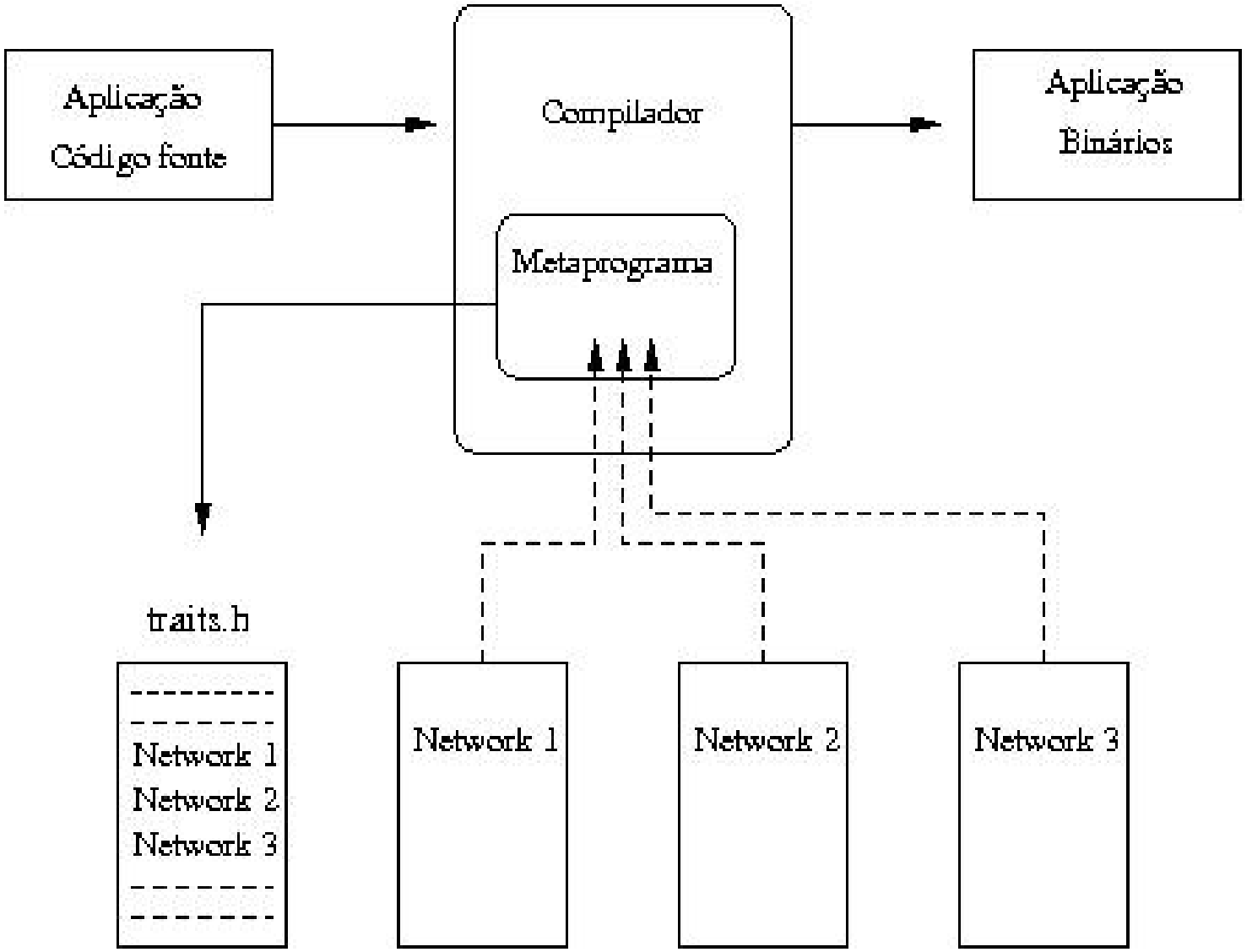
- Conditional: IF-THEN-ELSE type
- Comparision: type EQUALty check
- Container: linked LIST of types
  - Is EMPTY
  - Is POLYMORPHIC
  - COUNT
  - HEAD
  - TAIL
  - GET<>

Example: Network Gateway

```
...
Network net0(0);
Network net1(1);
...
net0.receive(&buf, &len);
...
net1.send(buf,len);
...
```

- The networks can be of same type or not
  - Only if they are different, `Network` is polymorphic
  - Transparent for user

Compilation Process



Traits

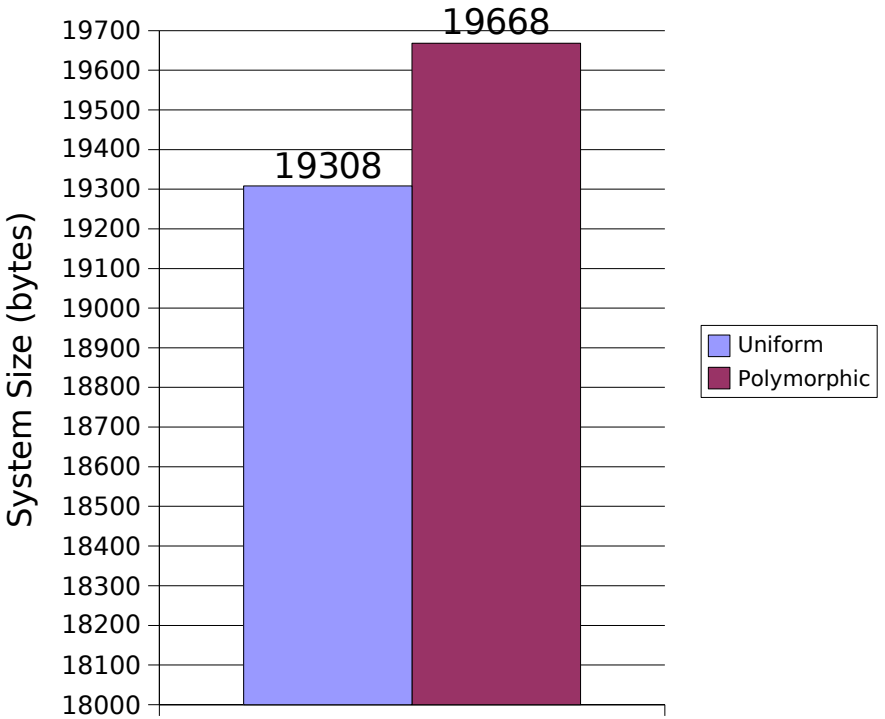
```
template <> struct Traits<PC_NIC>: public
Traits<PC_Common>
{
    typedef LIST<PCNet32, PCNet32> NICS;

    static const int PCNET32_UNITS =
        NICS::Count<PCNet32>::Result;
    static const int PCNET32_SEND_BUFFERS = 8;
    static const int PCNET32_RECV_BUFFERS = 8;

    static const int E100_UNITS =
        NICS::Count<E100>::Result;
    static const int E100_SEND_BUFFERS = 8;
    static const int E100_RECV_BUFFERS = 8;
};
```

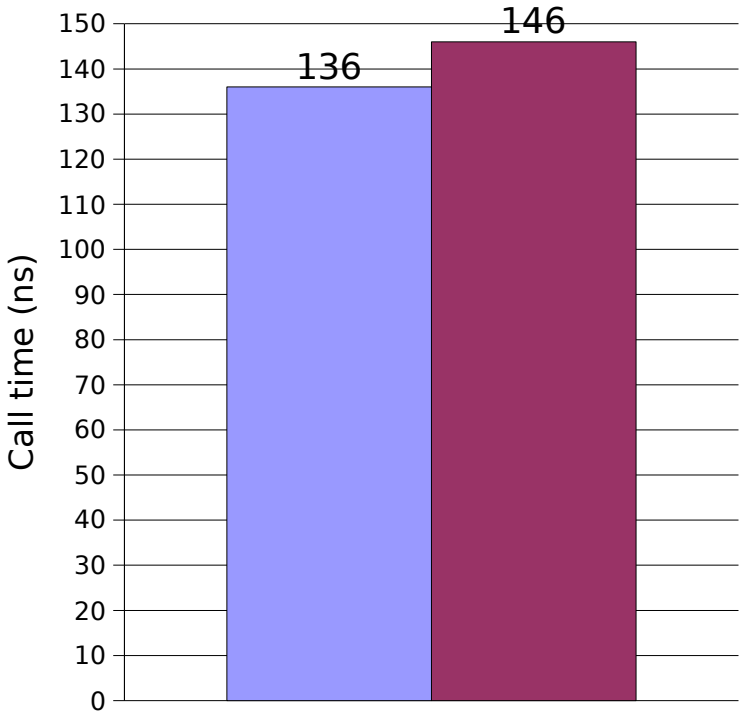
Polymorphism Handling

```
template<typename NICS>
class Meta_NIC
{
   //...
public:
    typedef typename IF<
        typename NICS::Polymorphic,
        NIC_Base,
        typename NICS::template Get<0>::Result
    >::Result Base;
  // ...
};
```

EPOS Implementation

- EPOS with multithreading and dynamic memory management on IA-32
  - 2 x PCNet32
  - PCNet32 + E100





Conclusion

- Deeply Embedded Systems deserve better Run-Time Support
- Ordinary implementation techniques are much too heavy for DES
- DES requirements are usually known beforehand
- Static Metaprogramming can bring traditional OS interfaces to DES at low overhead