

Demaq: A Foundation for Declarative XML Message Processing

Alexander Böhm

Carl-Christian Kanne

Guido Moerkotte

University of Mannheim

XML Messaging

Introduction

● XML Messaging

● Networks of XML Queues

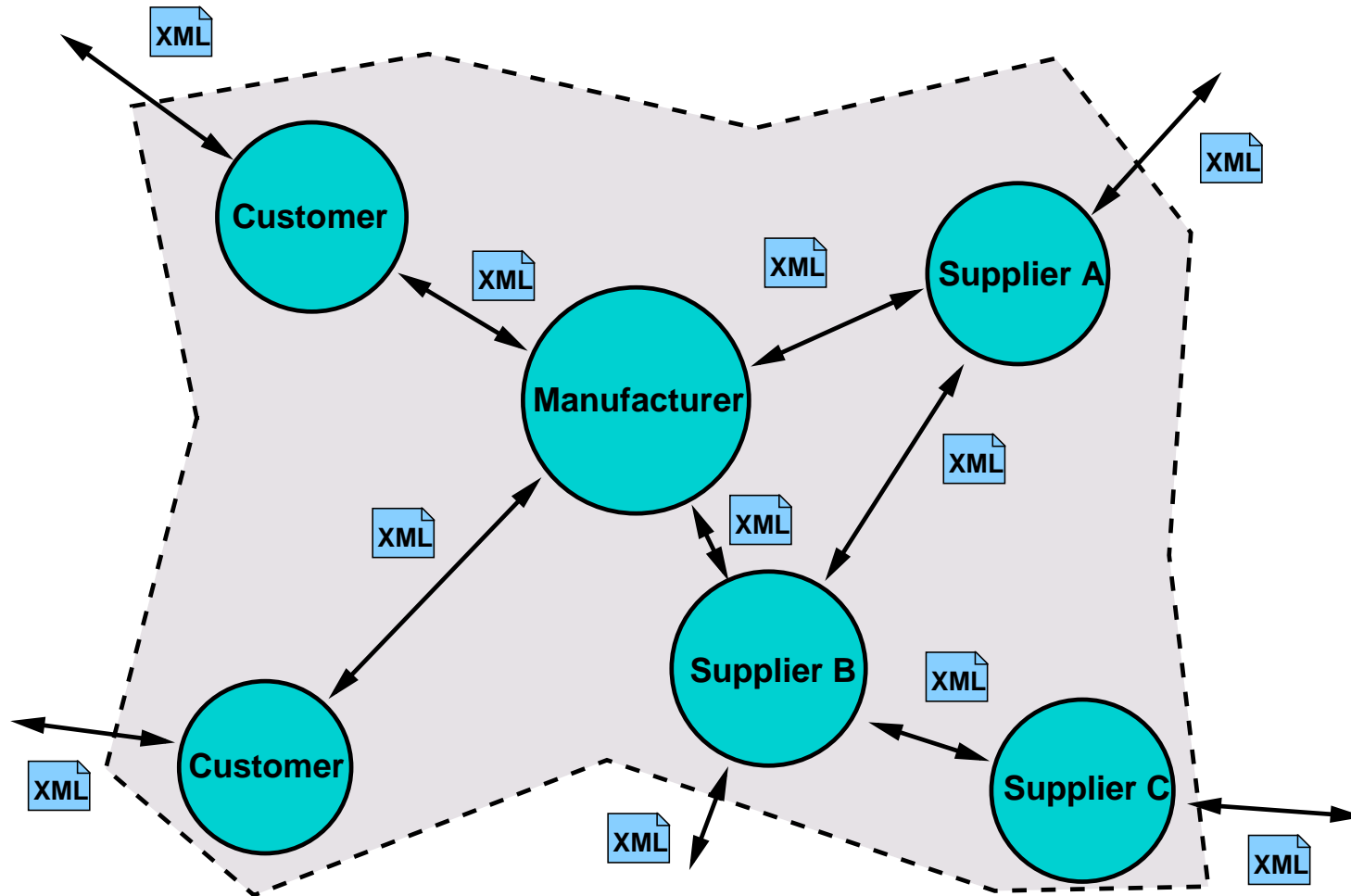
● Messaging Rules

● ∞-tier Architectures

● State of the Art

Demaq

Appendix



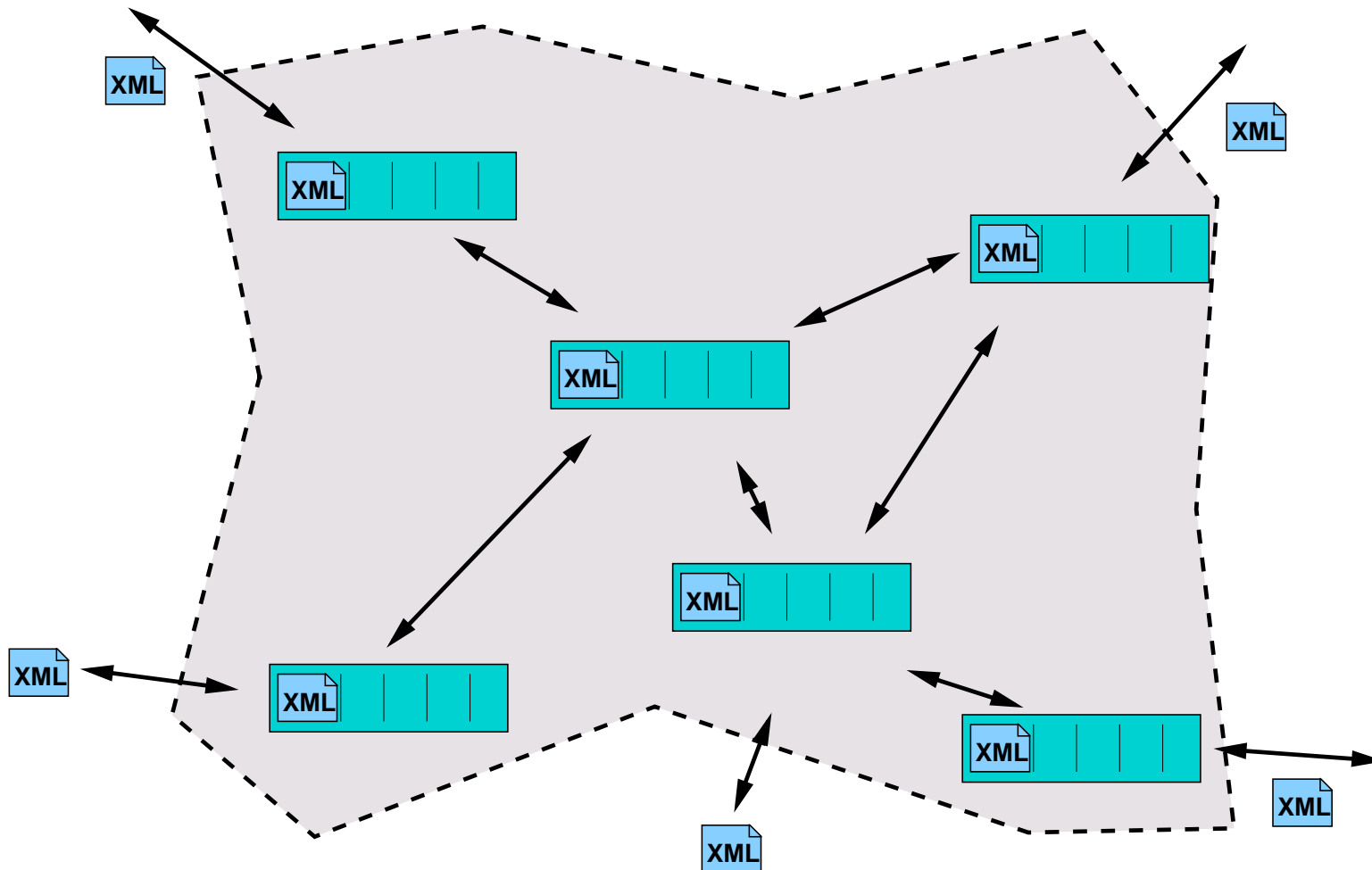
SOA, Web Services, AJAX, RSS/Atom...

Networks of XML Queues

- Introduction
- XML Messaging
- Networks of XML Queues
- Messaging Rules
- ∞-tier Architectures
- State of the Art

Demaq

Appendix



Messaging Rules

Introduction

- XML Messaging
- Networks of XML Queues
- Messaging Rules
- ∞-tier Architectures
- State of the Art

Demaq

Appendix

- "If a request for an offer comes in, forward it to the legal, finance, and planning departments"
- "If the delivery of all items has been confirmed, send a completion message to the customer"

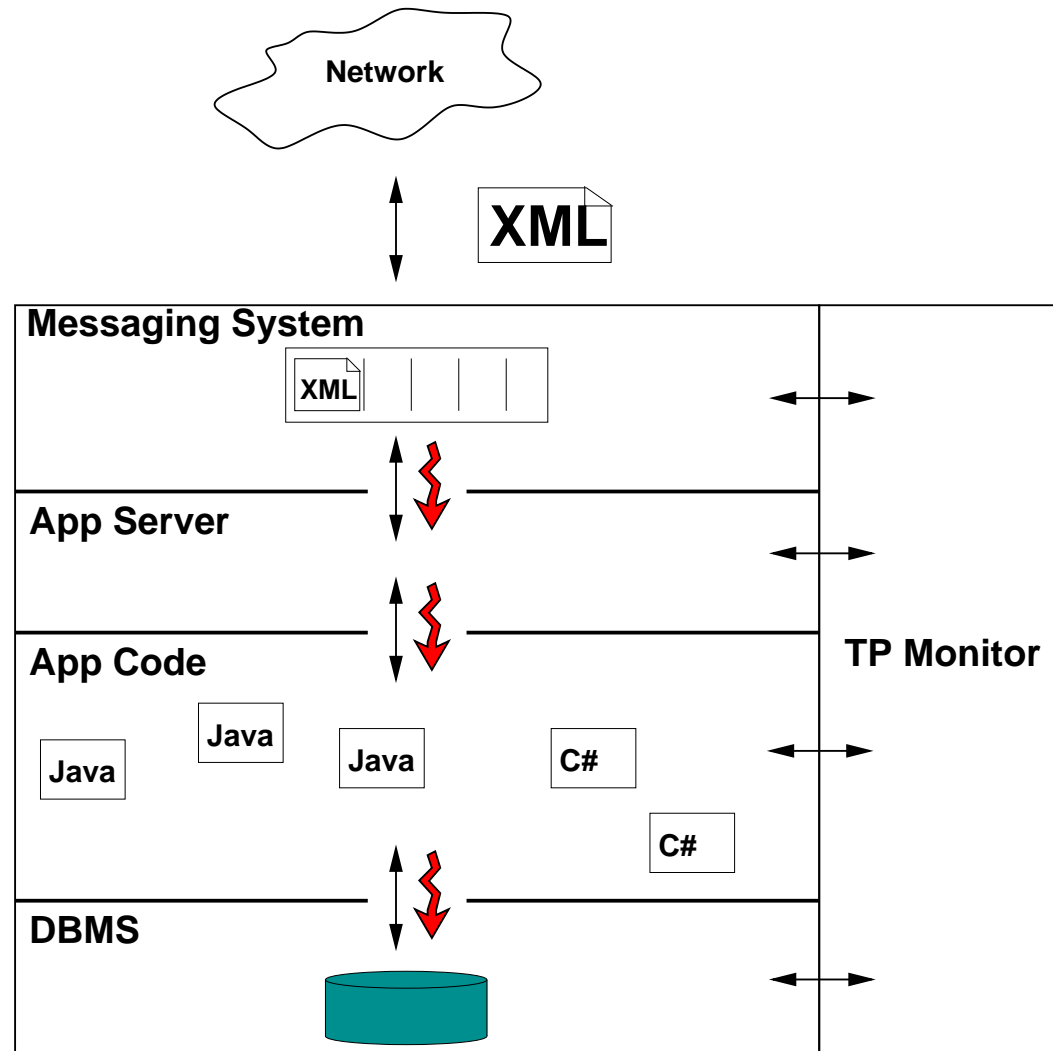
∞-tier Architectures

Introduction

- XML Messaging
- Networks of XML Queues
- Messaging Rules
- ∞-tier Architectures
- State of the Art

Demaq

Appendix



State of the Art

Introduction

- XML Messaging
- Networks of XML Queues
- Messaging Rules
- ∞-tier Architectures
- State of the Art

Demaq

Appendix

```
topic = ((QljmsSession)t_sess).getTopic("strmadmin", "oe_queue");
t_pub = t_sess.createPublisher(topic);
db_conn = ((QljmsSession)t_sess).getDBConnection();
agent = new QljmsAgent("explicit_enq", null);
adt_msg = ((QljmsSession)t_sess).createAdtMessage();
lcr_data = new StringBuffer();
lcr_data.append("<ROW_LCR ");
lcr_data.append(" xmlns='http://xmlns.tentacle.com/streams/schemas/lcr '");
lcr_data.append(" xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance '");
lcr_data.append(" xsi:schemaLocation='http://xmlns.tentacle.com/streams/schemas/lcr '");
lcr_data.append(" http://xmlns.tentacle.com/streams/schemas/lcr/streams_lcr.xsd' >");
lcr_data.append("<source_database_name>source_dbname</source_database_name >");
```

... MORE DOCUMENT CONSTRUCTION HERE ...

```
xml_lcr = tentacle.xdb.XMLType.createXML(db_conn, lcr_data.toString());
adt_msg.setAdtPayload(xml_lcr);
((QljmsMessage)adt_msg).setSenderID(agent);
System.out.println("Publish message 3 – XMLType containing LCR ROW");
recipList = new QljmsAgent[1];
recipList[0] = new QljmsAgent("explicit_dq", null);
((QljmsTopicPublisher)t_pub).publish(topic, adt_msg, recipList);
t_sess.commit();
```

Demag Application

Introduction

Demag

● Demag Application

● Demag Language(s)

● Demag QML Rules

● Demag Sample Rule

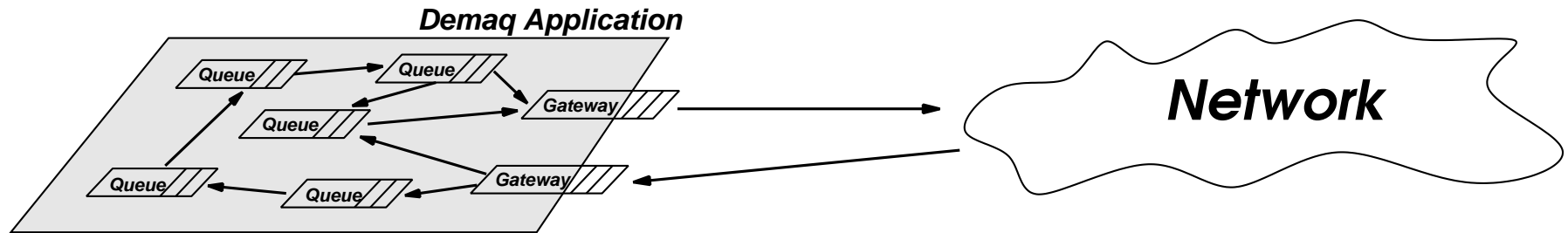
● Demag Server

● Messages all the way

● Demag Project

● Thank you

Appendix



- Complete
- Declarative
- Executable

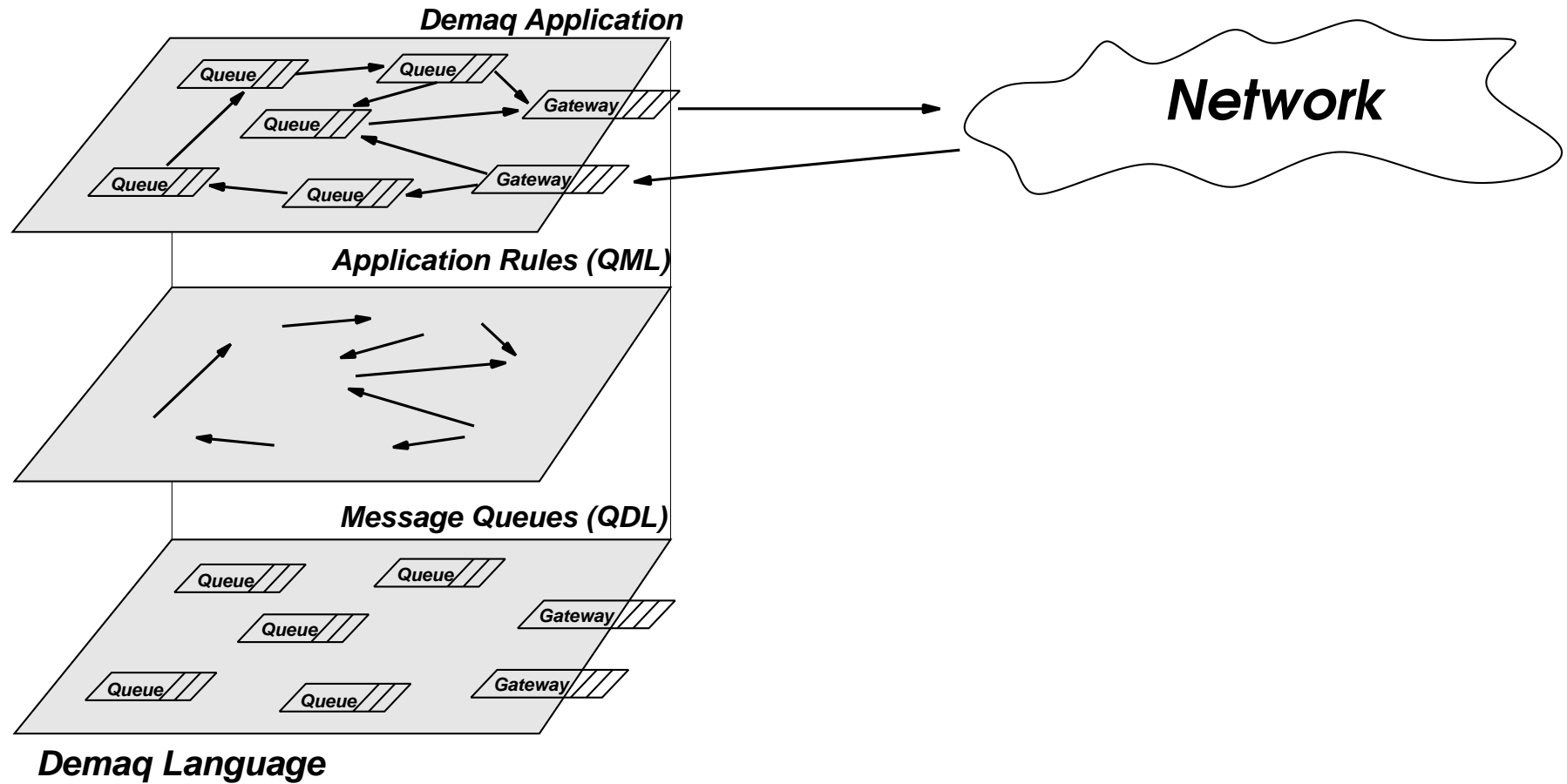
Demq Language(s)

Introduction

Demq

- Demq Application
- Demq Language(s)
- Demq QML Rules
- Demq Sample Rule
- Demq Server
- Messages all the way
- Demq Project
- Thank you

Appendix



Demaq QML Rules

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix

- "If the delivery of all items has been confirmed, send a completion message to the customer"

Demaq QML Rules

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix

- "If the delivery of all items has been confirmed, send a completion message to the customer"

XML messages



new XML messages

Demag QML Rules

Introduction

Demag

- Demag Application
- Demag Language(s)
- Demag QML Rules
- Demag Sample Rule
- Demag Server
- Messages all the way
- Demag Project
- Thank you

Appendix

- "If the delivery of all items has been confirmed, send a completion message to the customer"

XQuery Update Facility
+ Queuing Primitives

XML messages



new XML messages

Demaq Sample Rule

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix

```
create rule sendComplete for orderMsgs
if (// deliverymsg / @type = "confirm") then
  let $ordered := fn:count(qs:slice ()// ordermsg // item)
  let $delivered := fn:count(qs:slice ()// deliverymsg / item)
where $ordered eq $delivered
return do enqueue <done> {// orderID} </done>
  into customerReply
```

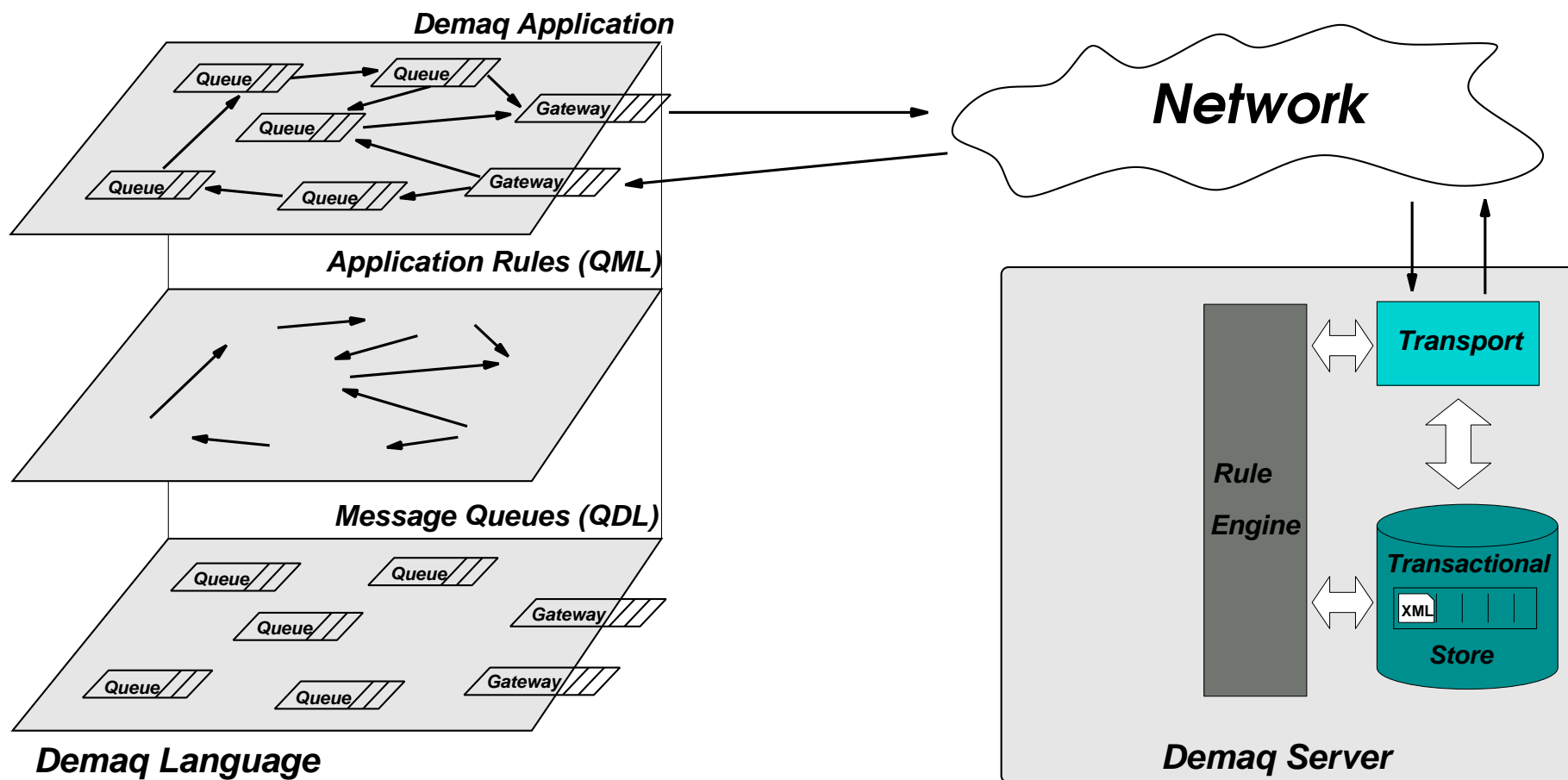
Demag Server

Introduction

Demag

- Demag Application
- Demag Language(s)
- Demag QML Rules
- Demag Sample Rule
- Demag Server
- Messages all the way
- Demag Project
- Thank you

Appendix



Messages all the way

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix

- Everything is an XML message
 - ◆ Rule Input
 - ◆ Rule Output
 - ◆ Errors
 - ◆ Timeouts
- Messages are processed once, but kept "forever"
- Message History
 - ◆ captures process state
 - ◆ organized into slices (virtual queues)
 - ◆ declarative expiration

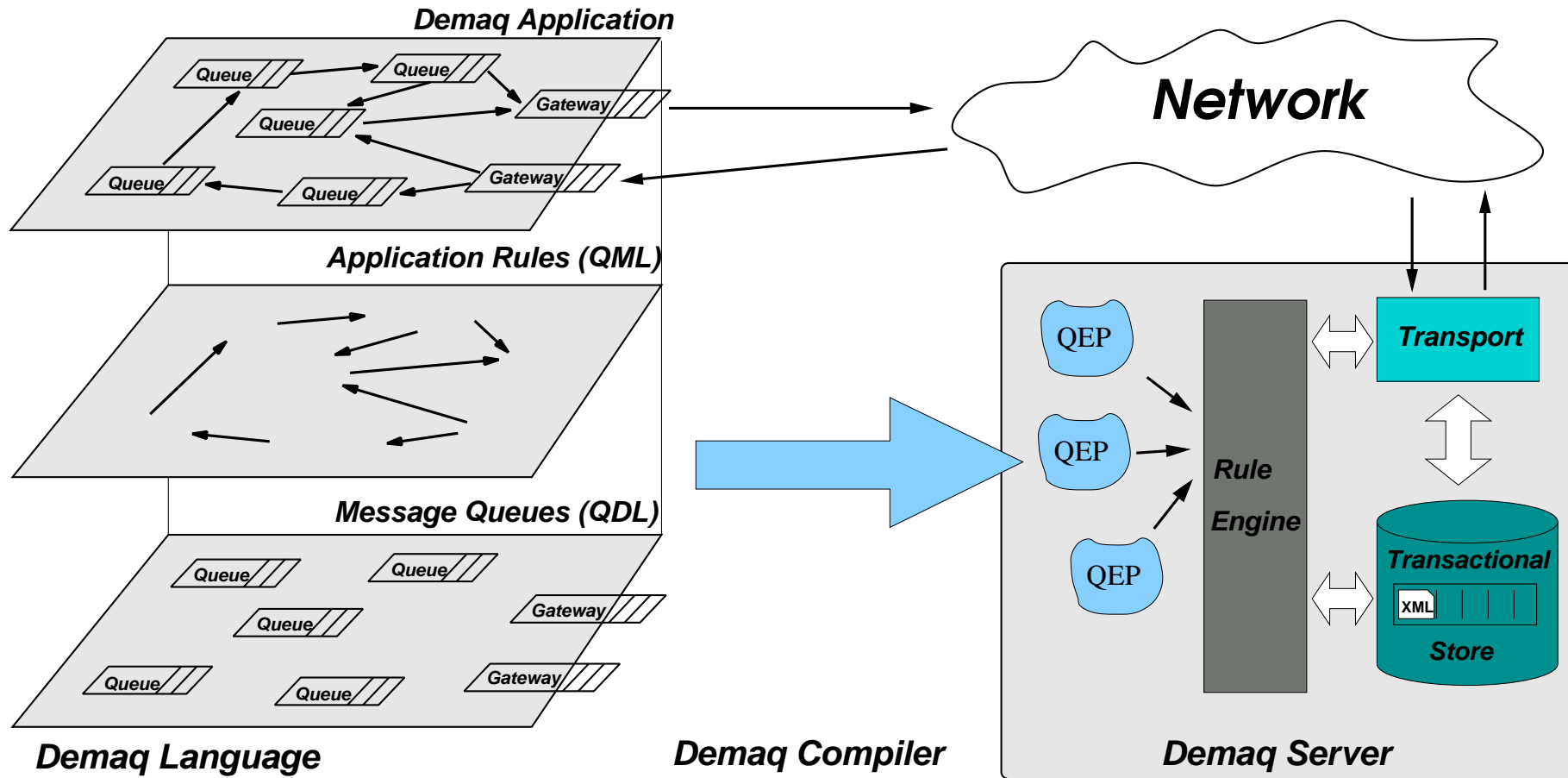
Demaq Project

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix



Thank you

Introduction

Demaq

- Demaq Application
- Demaq Language(s)
- Demaq QML Rules
- Demaq Sample Rule
- Demaq Server
- Messages all the way
- Demaq Project
- Thank you

Appendix

`http://demaq.net`

`http://db.informatik.uni-mannheim.de`

Slices

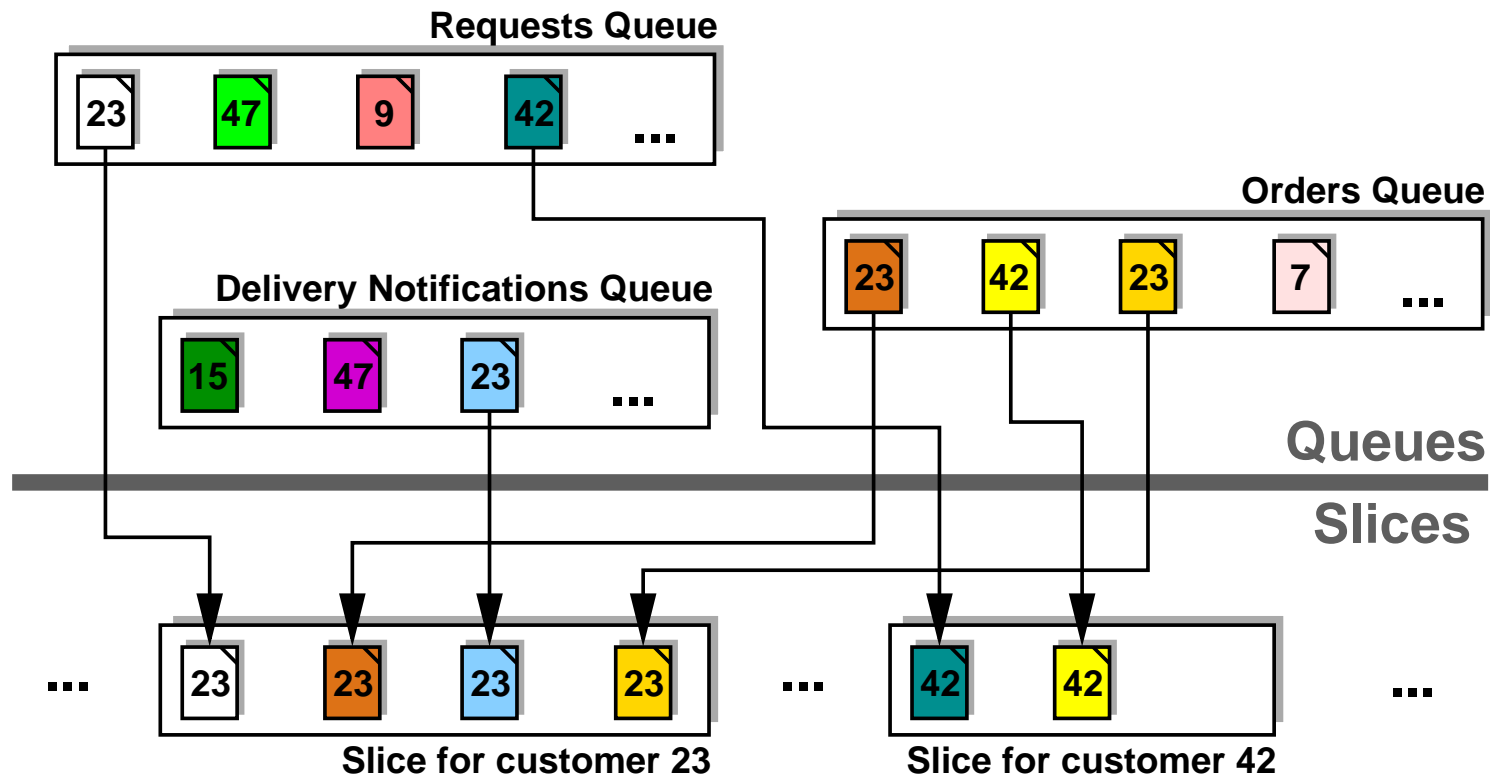
Introduction

Demaq

Appendix

● Slices

- Slice usage
- Error Handling
- A Demaq Rule
- Work in progress
- Demaq Goals



```
create property customerId fixed
queue requests, orders,
deliveryNotifications value //customerId
create slicing customers on customerId
```

Slice usage

Introduction

Demaq

Appendix

● Slices

● Slice usage

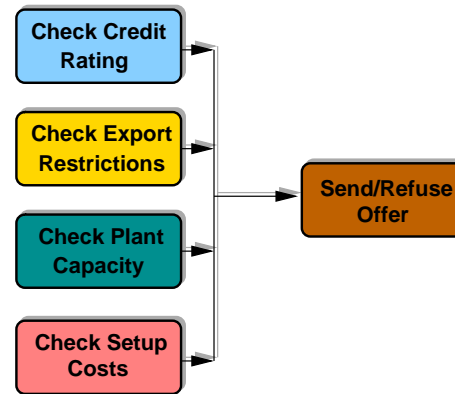
● Error Handling

● A Demaq Rule

● Work in progress

● Demaq Goals

■ Merge parallel control flow



```
create property correlationID fixed  
queue creditCheck , exportCheck ,  
plantCheck , setupCheck value // correlationID  
create slicing checkResults on correlationID
```

```
create rule merge for checkResults  
if (count(qs:slice ())) eq 4) then ...
```

Error Handling

Introduction

Demaq

Appendix

● Slices

● Slice usage

● **Error Handling**

● A Demaq Rule

● Work in progress

● Demaq Goals

- Plenty of error sources in distributed applications
 - ◆ Application-related (dynamic errors)
 - ◆ Message-related (invalid XML, wrong schema)
 - ◆ Network-related (disconnects, routing, ...)
 - ◆ ...
- Message-based error handling
- Error queues, e.g. for rules

```
create rule errorSource for foo errorqueue errors
```

A Demaq Rule

Introduction

Demaq

Appendix

- Slices
- Slice usage
- Error Handling
- A Demaq Rule
- Work in progress
- Demaq Goals

```
create rule newOfferRequest for customerMsgs
if (// offerRequest) then
  let $customerInfo :=
    <requestCustomerInfo reqID="{// requestID}" >
      <customer >{// customerID} </customer >
    </requestCustomerInfo >
  let $exportRestrictionInfo := ...
  let $plantCapacityInfo := ...
  return do enqueue $customerInfo into finance ,
    do enqueue $exportRestrictionsInfo into legal ,
    do enqueue $plantCapacityInfo into supplier
```

Work in progress

- Optimization across rules
- Optimization/verification across sites
- Template Folding [XIMEP06]
- Rules driven by XML Schema validation

Introduction

Demaq

Appendix

- Slices
- Slice usage
- Error Handling
- A Demaq Rule
- Work in progress
- Demaq Goals

Demaq Goals

Introduction

Demaq

Appendix

● Slices

● Slice usage

● Error Handling

● A Demaq Rule

● Work in progress

● Demaq Goals

- Declarative XML message processing language
 - ◆ Move work from programmer to system
 - ◆ Data independence
 - ◆ Optimizable
- Execution Engine
 - ◆ Reliability
 - ◆ Scalability
 - ◆ Reuse DB system knowledge