Lesson 16 – A Practical SOA Example

Service Oriented Architectures

Module 3 - Resource-oriented services

Unit 2 – Examples

Ernesto Damiani

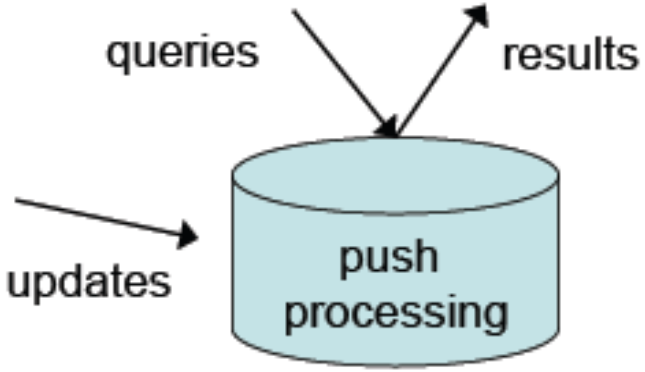
Università di Milano

XTream – SOA Driven to the Extreme

- Runtime platform and model for highly distributed, pervasive data stream processing
- Built on top of OSGi
- Extensive use of services
  - interaction between components of XTream
  - external, predefined services (R-OSGi, Configuration Admin, Logging, etc.)

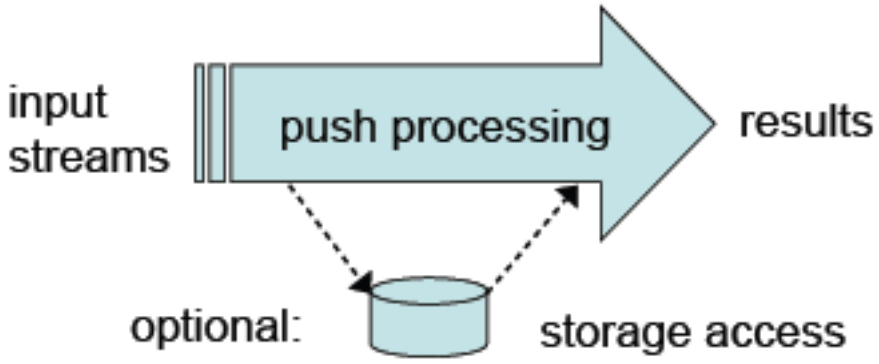
Stream Processing in a Nutshell (1)

- Traditional RDBMS
  - store data before processing
  - data is “static”
  - queries come and go



Stream Processing in a Nutshell (2)

- Stream processing
  - process data immediately at arrival
  - queries are “static”
  - data comes and goes



“Traditional” Streaming Applications (1)

- Network monitoring
  - intrusion detection
  - load monitoring
- Financial markets
  - quote updates
  - automatic trading
- Military

“Traditional” Streaming Applications (2)

- Well-defined application boundaries
- Logically centralized orchestration
- “Classic” requirements
  - low latency
  - high throughput

Highly Distributed Pervasive Data Stream Processing

- Federation of large number of heterogeneous, independent, autonomous, and widely distributed sources, sinks, and processors into a highly dynamic, loosely coupled mesh
- Primary foci differ from those of traditional stream processing
  - handle dynamism imposed by autonomy of entities
  - ensure privacy, confidentiality, and integrity of data

Processing and Exchanging Personal Information Streams

- Main motivation for XTream project
- Perfect instance of “highly distributed pervasive data stream processing”
  - covers technical properties and challenges
  - matches primary foci
  - strikingly simple
  - applicable to millions’ of people every day life

The Past

- Advances in networking, computing, and devices
- Proliferation of data sources
  - media (photos, video clips)
  - text (e-mail, blog entries, chat messages)
  - machine-generated data (sensor data, notifications)
- Possibility to access data sources from anywhere and at any time

Challenges

- Buffering of data
- Processing of data
- Distribution and dissemination of data
- Combination and interaction of applications
- Context dependency

The Present

- Custom solutions (standalone programs)
- Heavy engines (data base and stream engines)
- Web 2.0
  - sharing of media (e.g., Flickr, YouTube)
  - exchange of text (e.g., webmail, browser chats)
  - use of machine-generated data (e.g., Google calendar, RSS weather feed)
  - mashups

Issues

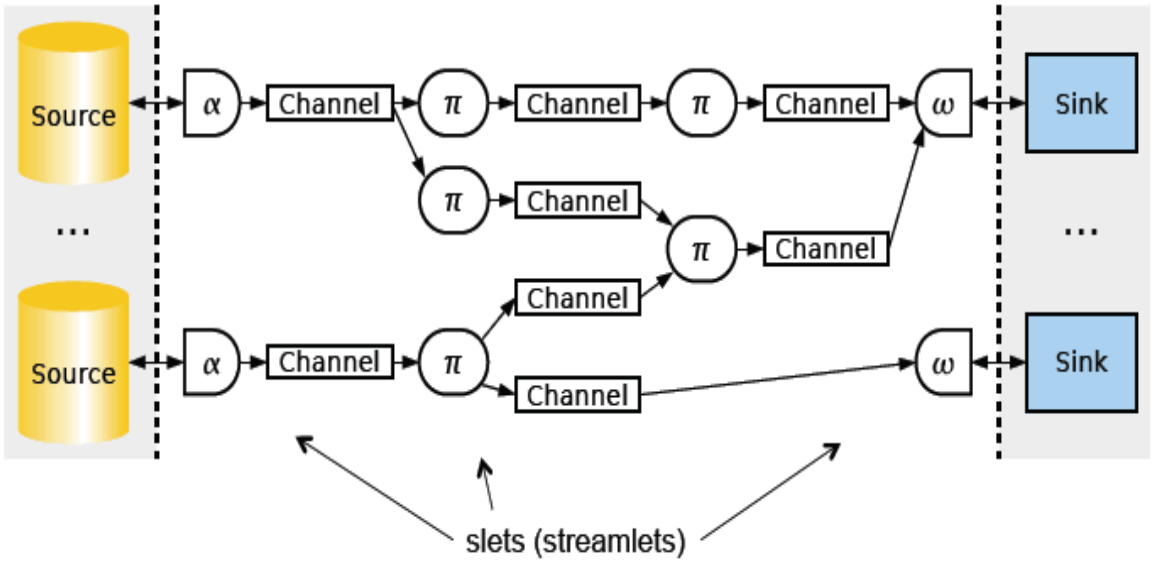
- Scalability (centralized infrastructure of Web 2.0)
  - more and more content created
  - interest for particular data restricted to small group
- Programmability
  - HTTP / web app model not designed for push
  - limited extensibility
- Privacy concerns

The Future

- An open and extensible platform that enables
  - everyday users to easily process personal information
  - groups of users to easily exchange information
  - developers to write extensible, interoperable apps
- A programming model that supports and deals with dynamic changes
- Direct communication between nodes

XTream

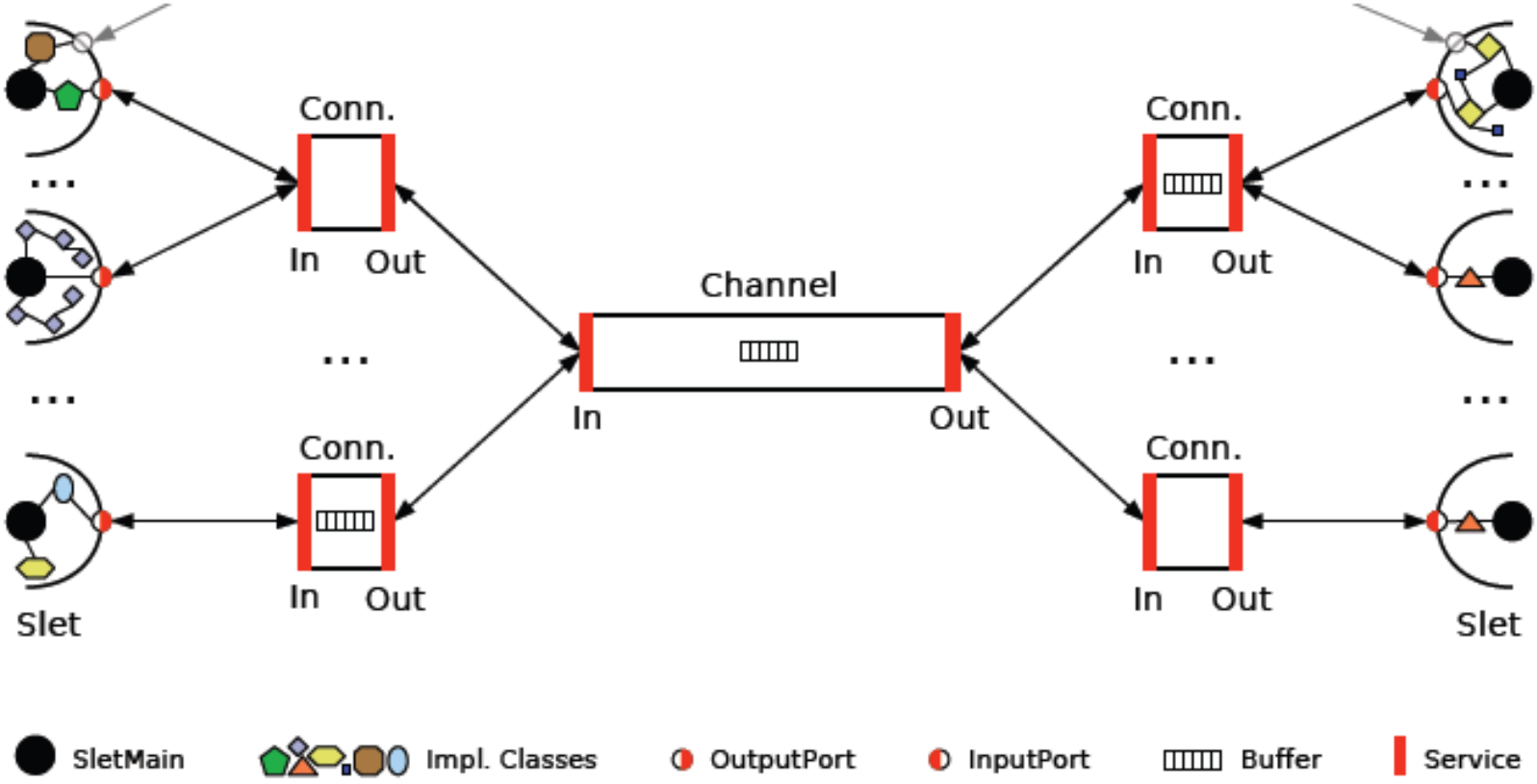
- Generalize data stream processing model



Generalizing the Data Stream Processing Model

- Treat personal information as data streams
  - new e-mails arriving
  - chat messages
  - …
- Integrate push and pull into slets and channels
- That’s all nice, but where’s the link to SOA…

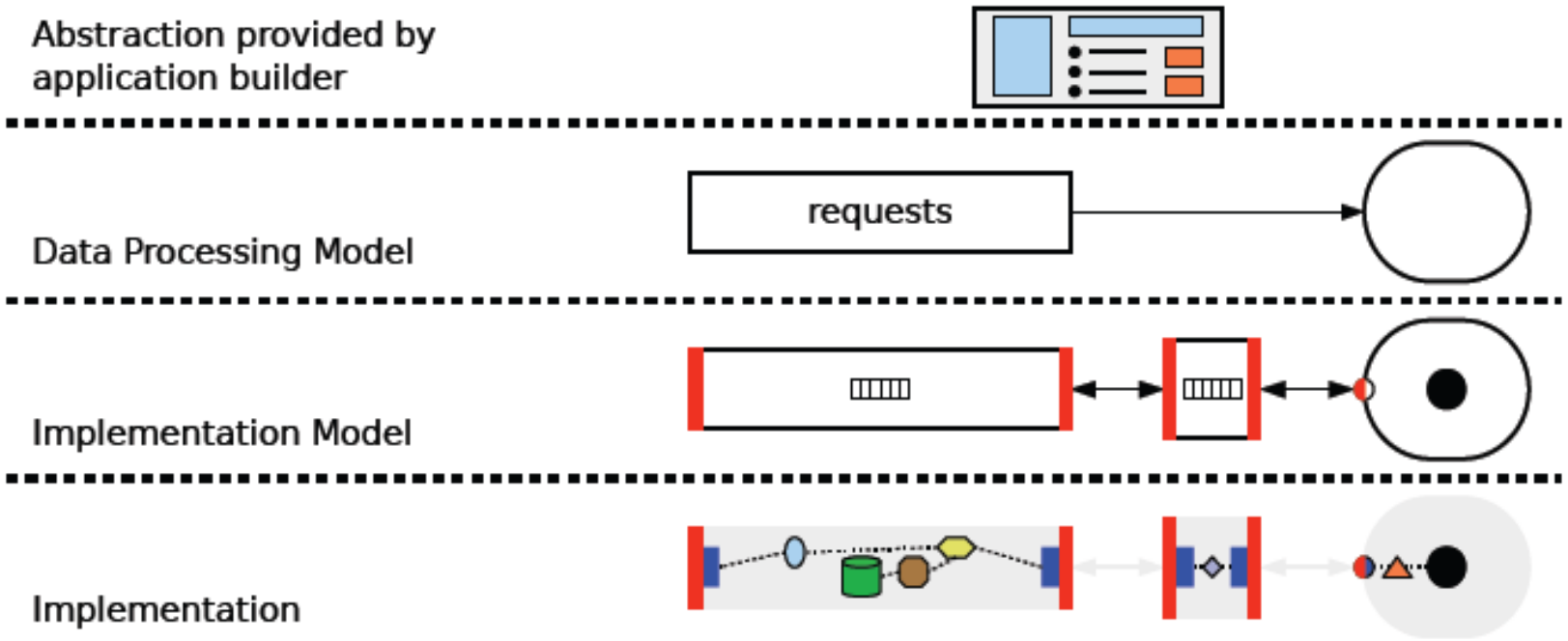
Services Ahead: Implementation Model



Implementation Model

- Loosely coupled components
  - interact with each other through services
  - can come and go at any time
- Connectors added as indirection
  - between slets and channels
  - entity in the model that covers communication

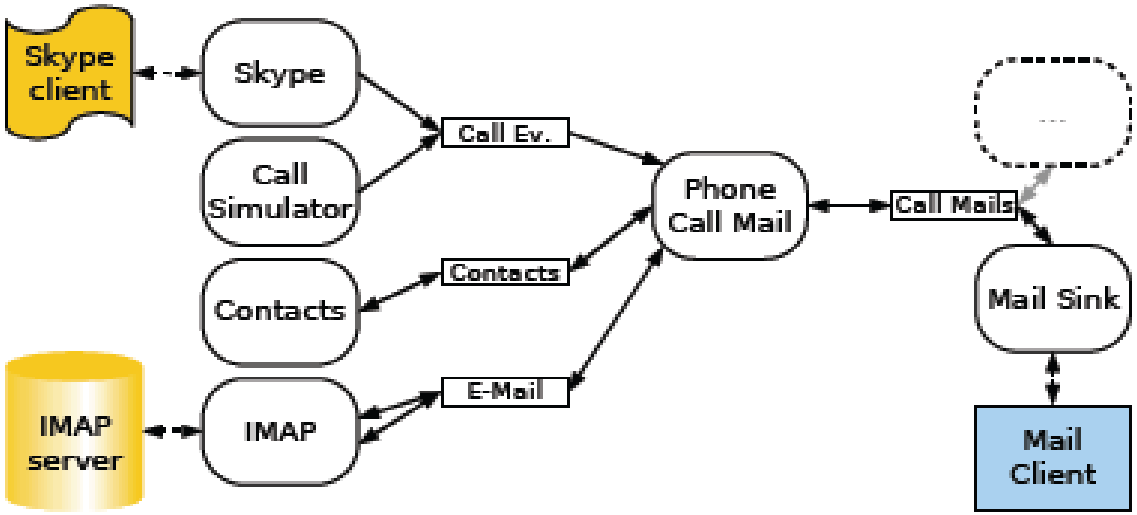
Programming Abstraction



Use Case: SkypeMail (1)

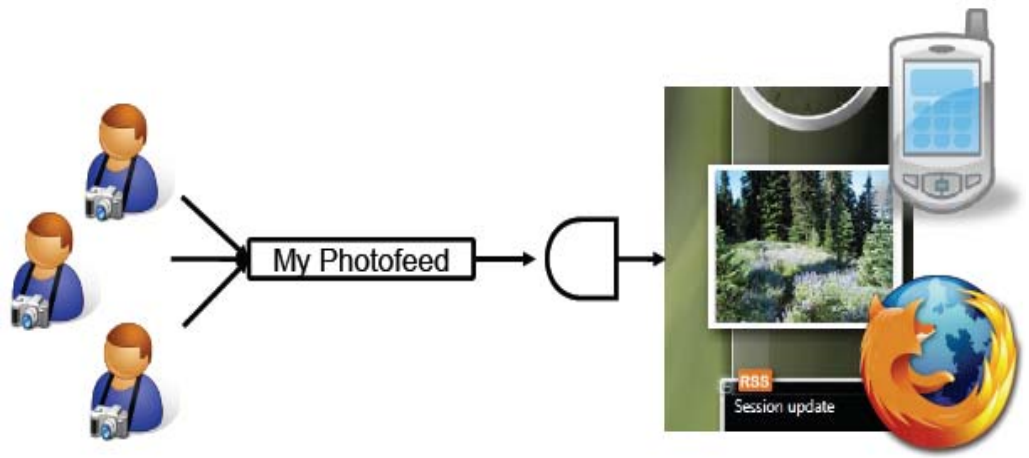
- Display e-mails exchanged with caller
- Plain and distinct application that covers a set of key challenges
  - heterogeneous data
  - push / pull
  - event-triggered
  - processing
  - reusability

Use Case: SkypeMail (2)



Use Case: Photo Exchange

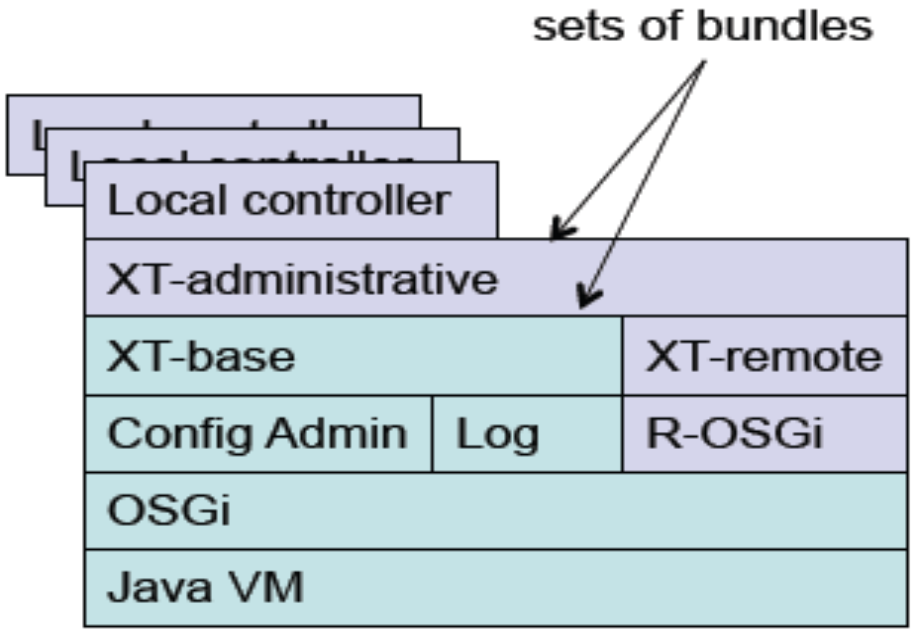
- Exchange recently taken photos with friends
  - each user accesses them in a different manner
  - aggregation of different photo feeds
  - independent computing systems



The Big Picture (1)

- Mandatory
  - Java VM, OSGi runtime
  - Config Admin, Log svc.
  - XT-base bundles
- Optional
  - XT-administrative bundles
  - XT-remote bundle
  - Controllers

The Big Picture (2)



XTream Base Bundles (1/2)

- Mandatory for operation
  - Core
    - exports common API (item container, channel interfaces, etc.)
    - helper methods
    - library bundle
  - Slet
    - exports slet API for concrete slet implementation bundles
    - implementation of common code: input and output ports
    - library bundle

XTream Base Bundles (2/2)

- Channel
  - provides implementations of XTream channels
  - registers a *ManagedServiceFactory* service to receive configuration
  - data for channels
- Connector
  - provides implementations of local XTream connectors
  - registers a *ManagedServiceFactory* service to receive configuration data for connectors

XTream Administrative Bundles

- Optional, can be loaded and unloaded at runtime
  - Monitoring
    - exports API for clients that monitor an XTream system
    - tracks clients and notifies them of changes
  - Management
    - exports API for clients that manage an XTream system
    - provides services for installing slet implementations, creating instances of slets or channels, wiring slets to channels, etc.

Slet Implementations

- Provide the actual functionality of an slet
- Implement slet API exported by slet bundle
- Plain JARs with one mandatory manifest entry:
- SletMain Class (name of slet main class)
- Multiple instances of the same type of slet can be created

XTream Remote Bundle

- Optional, needed for distributed operation
- Uses R-OSGi to communicate with remote peers
- Provides implementation of remote connectors

Local Controller Bundles

- Use monitoring and management bundles to interact with the system
  - install slet implementations
  - create instances of slets and channels
  - connect slets to channels
  - parametrize instances of slets
- Can be loaded and unloaded at runtime

The Whiteboard Pattern

- Decouple event listener and event source through the OSGi service registry
  - event listener registers itself as service
  - event source fetches and calls all listener service when an event is to be dispatched
- Big plus: no management of (stale?) registrations
- Whitepaper: http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf

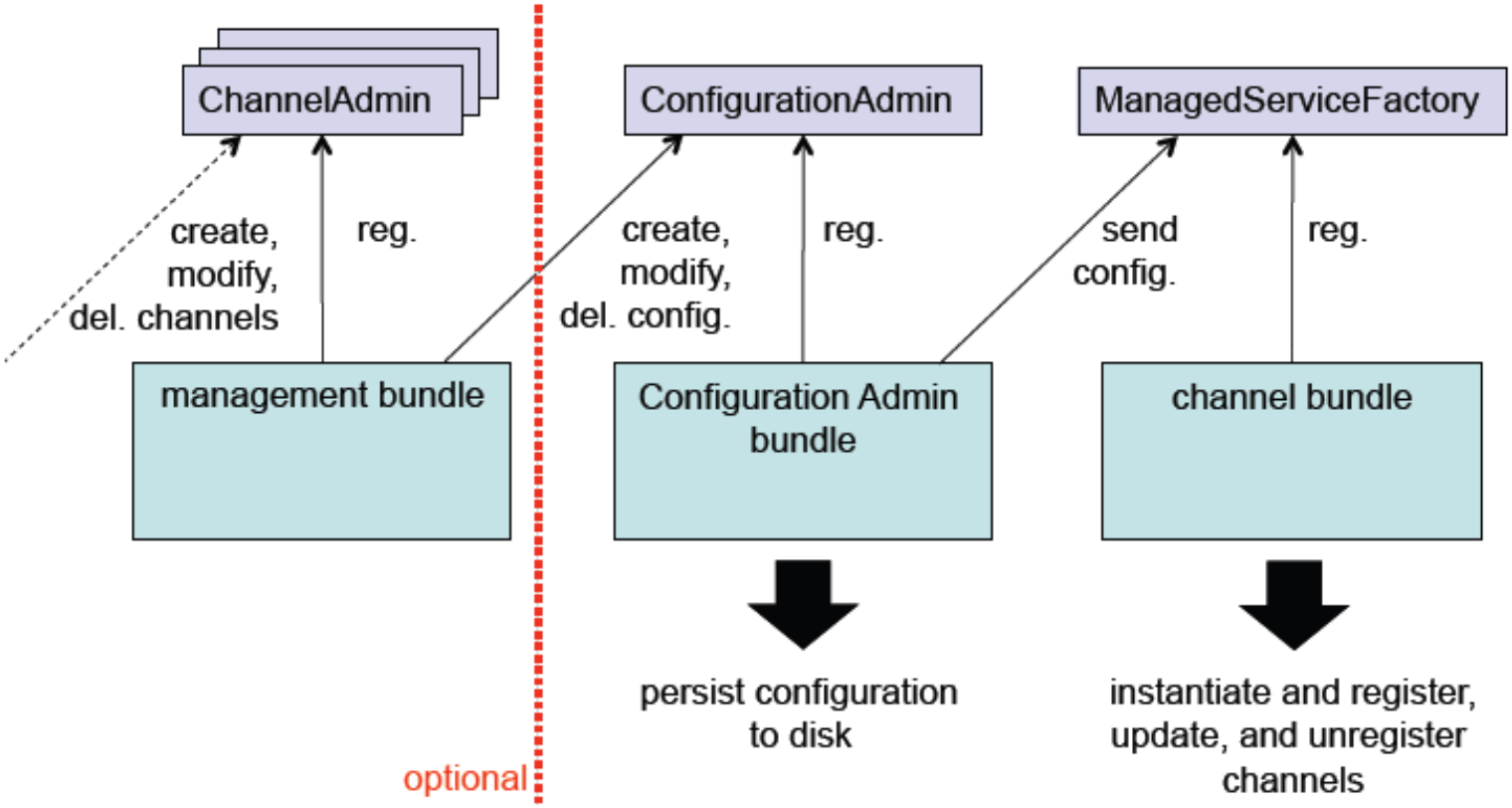
Whiteboard Example: Monitoring Bundle and its Clients

- Monitoring bundle exports API for listeners: SletListener, WiringListener, PortListener, etc.
- Clients implement these interfaces and register the implementing class as service
- E.g., if an slet creates an input port the monitoring bundle fetches all PortListener services and calls sletCreatedInputPort(...) on them

Configuration Admin Service

- Persistently saves configuration for services
  - for services (service is managing itself)
  - for factories (service factories manage services)
- When a managed service or managed service factory is registered, the Configuration Admin service passes the persisted configuration to it
- Extensively used in XTream to persist state

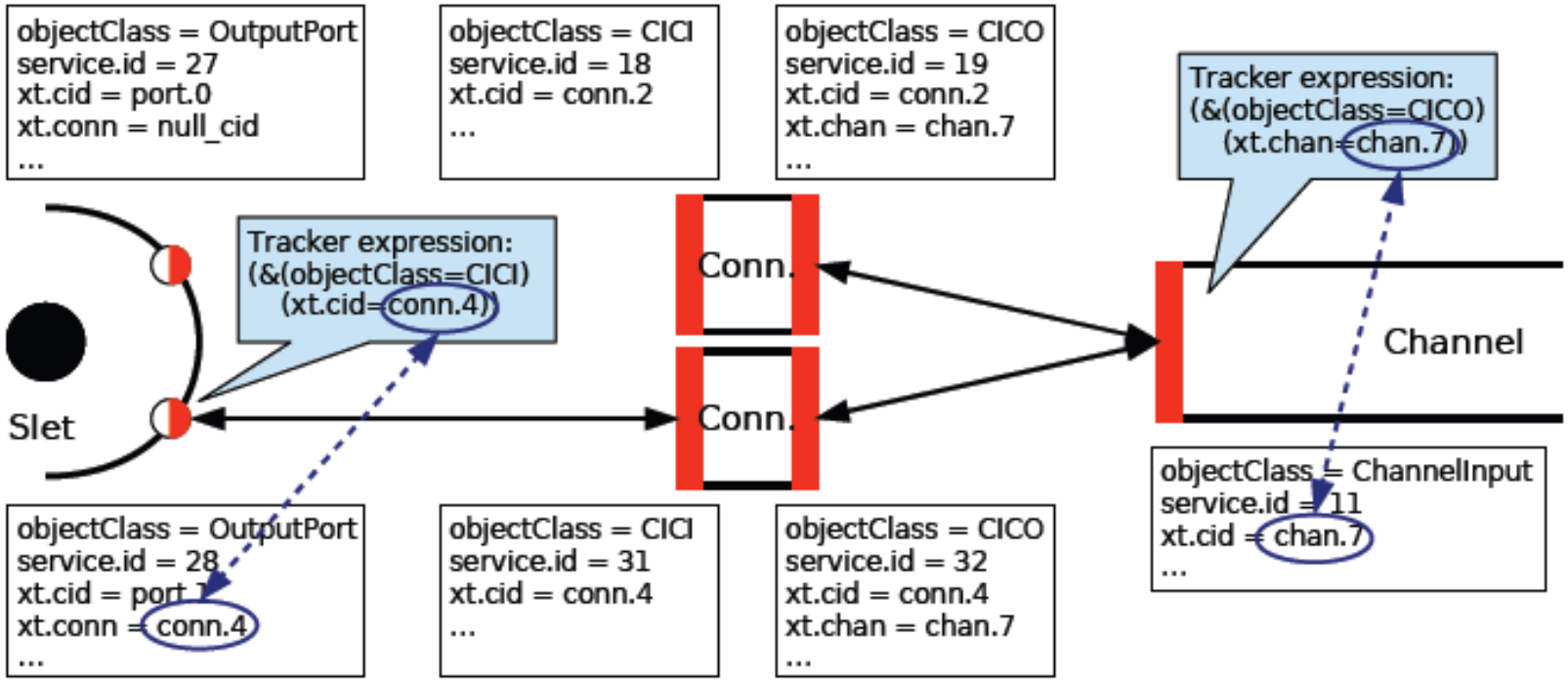
Example of Configuration Admin in XTream: Channels



Efficient Implementation of Fully Dynamic Binding (1/2)

- Use OSGi service tracker
  - proactively tracks and caches services
  - retrieve up-to-date array of tracked services (Java objects) when interacting with other components
  - proactive tracking only causes overhead once every time bindings actually change
  - binding between 2 components recorded in one place
    - consistency
    - performance (length of tracker expression is O(1))

Efficient Implementation of Fully Dynamic Binding (2/2)



FINE