TeleScope

XML Data Stream Broker/Replicator Platform

Kirill Belyaev

Department of Computer Science

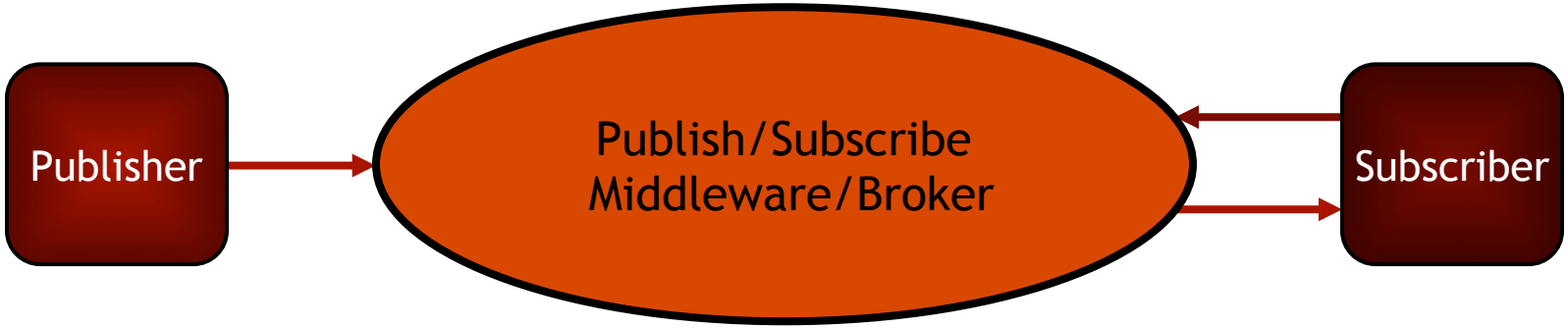
Colorado State University

Fort Collins, CO, USA

With a number of slides borrowed from Roberto Baldoni

Introduction: The Publish/Subscribe

- Communication Style



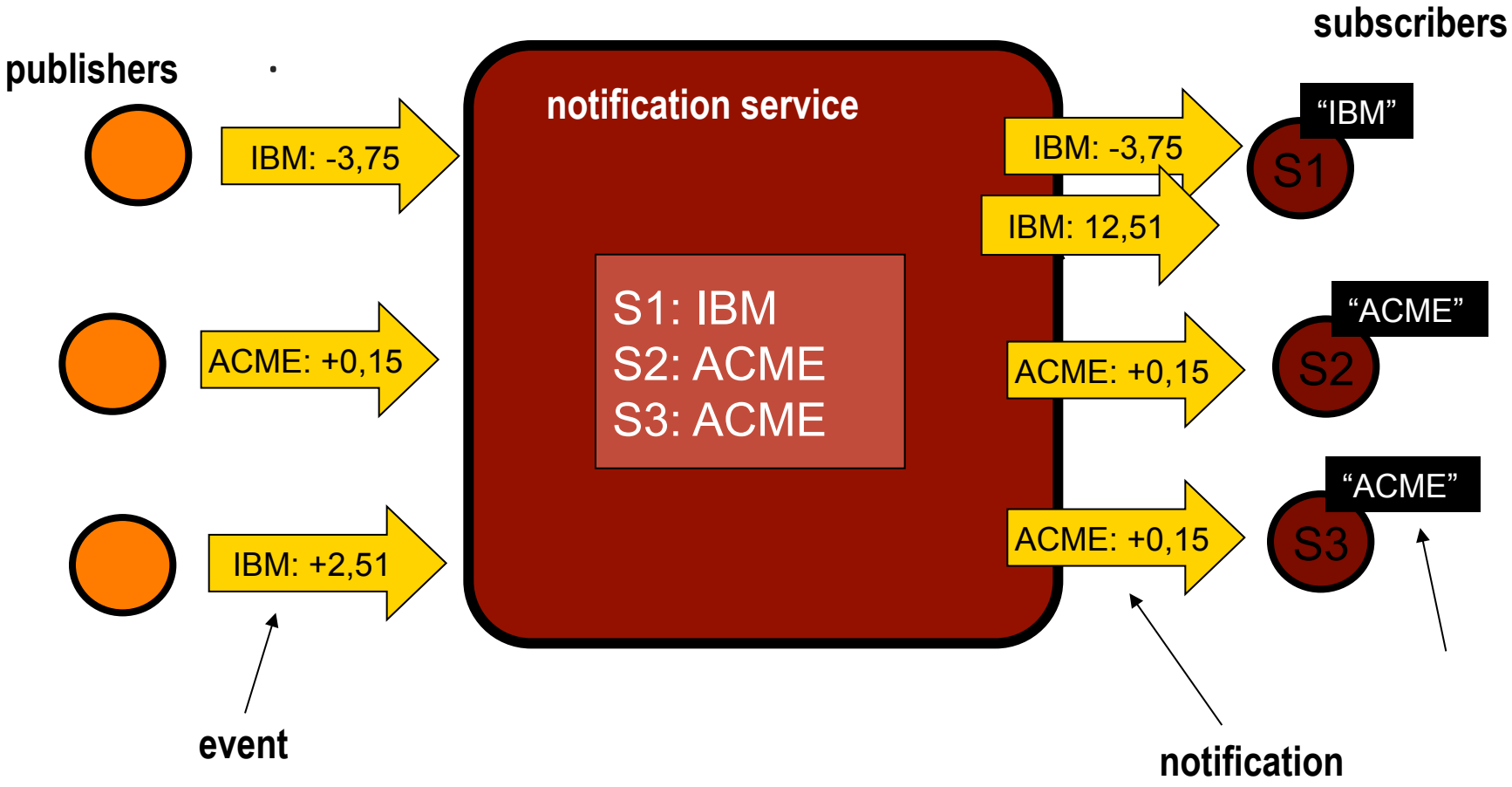
Introduction: The Publish/Subscribe

- Publish/Subscribe (pub/sub): a powerful abstraction for building distributed applications
  - Message-based, anonymous communication
  - Participants are decoupled
    - in space: no need to be connected or even know each other
    - in flow: no need to be synchronized
    - in time: no need to be up at the same time
- Good solution for highly dynamic, decentralized systems (e.g., wired environments with huge numbers of publishers and subscribers, P2P etc)
- Many research issues, involving several research areas (e.g., systems, software eng., databases etc)

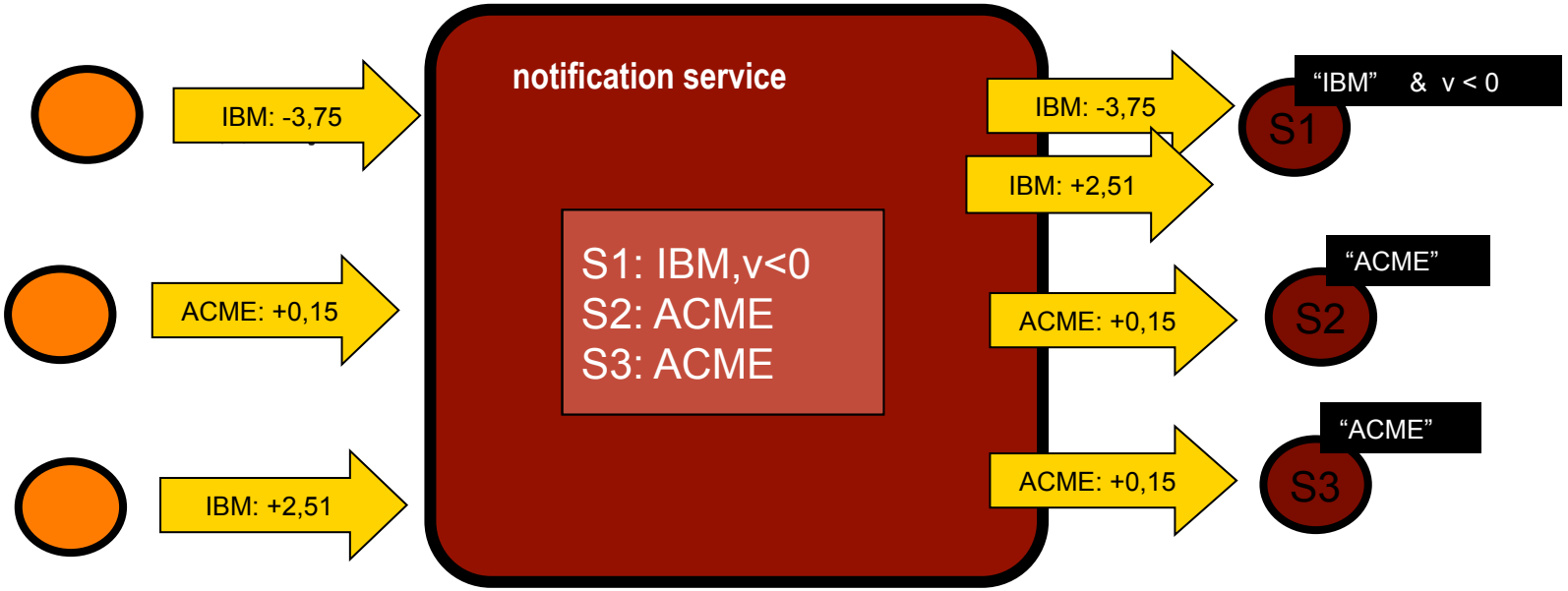
Introduction: The Publish/Subscribe

- Widely popular both in the industry and in the research
- Industry solutions for pub/sub are almost centralized
- Research has mainly to face the scalability aspect
- Applications
  - Stock information delivery
  - Auction system
  - Air traffic control
  - Web Services
- Many names for pub/sub systems: notification service, data distribution service

Basic Interaction Model



Subscription Models



Topic-Based [Oki et al. 93]:
- events are divided in topics
- subscribers subscribe for a single topic

Type-Based [Eugster 2001]:
- notifications are objects
- type is the discriminating attribute

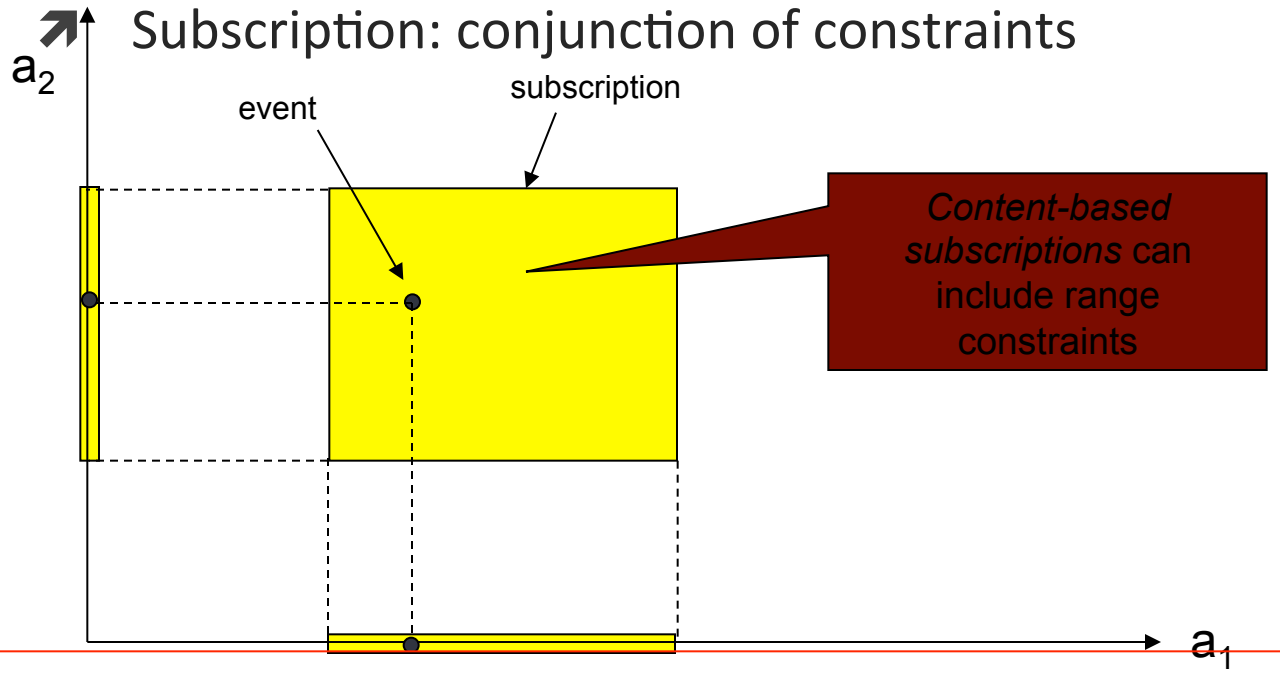
Content-Based [Carzaniga et al 2001]:
- subscriptions are generic queries SQL-like on the event schema

Pub/Sub Variants: Topic-based

- Event space is divided in topics, corresponding to logical channels
- Participants subscribe for a topic and publish on a topic
- Receivers for an event are known a priori
- Channel = Group
  - Therefore often exploit network-level multicast
  - Group communication

Content Based pub-sub: event schema

- Subscriptions and events defined over an n-dimensional *event space* (E.g. `StockName = "ACME" and change < -3`)
  - Subscription: conjunction of constraints



Topic vs. Content

- Topic-based pub/sub
  - Recipients are known a-priori
  - Many efficient implementations exist
  - Limited expressiveness
- Content-based pub/sub
  - Cannot determine recipients before publication
  - More flexible
  - More general
  - Much more difficult to implement efficiently

Content-based pub-sub

- Assumes publish-subscribe infrastructure
- But rather than limiting matching to "topics" goes further and allows queries against the actual content of messages (XML messages stream)
- Problem becomes one of matching at high speeds with a large number of subscribers expressing complex queries

Architecture Model of a pub/sub

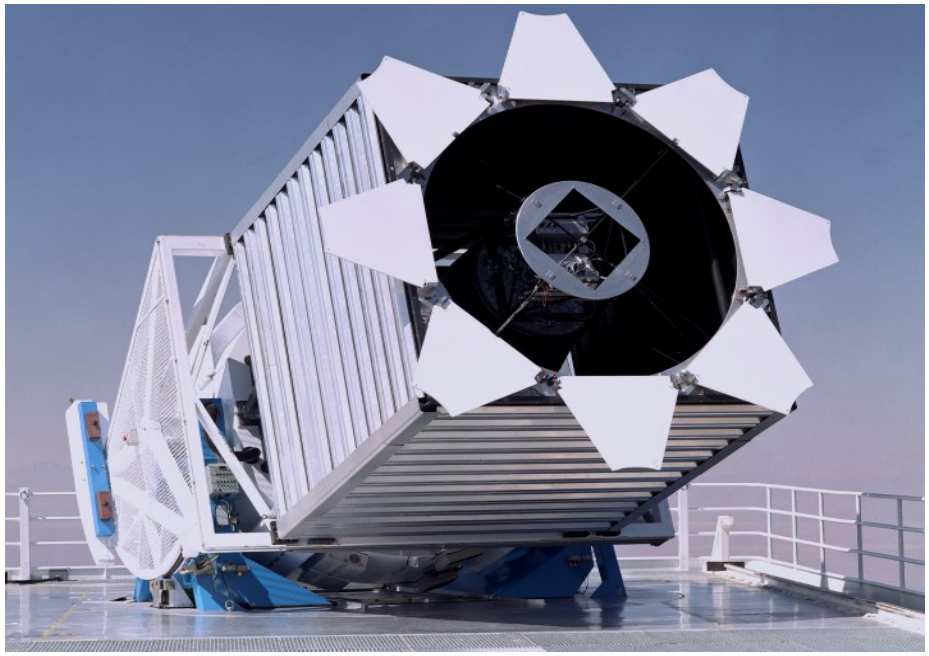
- Network Multicasting
  - Use of multicast networking facilities (also at data link level)
- Broker layer overlay
  - Based on transport level connections between nodes
  - Hierarchical (Decision tree from publisher to subscribers)
  - Undirected Acyclic graph spanning all brokers
- Structured Overlay
  - DHT (abstracting from physical nodes)

XML Stream Processing Requirements

- Content-based Pub-Sub requires a stream broker that can:
  - Publish xml messages from data file
  - Subscribe to xml stream over the network/file IO
  - Provide simple query semantics
  - Support concurrent subscribers
  - Provide fair queuing for all subscribers
  - Reload query transaction on the fly
  - Create a content distribution mesh
  - Be fast!

TeleScope XML Broker

Is there anything like that out there?



Let’s look in the TeleScope! ☺

TeleScope Features

- Written in pure C (yes – it is possible ☺)
- Extremely fast real-time XML stream processing
- Content based Publish-Subscribe architecture
- Support for a large number of concurrent stream subscribers

TeleScope Features

- Efficient Continuous Query Engine
- Dynamic Transaction Altering/Reset on the fly
- Real-time query statistics support

TeleScope Features

- High-bandwidth load tolerance
- Dual mode of operation – either subscriber or publisher or both modes
- XML stream filtering and reprocessing in a nodes chain (content distribution mesh via the overlay network creation)
- Loading of XML data from the data file for publishing

TeleScope Features

- Thread Pool subscribers management mechanism
- Queue based publications management
- XML processing powered by the Gnome Libxml2 library
- Remote command line access to the system via telnet

Language operators

Operator	Description	Example use
=	equality	ORIGIN = EGP
!	Not-equal	SRC_AS ! 6447
<	relational less than	MULTI_EXIT_DISC < 10
>	relational greater than	MULTI_EXIT_DISC > 100
&	Logical AND	ORIGIN = EGP & value = 0
|	Logical OR	ORIGIN = EGP | value = 1
()	Parentheses	(ORIGIN = EGP & value = 0) | (type = MESSAGE)

Language operators – CIDR Prefix match

Operator	Description	Example use
e	exact prefix match operator - matching the networks with the exactly defined network prefix range.	PREFIX e 211.64.0.0/8
l	less specific prefix match operator - matching the networks with less specific network prefix range.	PREFIX l 211.64.0.0/8
m	more specific prefix match operator - matching the networks with more specific network prefix range.	PREFIX m 211.64.0.0/8

Simple/complex expressions

Language provides the construction of simple and complex expressions to specify the filtering parameters in the XML data stream:

- ”type = UPDATE” – simple expression
- ”MULTI_EXIT_DISC = 100 & SRC PORT = 4321” – simple expression
- ”type = UPDATE | type = MESSAGE | type = KEEPALIVE” - simple expression
- ”(type = STATUS) | (type = UPDATE)” – complex expression
- ”(type = STATUS) | (type = UPDATE) | (type = KEEPALIVE)” – complex expression

Example XML Message

<XML_MESSAGE length="00001262" version="0.4" xmlns="urn:ietf:params:xml:ns:xfb-0.4" type_value="2" type="UPDATE"><BGPMON_SEQ id="2128112124" seq_num="867076586"/><TIME timestamp="1338060434" datetime="2012-05-26T19:27:14Z" precision_time="239"/><PEERING as_num_len="4"><SRC_ADDR><ADDRESS>2001:de8:6::6447:1</ADDRESS><AFI value="2">IPV6</AFI></SRC_ADDR><SRC_PORT>179</SRC_PORT><SRC_AS>6447</SRC_AS><DST_ADDR><ADDRESS>2001:de8:6::3:71:1</ADDRESS><AFI value="2">IPV6</AFI></DST_ADDR><DST_PORT>179</DST_PORT><DST_AS>30071</DST_AS><BGPID>0.0.0.0</BGPID></PEERING><ASCII_MSG length="34"><MARKER length="16">FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</MARKER><UPDATE withdrawn_len="0" path_attr_len="11"><WITHDRAWN count="0"/><PATH_ATTRIBUTES count="1"><ATTRIBUTE length="8"><FLAGS optional="TRUE"/><TYPE value="15">MP_UNREACH_NLRI</TYPE><MP_UNREACH_NLRI><AFI value="2">IPV6</AFI><SAFI value="1">UNICAST</SAFI><WITHDRAWN count="1"><PREFIX label="WITH"><ADDRESS>2404:d8::/32</ADDRESS><AFI value="2">IPV6</AFI><SAFI value="1">UNICAST</SAFI></PREFIX></WITHDRAWN></MP_UNREACH_NLRI></ATTRIBUTE></PATH_ATTRIBUTES><NLRI count="0"/></UPDATE></ASCII_MSG><OCTET_MSG><OCTETS length="34">FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00220200000000B800F08000201202404000D8</OCTETS></OCTET_MSG></XML_MESSAGE>

TeleScope Architecture

TeleScope is a complex multithreaded network application that consists of several modular parts:

- Writer Thread - (main function) – inserts XML messages into the Filtered Queue Array – executes XML parsing engine and provides network send/ receive infrastructure
- Filtered Queue Array (Queue module) for storing XML messages

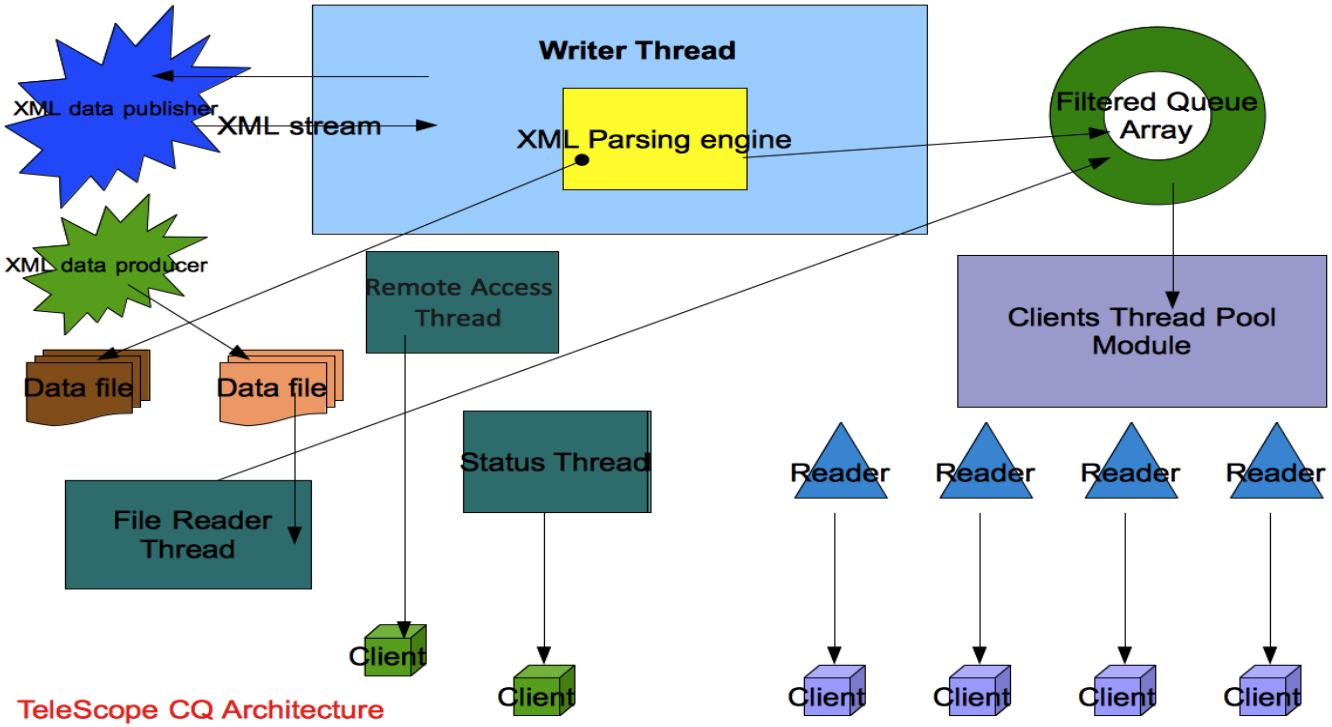
TeleScope Architecture

- Status Thread for dumping system statistics data
- Remote Access Thread – provides Command Line Interface for access to the broker across the network using password authentication
- File Reader Thread - reads XML messages from the disk file and inserts them into the Filtered Queue Array – used for publishing XML data from producers

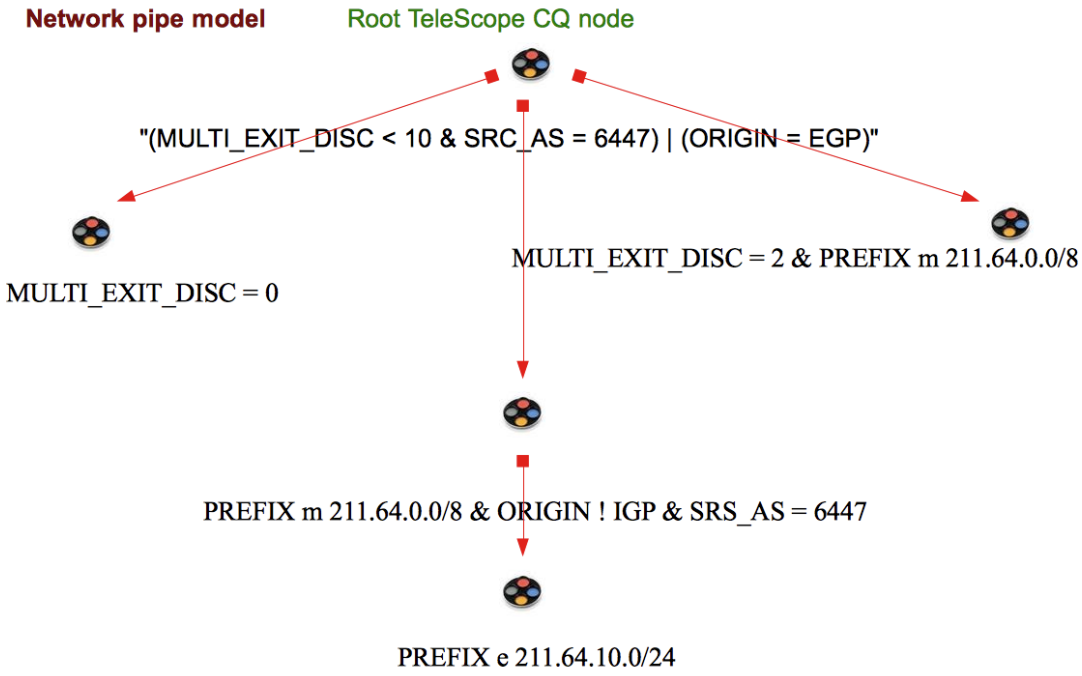
TeleScope Architecture

- XML parsing engine – implements the internal language and processing of individual XML messages coming across the network
- Clients Thread Pool – implements the Readers abstraction – allocates a separate Reader Thread per each connecting subscriber

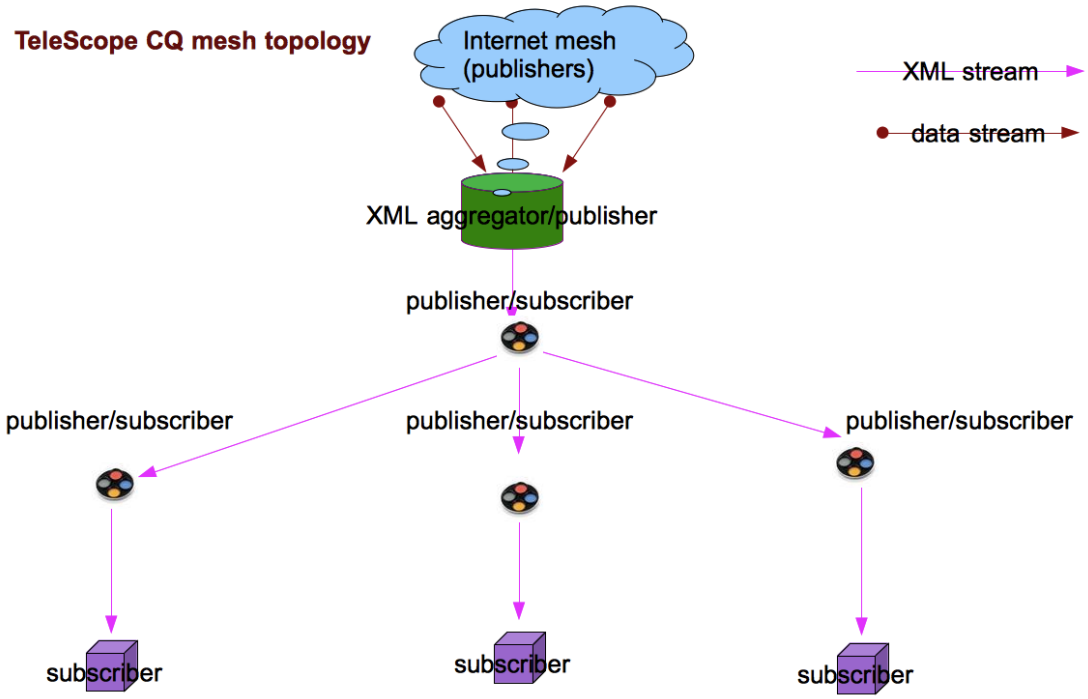
TeleScope Architecture



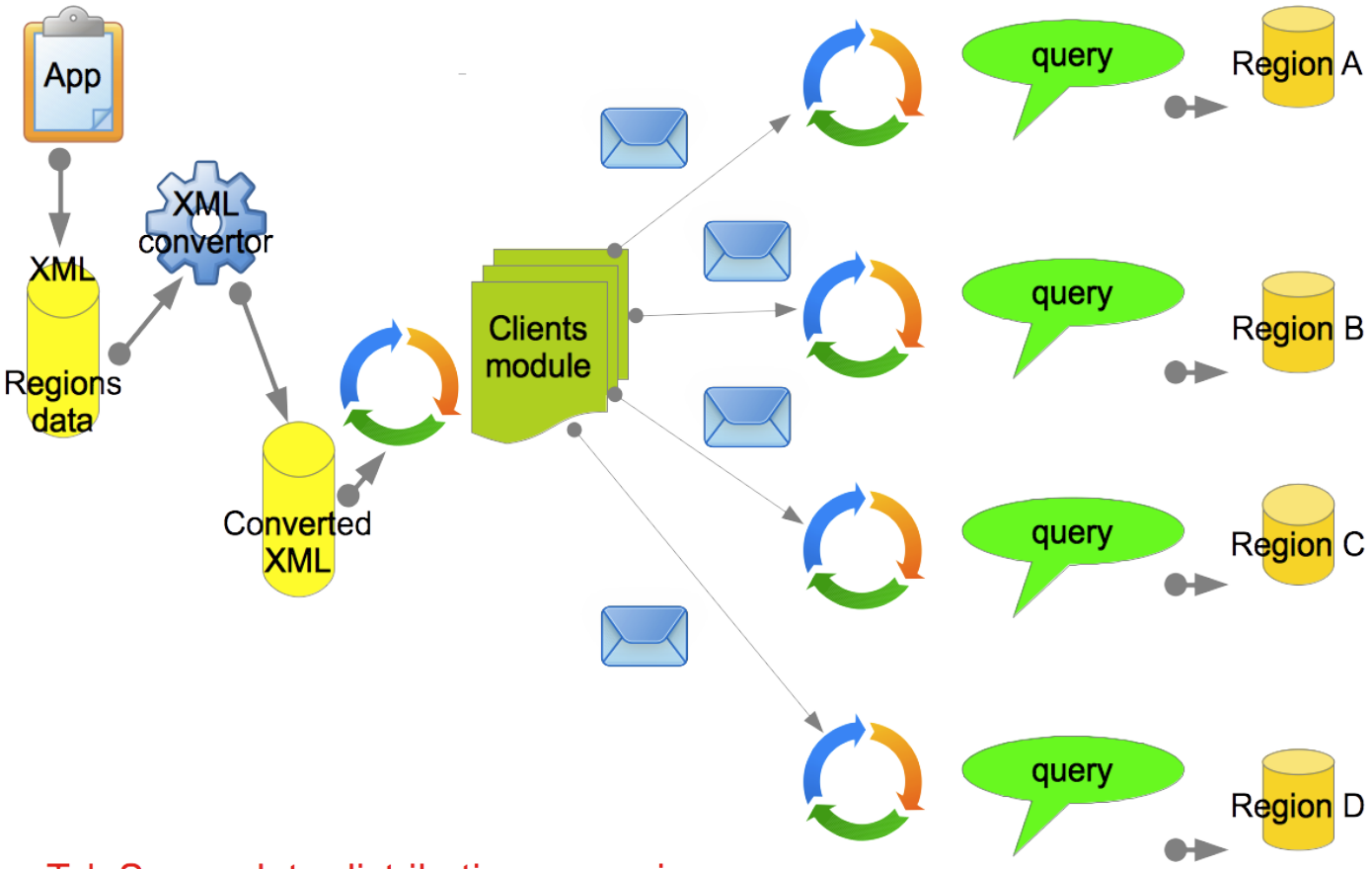
TeleScope distributed content filtering



TeleScope content mesh topology

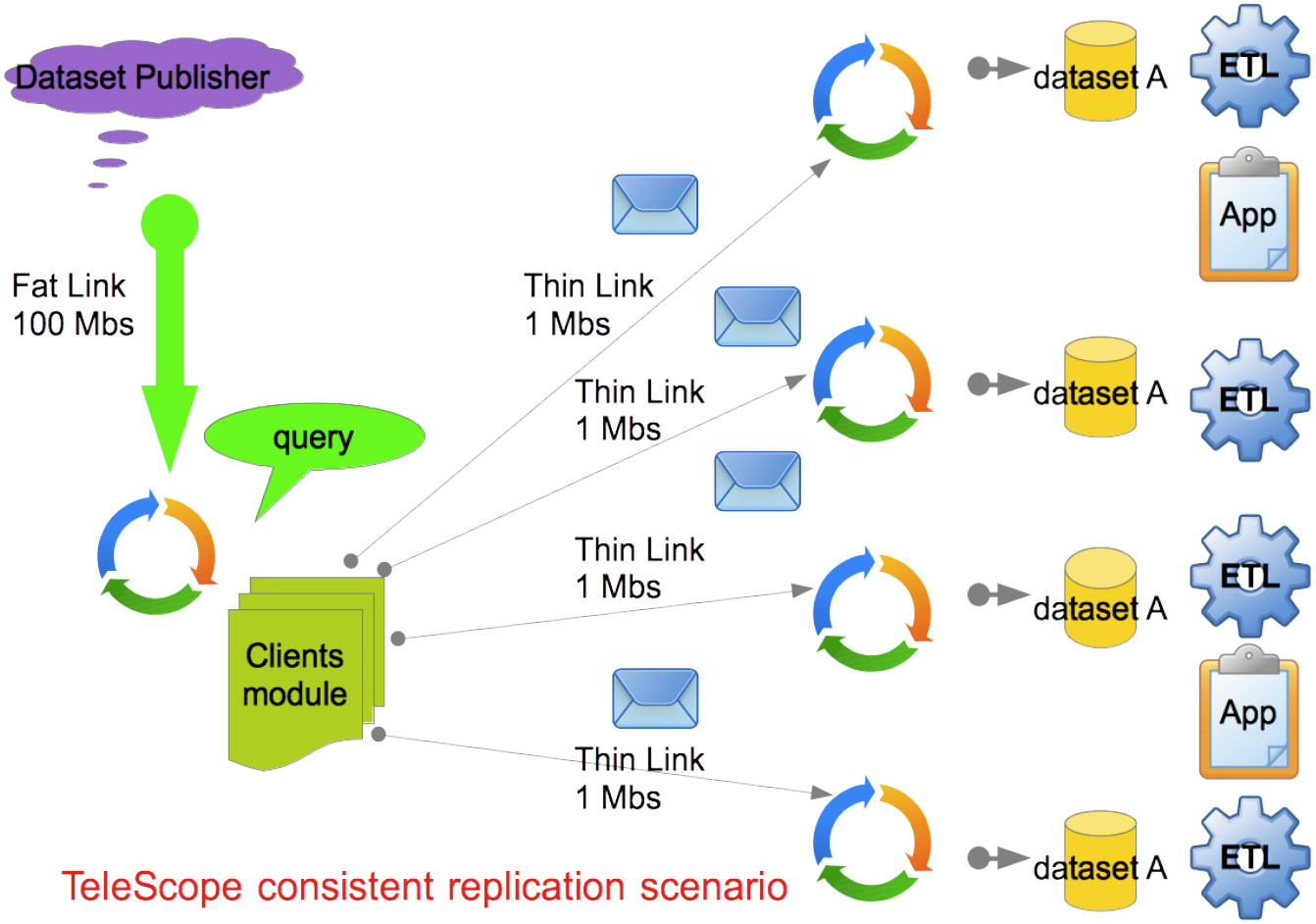


Deployment Scenarios

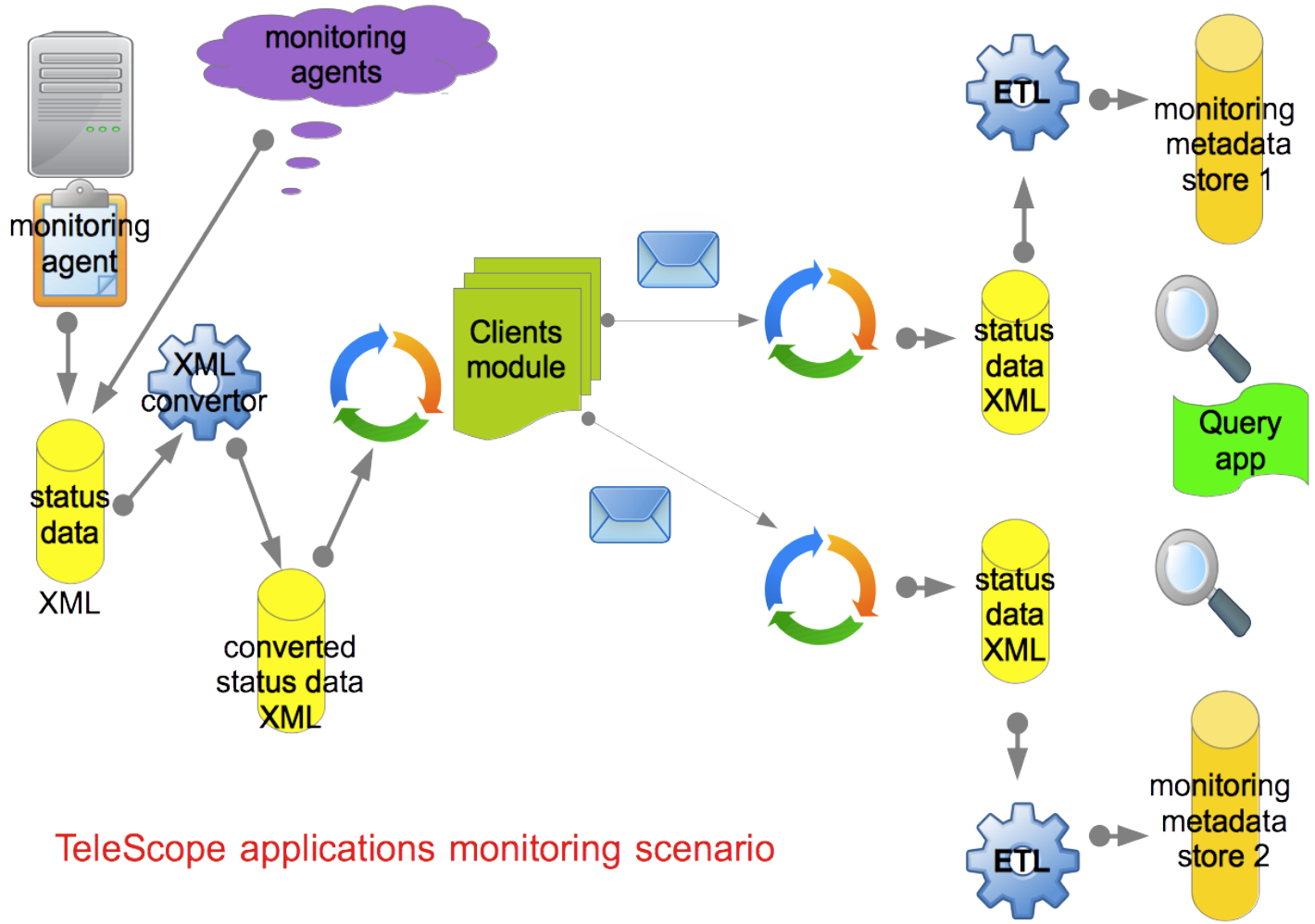


TeleScope data distribution scenario

Deployment Scenarios



Deployment Scenarios



TeleScope applications monitoring scenario

Future Directions

- Lots of things to improve and add new features as requirements mature