Channel - A C++ Template Framework for Distributed Message Passing

Yigong Liu
Topics

- Introduction
- Template generic programming
- Plan9 namespace
- Channel dynamic configuration - pub/sub namespace management
- Channel static configuration - polymorphic channels
- Examples
- Q&A
Introduction (1)

template
<
class IdType,
class IdTrait = IdTrait<Id_Type>,
class SynchPolicy = ACE_MT_SYNCH,
class DispatchPolicy = BroadcastPolicy,
class Router = MapRouter<Id_Type, Id_Trait, SynchPolicy, DispatchPolicy>
>

class Channel
The design of message passing system involves many aspects:

- static configuration: routing/dispatching/...
- dynamic configuration: pub-sub scope/remote connection/...

C++ template framework to allow users customize these aspects by choosing or designing proper trait/policy classes and create publish-subscribe message passing system **best-fit for a particular application**
Template Generic Programming

• C++ template is Turing complete (compile time computation):
  – template specialization as conditional constructs
  – template recursion as looping construct

• Generic programming techniques:
  – trait/policy classes
  – compile-time/static method dispatch
  – structure customization
template <int N>
struct Factorial
{
    enum { value = N * Factorial<N - 1>::value  };
};

template <>
struct Factorial<0>
{
    enum { value = 1 };
};
Plan9 Namespace (file-system)

- every resource (local/remote) is represented as a hierarchical file system:
  - window system, network stack, ...
- each process has a private mutable view/namespace of system resource
  - processes can customize its namespace and have different views
- remote resource sharing thru 9p protocol
Channel Dynamic Configuration - pub/sub namespace management

- Channel – a process local namespace.
- Peers (threads/callbacks) communicate thru channels by pub-sub messages(Ids).
- Remote channels can be connected for distributed communication.
- Publish/subscribe scope control
  - local, remote, global
- Namespace "merge" operations
  - A -> B, B -> A
- translators and filters
Channel Static Configuration - Polymorphic Channels

- By instantiating channel template with proper designed trait/policy classes, obtain a best-fit messaging framework for application
- Id_Type and Id_Trait
- Routing Data Structures and Algorithms:
  - Hash/Map
  - Trie/tree and pathname prefix matching
  - Associative matching
- Dispatching Algorithms
  - Broadcast, RoundRobin, Random, ....
typedef Channel<int> Chan;

......

struct StructId {
    int family;
    int type;
};

typedef Channel<StructId> Chan;
Examples (2)

typedef StringPathId<'/'> IdType;

typedef Channel<IdType, IdTrait<IdType>,
ACE_MT_SYNCH, RoundRobin Dispatcher, TrieRouter<IdType,
IdTrait<IdType>,
ACE_MT_SYNCH, Round Robin Dispatcher>
> Chan;
http://channel.sourceforge.net