Introduction to MINA

A **M**ultipurpose
Infrastructure
for **N**etwork
Applications

April 2005, Trustin Lee, ASF

Contents

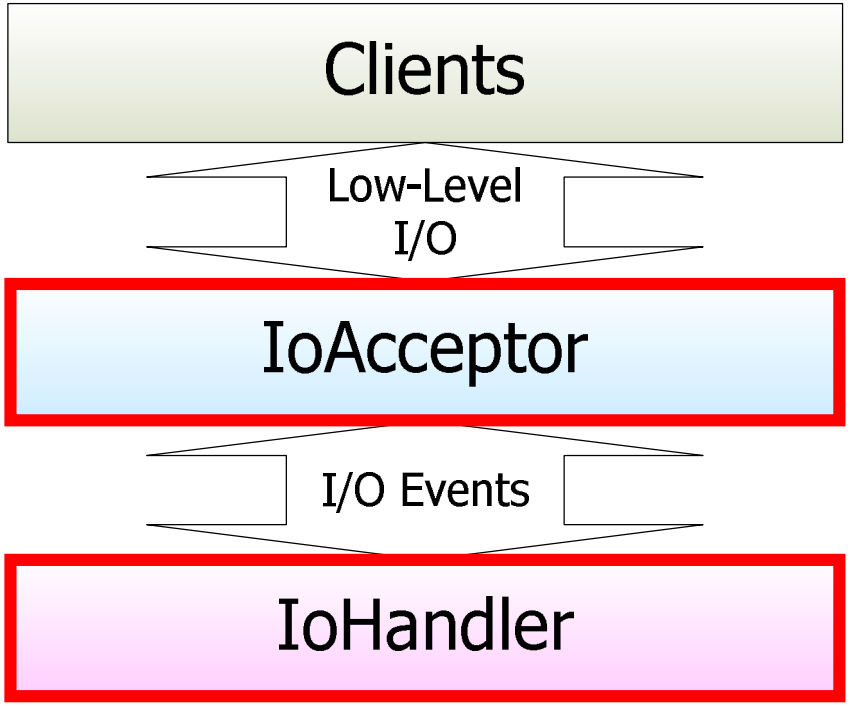
- Overview
- How to Program
- Filter Mechanism
- Proof of Productivity
- Architecture Review
- Conclusion

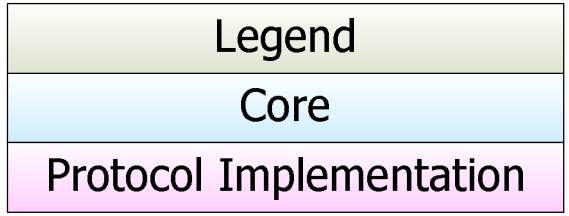
Overview

What is MINA?

- A network application framework
  - Feature-rich
  - Extensible
  - Designed for agile client/server programming
    - Client- or server-less unit testing
    - Very high reusability and maintainability
  - Yet scalable / high performance

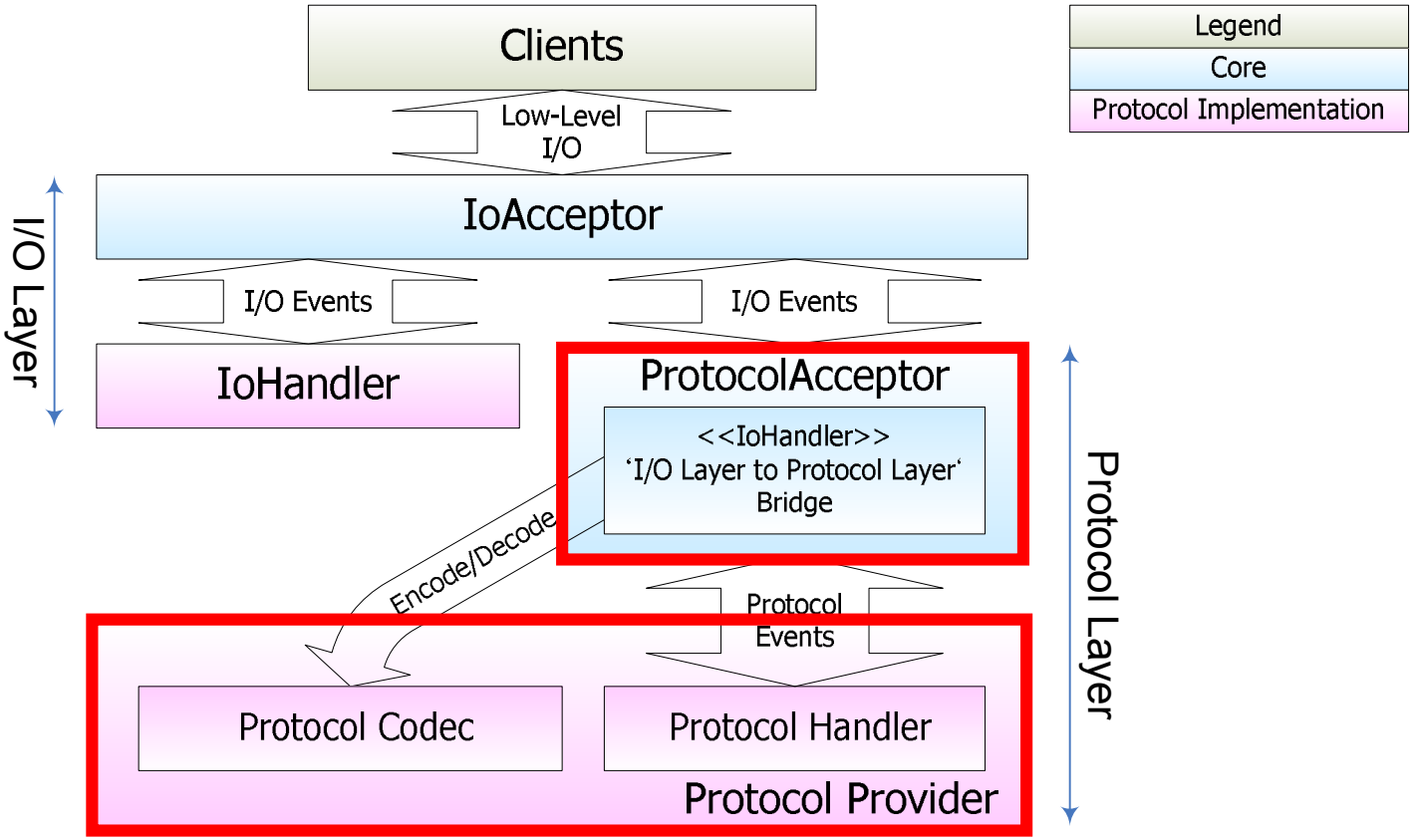
Architecture: I/O Layer





- MINA abstracts all low-level I/O via abstract API.
- **`IoHandlers`** get notified when I/O events occur.
- You communicate by reading and writing data buffers.

Architecture: Protocol Layer



- Built upon I/O layer
- Good when you implement complex protocols
- You communicate by sending and receiving message objects..

MINA Abstract API

- Single API for various transport types
- Highly extensible
- Unit-test your server using mock objects.
  - ⇒ no real clients anymore!

MINA Abstract API (Cont'd)

- Once a protocol implemented, it works for:
  - NIO sockets
    - TCP/IP
    - UDP/IP
  - In-VM pipe
  - Coming soon:
    - Non-NIO sockets
    - Serial port
    - Parallel port
    - Multicast (when Mustang is ready)

How to Program

What MINA Does For You

- You NEVER need to program...
  - Stream I/O
  - NIO
  - Thread management
  - Buffer management

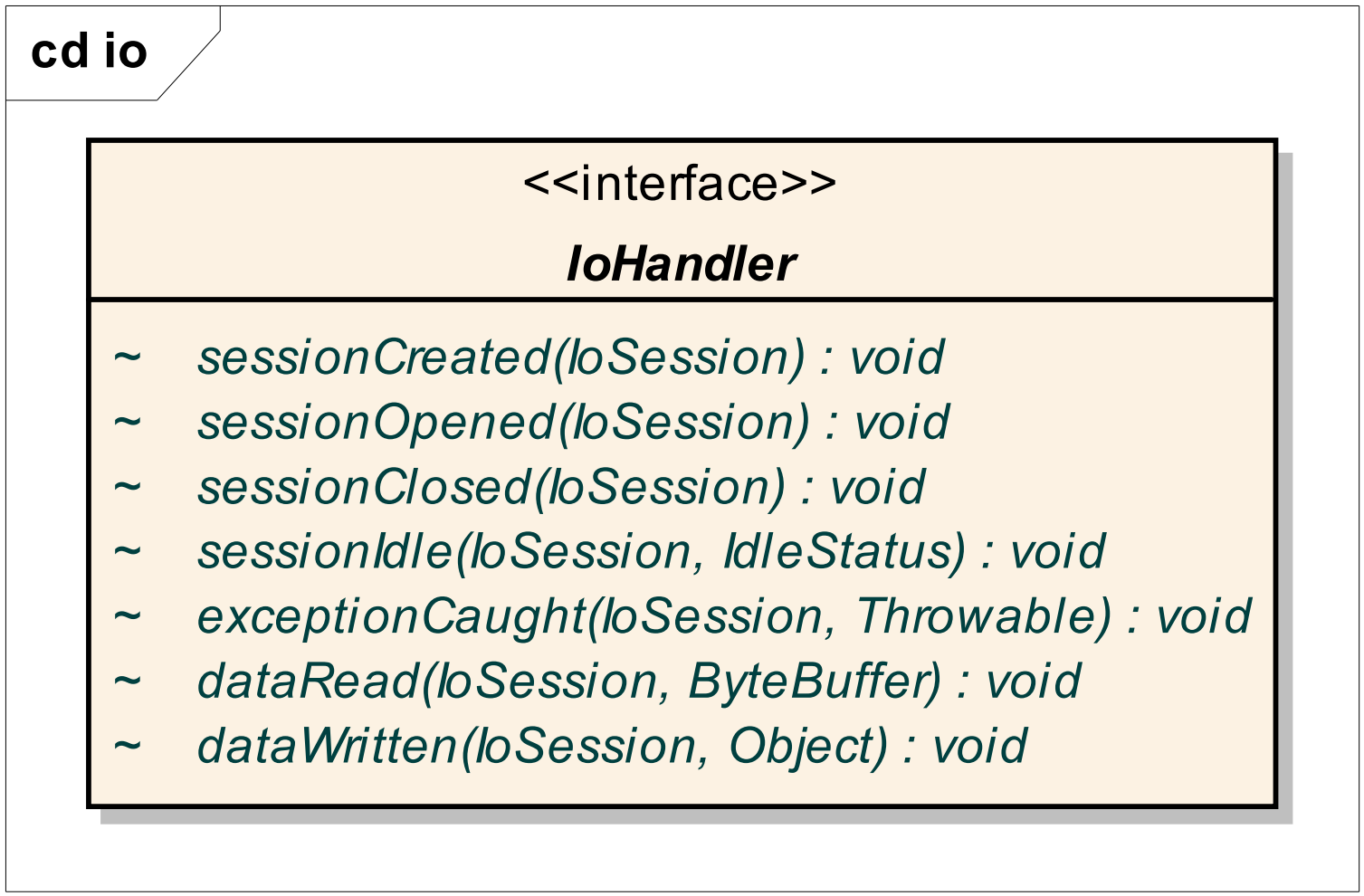
because it does ALL of them for you!

Then what do you have to do?

What You Should Do

- The first way to implement your protocol:
  - Using **I/O Layer**: **`IoHandler`**
    - You communicate by reading and writing data buffers.

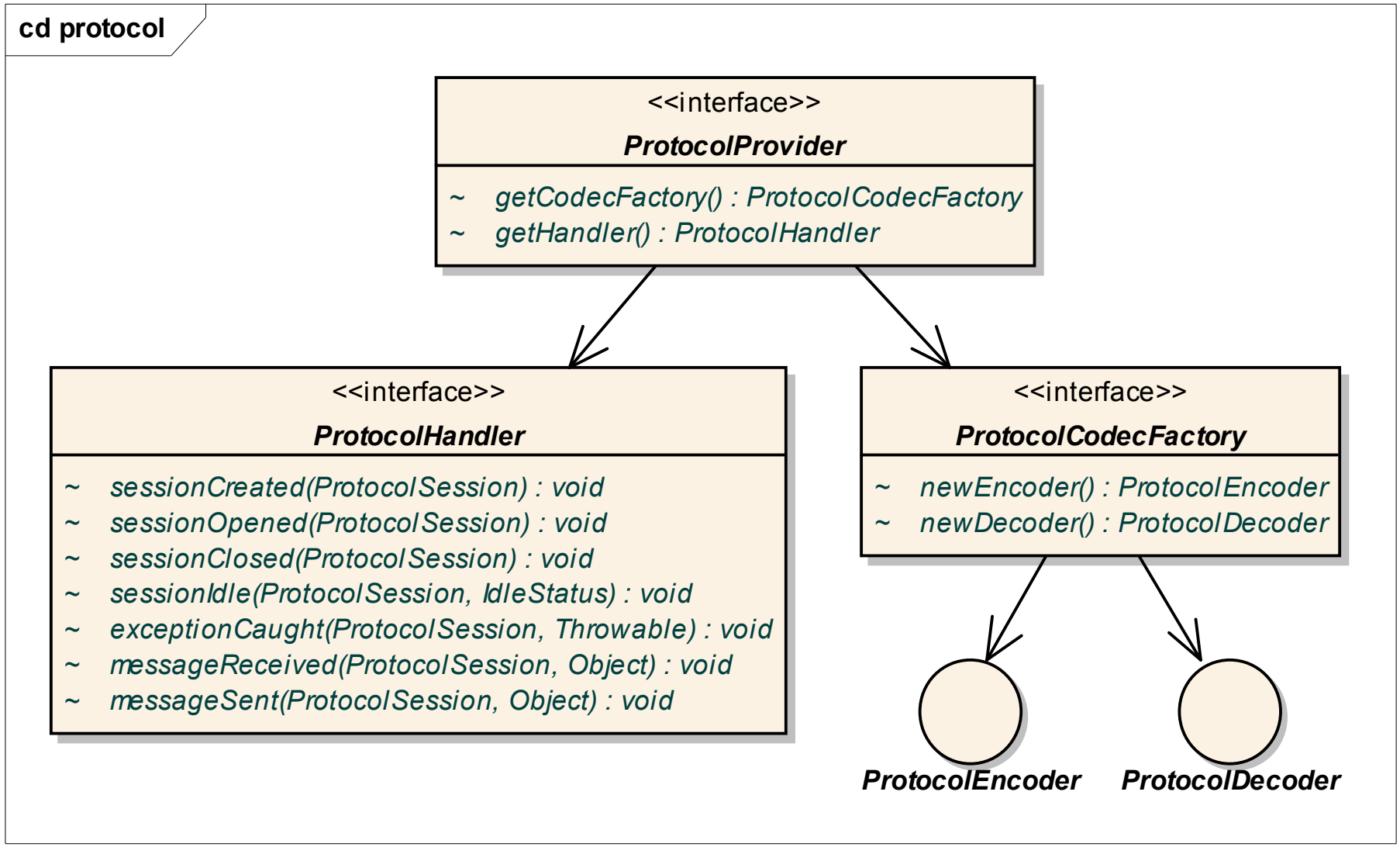
What You Should Do (Cont'd)



What You Should Do (Cont'd)

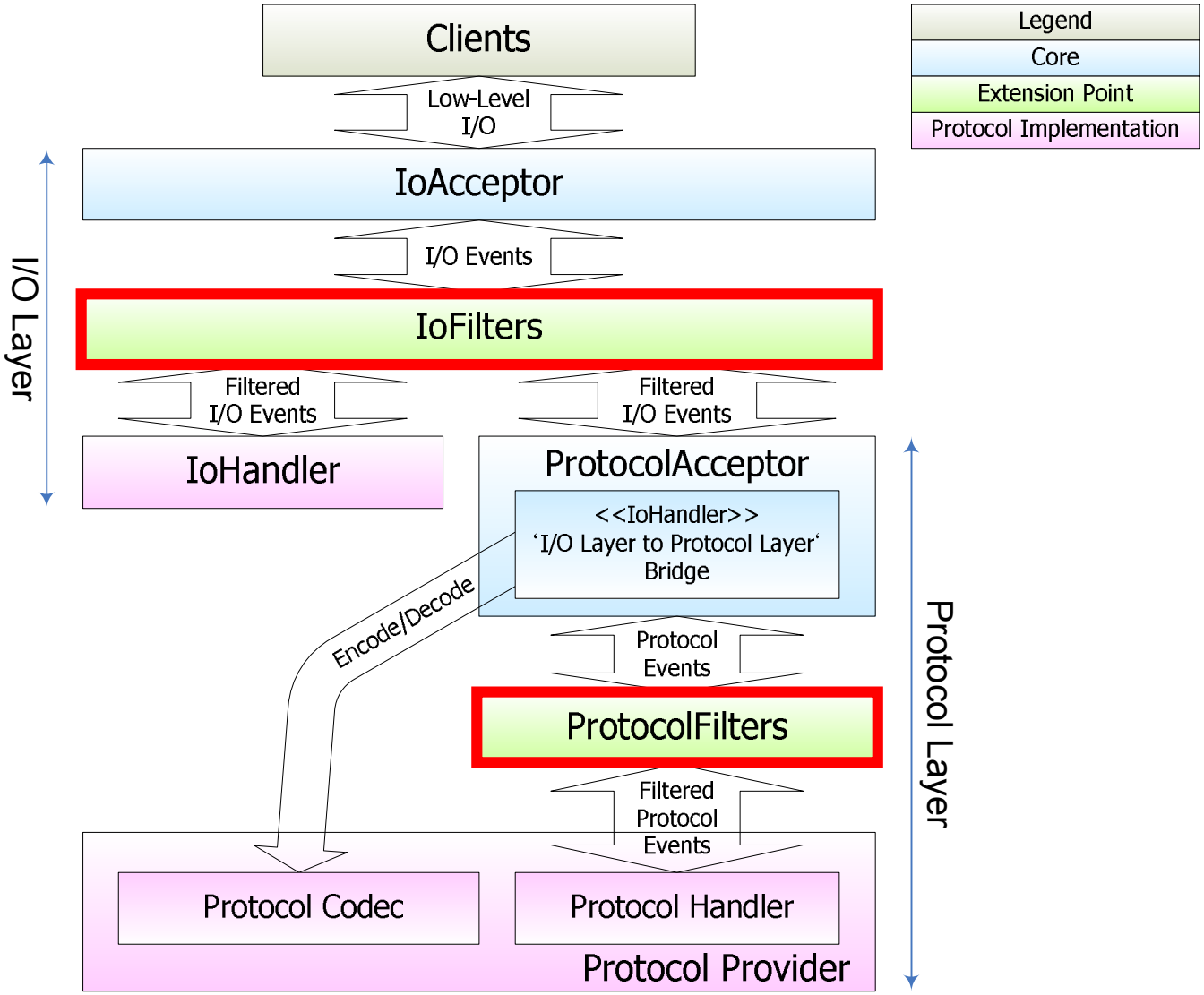
- The second way to implement your protocol:
  - Using **Protocol Layer**: `ProtocolProvider`
    - You communicate by exchanging objects (POJO).
    - Your codec performs transformations between data buffers and message objects.
      - Reusable
      - Pluggable (thanks to polymorphism)
    - You take full advantage of OOP for message objects
      - Inheritance

What You Should Do (Cont'd)



Filter Mechanism

Architecture (with Filters)



What is Filter

- A reusable event interceptor
  - Similar to Servlet filters
- Can be added and removed “on-the-fly”
- Works in both coarse- and fine-grained way:
  - Per Server Port
  - Per Individual Session

Filter Use Cases

- Implemented filters:
  - Thread pool (= customizable thread model!)
  - SSL
  - Client blacklisting
- Coming soon:
  - Logging, Profiling, StartTLS, Peak Point Control, Traffic throttling, Firewall, and many more ...
  - Any contributions are welcome!

Filter Use Cases (Cont'd)

- Customizable thread models
  - MINA runs in single thread mode by default
    - ⇒ Good for low-latency apps
  - Add a **`ThreadPoolFilter`** to make MINA multi-threaded
    - ⇒ Good for high-scalability apps

Proof of Productivity

Comparison

	Plain NIO	MINA
Echo server	109 lines*	50 lines (45%)
Reusability	Poor	All reusable: Filters, Codecs, Handlers
Maintainability	Poor	Very good

*) A Core Java Tech Tips example

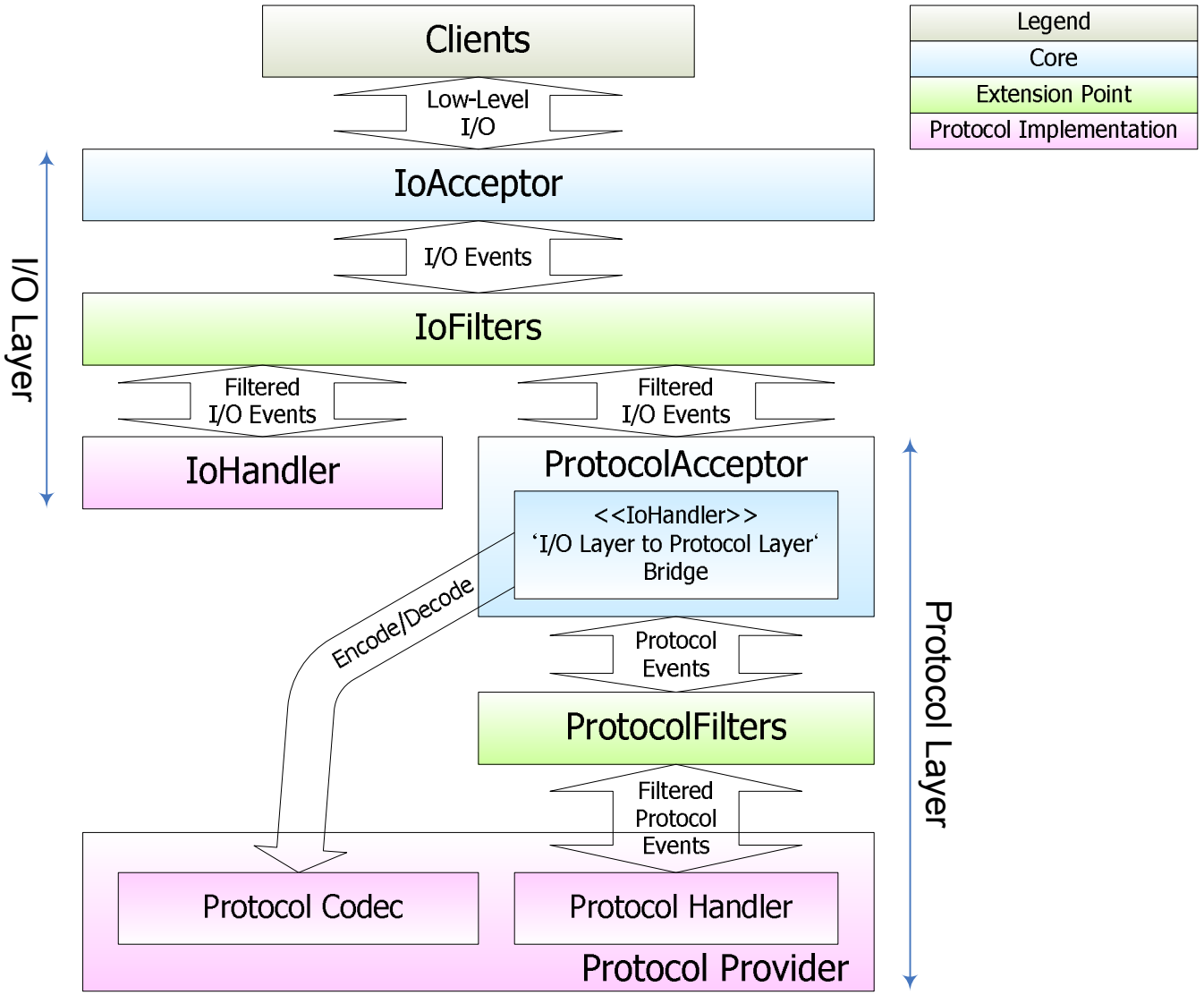
- 100% CPU consumption while socket buffer is full. (doesn't register for OP_WRITE)
- No SSL support (never trivial)

More Complex Protocols

- Even echo server is hard to maintain.
  - Writing complex protocols with plain NIO is the beginning of your nightmare.
- MINA Protocol Layer is your cozy pillow.
- Known implementations:
  - LDAP
  - SMTP
  - DNS
  - Kerberos
  - IMAPv4
  - NTP

Architecture Review

Architecture Review



Conclusion

Conclusion

MINA is
a flexible and extensible
network application framework
that boosts developer productivity.

How to Contribute

- MINA is a subproject of
  the Apache Directory Project
  - Homepage:
    http://directory.apache.org/subprojects/network
  - Mailing List:
    dev@directory.apache.org (Use '[mina]' prefix)