Site-Select Messaging for Distributed Systems

Message Addressing and Delivery via Selection Over a Distributed Relational Model of Receiver Attributes, and its Efficient Implementation on a Content-driven, Peer-to-peer, Publish/Subscribe Mechanism

Jonathan Hill
jch8f@cs.virginia.edu
Department of Computer Science
University of Virginia

Technical Report CS-2002-06

1 Introduction

The size and complexity of future distributed systems will render their management and adaptation too difficult for traditional top-down command-driven techniques. End-to-end business systems, the World Wide Web, and critical national computer systems already operate on a scale that stretches thin available management resources. More advanced management will provide greater potential at lower cost; enhancing the adaptation capabilities of existing systems. Research shows promising advances in control-driven adaptation strategies [5] [6]. The continued success of configuration programming [7] requires flexible and scalable command and control mechanism for large ad hoc systems at run time. The lack of an adequate mechanism has been a primary limiting factor in previous designs.

This paper evaluates a form of message communication we call *Site-Select Messaging. Site-Select Messaging* first appeared in the context of publish/subscribe and command of heterogeneous components in the Murdoch [2] system. Its utility for control of heterogeneous distributed systems is profound. Site-select messages enter a network with a description of the kinds of receivers to which they should be delivered. Underlying algorithms efficiently route messages to all and only destinations matching their requirements. In more detail, each receiver describes itself with a set of attributes. Messages address receivers by stating a receiver attribute model constraint. Any and all receivers obeying this constraint receive a copy of the message. Receivers are free to modify their attributes at run-time. Thus the set of receivers obtaining any message is a run-time property of the distributed system, and reflects, precisely, the currently modeled state of the distributed system.

The benefits of this messaging paradigm include

- forming a natural paradigm for the sending and receipt of commands in a large distributed system,
- a simple mapping to the selection expressions of relational algebra (from database query languages),
- efficient implementation on a content-driven publish-subscribe mechanism, and
- simple degeneration, if desired, to more traditional addressing forms such as unique names or locations.

Site-Select Messaging has many analogies in biology. Consider a system of antibodies and antigens. Each receiver wears an antigen on its surface that models the receiver’s conditions. A site-select message in transport is similar to a liquid soup containing many millions of copies of the message, where each copy of the message carries the antibody designed to bind to only a prescribed set of antigen forms. Copies of the message bind to any receiver where the antibody is sufficiently matched by the receiver antigen.

Site-Select Messaging is a powerful communications mechanism for distributed system control, as it allows transmitters to describe the kinds of sites to which commands should apply *as the address of the message*. For example, a transmitter could send a message to any site “where x is installed, three intrusion detection alarms have gone off, 2 hard drives are locally affixed, and 2 sensors have ceased responding.” In this example, attributes model conditions at a receiver site so that selection expressions define receivers by their formally represented qualities. Directive by local quality allows meaningful direction of a system without quantification or exact knowledge of widely distributed state.

This paper presents a formal model of Site-Select messaging and then reduces the model to an efficient publish/subscribe event architecture. The result is analyzed as a potential implementation strategy and upper bound on the required computational complexity and a lower bound on the expressability of Site Select Messaging. Section 2 maps

Site-Select Messaging to relational algebra. Section 3 demonstrates the effective implementation of a Site Selective Messaging on a content-driven, publish-subscribe event notification system. This section details the expressiveness and efficiency constraint of Site-Select Messaging implemented on a peer-to-peer publish/subscribe framework.

2 Site-Select Messaging, Relational Addressing Algebra

Relational algebra describes a model of relations and operations that can be utilized to capture information from defined relations. A relation is similar to a table of attributes and instances of their values. Columns of the table are labelled with the names of attributes within the data model. Each row, called a tuple, is an assignment of a value to each of the data model's attributes [1].

Both Site-Select Messaging and Publish/Subscribe Messaging are presented in relational models in this section. The models serve to compare and contrast the two messaging paradigms. In addition, the models are used in the reduction of a useful subset of Site-Select Messaging to Publish/Subscribe in Equation 3.

2.1 The Site-Select Messaging Model in Relational Algebra

There is a simple mapping of the Site-Select Messaging Paradigm to the standard relational data model and its **select** operation. Define a **Receiver Attribute Model Domain** (**RecAMD**) as a set of typed attributes with each attribute having a unique name. This is the attribute data model for receiver objects. From this define the **Receiver Relation** (**RR**) as the table formed such that each row, or tuple, is an instance of the attribute model domain as assigned at each receiver object. Let the Receiver Relation contain a key, K, giving a unique name to each receiver site under a 'name' attribute. Then receiver objects represent a distributed data structure defining the Receiver Relation. The imposition of a proper relation containing a key is irrelevant to Site-Select Messaging but simplifies our presentation in this paper.

Each message to be sent by Site-Select Messaging maintains a selection expression. A selection expression maps to a Boolean expression over the Receiver Attribute Model Domain. Hence, the selection expression is similar to the selection expression in a SELECT operation of a relational database. The following mapping exists between the two concepts: Given a Receiver Attribute Model Domain, a Receiver Relation defined by the current Attribute Model Instances at all receiver sites, and a Message with a selection expression E, then

$$S = \pi_{name}(\sigma_E(RR))$$

where **S** is the set of receiver site names of receivers to which the message will be delivered. Here, **σ** is the relational calculus selection operation (here selecting **E** over **RR**) and **π** is the project operation extracting only the name attribute from the selected receiver sites.

2.2 The Content-Driven Publish-Subscribe Model in Relational Algebra

Attribute-based content-driven publish/subscribe selects a set of messages, for a given receiver, such that the set of messages received matches the selection criteria put forth by the receiver. The selected messages are those that a receiver will receive if they are transmitted.

Define the **Message Attribute Model Domain** (**MesAMD**) as a set of typed attributes, each attribute having a unique name. This is the attribute data model for the content of messages. From this define the **Message Relation** (**MR**) as the table formed such that each row, or tuple, is a message, consisting of an instance of the attribute model. Let the Message Relation contain a key, **K**, giving a unique name to each message under a ‘msgID’ attribute.

In Publish/Subscribe each receiver site maintains a selection expression. The expression selects messages that can arrive at the site. Given a MessageAttribute Model Domain, and a Message Relation and a Receiver with Selection Expression **F**, then

$$M = \pi_{msgID}(\sigma_F(MR))$$

where M is the set of message identifiers of messages that if transmitted, would be received by the publish/subscribe receiver.

2.3 Functional View: Selection via Routing

In Site-Select Messaging and Publish/Subscribe, messages are routed to receiver sites based on their content. In the relational model, Site-Select Messaging selects receiver sites. Functionally, Site-Select Messages are delivered to any sites that they select. The same is true of publish/subscribe. Messages that are selected by a receiver are delivered to that receiver. Important to the implementation of publish/subscribe is that the routing process is the selection process. The two are not separated. Hence, routing/selection are distributed. The result is efficient message transport for publish/subscribe. In the next section, we demonstrate that a significantly expressive class Site-Select can be implemented on top of publish/subscribe, and that a useful subset of this class can be implemented efficiently.

3 Implementation over Content-Driven Publish/Subscribe

Site-Select Messaging delivers a message to all receiver sites where the *receiver's* attributes meet the criteria established by the *message*. Attribute-driven publish/subscribe systems deliver messages to all receiver sites where the *message's* attributes meets the criteria established by the *receive*r. This was presented formally in the previous section. Interestingly, Site-Select Messaging can be implemented efficiently on top of a content-driven, Publish and Subscribe architecture. A content-driven publish/subscribe architecture can be peer-to-peer and efficient [8]. Thus events can be routed efficiently for Site-Select Messages in a peer-to-peer routing mechanism. The required restrictions placed on site selection expressions are acceptable for a wide range of potential applications.

We will implement Site-Select Messaging from the following observation:

> If a receiver attribute model represents local state of the receiver, then a message attribute model can represent selection criteria over the receiver attribute model. Our goal is to construct a selection expression for publish/subscribe at each receiver, such that the instance of the publish/subscribe at the receiver selects a message if and only if the messages would select the receiver.

Mathematically, for each receiver **s**, we seek to maintain a selection function **G** in publish/subscribe to select a set of messages $\mathbf{M_s}$, so that given any message **m** with any Site-Select function $\boldsymbol{\sigma}_{\mathbf{Em}}$:

Given $S_m = \pi_{name}(\sigma_{E_m}(RR))$
Given $M_s = \pi_{msgID}(\sigma_{G_s}(MR))$
We require $m \in M_s \Leftrightarrow s \in S_m$
Which implies $E|_{MR_m}(RR_s) = G|_{RR_s}(MR_m)$

where $\mathbf{MR_m}$ represents the row of **MesAMD** attributes of the message relation for a particular message **m**. $\mathbf{RR_s}$ is the tuple of **RecAMD** attributes of receiver **s** in the receiver relation **RR**. The evaluation with a subscripted vertical bar indicates the generation of an instance of a selection expression with respect to a constants attribute set. In the case of selection expression **G**, it is generated at each receiver site **s** based on the receiver attribute model instance held at site **s**. In the case of selection expression **E**, it is generated with respect to the message attribute model instance held in message **m**. The parameters of a selection expression are the relations over which the selection expression can be evaluated. In the above implication, we must generate **E** and **G** such that the required relationship between elements of **S**, **s**, and elements of **M**, **m**, holds.

Note that the **G** utilized to generate **E** has the constraint property of requiring evaluation for the current **RecAMD** attributes at site **s**. (Again, this is represented by the bar subscripted with $\mathbf{RR_s}$.) This will require unsubscription and resubscription for the filters of receiver **s** whenever the value of $\mathbf{RR_s}$ changes.

3.1 A Simple Example

Consider the following selection expression for Site-Select Messaging carried by a message **m**:

$$E_m = (x > 0) \wedge (x < 10)$$

We can construct this within a Publish/Subscribe generic selection function by creating the attributes **xlow** and **xhigh** in **MesAMD** of the same type as **x** from **RecAMD**. Then we define G as:

$$G(MR) = xlow \le x \wedge xhigh \ge x$$

where **xlow** and **xhigh** are attributes of **MesAMD** and **x** is an attribute of **RecAMD**.

In this case $\boldsymbol{\sigma_G}$ at a site **s** is defined by:

$$\sigma_G\big|_{RR_s} = \sigma_G\big|_{x = x_s} = \sigma_{xlow \le x_s \wedge xhigh \ge x_s}$$

Site **s**, will publish/subscribe with the above filter, with $\mathbf{x_s}$ being a constant value of **x** at site **s**. If the value of **x** at **s** changes, **s** unsubscribes the above filter with its old value of **x**, and resubscribes with its new value of **x**.

Notice that the above $\boldsymbol{\sigma_G}$ defines a set of $\boldsymbol{\sigma_E}$. In this case, any $\boldsymbol{\sigma_E}$ that selects a range of values of **x** is expressible from $\boldsymbol{\sigma_G}$. When a message **m** carries Site-Select function $\boldsymbol{\sigma_{Em}}$, the selection function is represented by an instance of **MesAMD**, which in this case is an instance of **<xlow, xhigh>**. To evaluate our original example, the message would evaluate as follows via publish/subscribe through the publish/subscribe routing lattice:

$$\sigma_{E_m}(RR_s) = \sigma_G\big|_{x = x_s}(MR_m) = \sigma_G\big|_{x = x_s}([xlow = 1, xhigh = 9]) = \sigma_{(1 \le x_s) \wedge (9 \ge x_s)}$$

where many values of **x** from throughout the receiver sites of the network would direct routing, so that $\boldsymbol{\sigma_{Em}}$ is sent to any site **s** where $\mathbf{x_s}$ results in an evaluation to ***true***.

At a site where $\mathbf{x_s}$=4, in a publish/subscribe mechanisms such as Siena [8], the Site-Select Function would be represented as two selection filters:

$$\begin{gathered} xlow \le 4 \\ xhigh \ge 4 \end{gathered}$$

3.2 Expressible σ_E

Our goal is now to discuss the class of $\boldsymbol{\sigma_E}$ that are legally representable utilizing the above mathematics. The set of $\boldsymbol{\sigma_E}$ that can be constructed consists of any $\boldsymbol{\sigma_E}$ stated as a message selection from a legal $\boldsymbol{\sigma_G}$. For this discussion we will limit our $\boldsymbol{\sigma_G}$ to those legally expressible by Siena, which in turn limits its expressions by what can be efficiently calculated through peer-to-peer routing lattices. The set of legal $\boldsymbol{\sigma_G}$ generates the set of legal $\boldsymbol{\sigma_E}$.

Building Site-Select messaging on publish/subscribe requires two attribute models. To review, they are a Receiver Attribute Model Domain (**RecAMD**) and a Message Attribute Model Domain (**MesAMD**). The two models are separate domains, but are constrained by type compatibility constraints in the grammar of legal $\boldsymbol{\sigma_G}$. In Siena, a $\boldsymbol{\sigma_G}$ must be constructed so that it is a series of independent filters, such that each filter contains a list of simple Boolean comparisons between like-typed left-hand and right-hand scalar (integer, floatingpoint or string) operands. The left-hand operand must be a **MesAMD** attribute, and the right hand must be a constant value. The rules for a legal $\boldsymbol{\sigma_G}$ in Siena result in the following rules governing a generic notation legal $\boldsymbol{\sigma_G}$:

$$\begin{array}{l}
Legal\sigma_G \rightarrow \sigma_X \\
X \rightarrow Y | X \vee X \\
Y \rightarrow Z | Y \wedge Y \\
Z \rightarrow \text{symbolL COMPOP express \{type match required\}} \\
\text{symbolL} \rightarrow \text{AttributeNameFromMesAMD} \\
\text{express} \rightarrow A | \text{express ALGEBRAOP express} | (\text{express}) \\
A \rightarrow \text{AttributeNameFromRecAMD} \\
\text{COMPOP} \rightarrow < \ | \ > \ | \ <= \ | \ >= \ | \ = \ | \ != \\
\text{ALGEBRAOP} \rightarrow + \ | \ - \ | \ * \ | \ / \ | \ \%
\end{array}$$

To summarize, a selection expression $\boldsymbol{\sigma_G}$ is a Boolean expression composed such that its smallest Boolean sub-expressions each compare two integer, floatingpoint, or String subexpressions **Z**. These “smallest Boolean-typed expressions” must always maintain a left-hand subexpression of a **MesAMD** attribute. No **MesAMD** attribute may appear in the expression’s right-hand subexpression. Right-hand subexpressions of a **Z** may contain any algebraic expression where only **RecAMD** attributes appear as atomic value expressions. No **RecAMD** attribute can appear on the left-hand side of a **Z**.

Some of these grammar restrictions do not restrict expression. Restrictions in the placement of RecAMD attributes are overcome by replacing operators. For example given integers **x** and **y**, **x** < **y** is equivalent to **y** >= **x**+1. **RecAMD** attributes can also move across comparison operators. For example y+b>x is equivalent to y>x-b. Remaining, however, is the expressiveness restriction allowing one and only one **MesAMD** attribute per comparison expression.

E is any selection function generated by replacement of all Message Attribute Model Domain attributes with any legal values, in the above grammar for **G**. Hence

- expressible **E** is the set of Boolean expressions with at most one MesAMD attribute in each scalar comparison subexpression.

3.3 Complexity of Computation on σ_E

Consider any publish/subscribe selection expression **G** represented as follows:

$$G = \sum_i \prod_j \text{MessageAttribute}_{(i,j)} \phi_{(i,j)} \text{AlgebraicReceiverAttributeExpression}_{(i,j)}$$

where **MessageAttribute** is any legal Message Attribute Model Domain attribute, and **AlgebraicReceiverAttributeExpression** is a legal algebraic expression containing Receiver Attribute Model Domain attributes as its atomic elements. The **Psi** operator represents any value comparison operator resulting in a Boolean value. The **summation** and multiplication operations over **i** and **j** are index disjunction and conjunction operations, respectively. **G** is the disjunctive normal form representation of possible $\boldsymbol{\sigma_G}$ selection expressions, where assignment to the variables $\mathbf{MessageAttribute_{(i,j)}}$, $\mathbf{Psi_{(i,j)}}$, and $\mathbf{AlgebraicReceiverAttributeExpression_{(i,j)}}$ for finite sequence of **i** and **j** generates a specific $\boldsymbol{\sigma_G}$.

Unfortunately, not all **G** are equally expressible in efficient peer-to-peer Publish/Subscribe implementations. In Siena the **G** of the above notation requires **i** publish/subscribe filters of **j** filter terms. Hence, the number of publish/subscribe filters required to program σ_G is directly proportional to the number of disjunctions in **G**. Complex **E** may require **G** with many disjunctions and this will require many publish/subscribe filters at each receiver site.

Of particular complexity are any **G** containing embedded disjunctions while in conjunctive normal form. In such a **G**, the number of required publish/subscribe filters can be *exponential* in the number of disjunctions. Furthermore, the complexity of algebraic expressions on **RecAMD** attributes within a **G** will be linearly proportional to the required computation time whenever a value of a **RecAMD** attribute changes at a receiver site.

In the worst case, long algebraic expressions will require costly re-evaluation computations, and the cancellation and re-subscription of an exponential of publish/subscribe filters whenever a RecAMD attribute value changes at a receiver site.

3.4 Generating Useful G

Expressing a set of σ_E requires predetermination of the appropriate σ_G. Only expressions **E** expressible within the parameters of **G** installed at receiver sites will be available for Site-Select Command. Since σ_G must be in place at all receiver sites before Site-Select Command can be applied to those sites, we must have a means of generating useful σ_G.

This paper explores two approaches to creating useful σ_G. These techniques are useful, but not fully representative of the evident techniques. First, given a **RecAMD** we can generate a generic **MesAMD** and **G** to accommodate a large class of commonly applicable Site Selection functions. Secondly, we can designate an architectural method whereby special purpose **G** can propagate to all receivers and be instantiated as subscriptions at run-time for use by anticipated Site Selection operations. The second technique should be generally applicable where the first technique's generated σ_G is insufficient to cover a required σ_E.

3.4.1 Generic G

When each receiver site utilizes a universally known attribute model, then senders can create an equally universal message attribute model from which a general purpose **G** can be created. This **G** supports a useful and very general set of **E**. We refer to this general-purpose publish/subscribe selection function as $\mathbf{G^g}$.

Given a set **V** of integer, floatingpoint, and String attributes in an **RecAMD** we can construct a selection expression $\mathbf{G^g}$ for selection of a single closed-ranged interval and a single open-ranged interval for each $\mathbf{v_i}$ by the following procedure:

1. For each $\mathbf{v_i}$ in **V**, generate attributes $\mathbf{ClosedLowv_i}$, $\mathbf{ClosedHighv_i}$, $\mathbf{OpenLowv_i}$, and $\mathbf{OpenHighv_i}$ in **MesAMD** with the same type as $\mathbf{v_i}$.

2. Generate a subexpression in $\mathbf{G^g}$ of the form:

$$\prod_i (ClosedLowv_i \le v_i) \wedge (ClosedHighv_i \ge v_i) \wedge [(OpenHighv_i > v_i) \vee (OpenLowv_i < v_i)]$$

where the multiplication operation over **i** represents conjunction of the generated expression for each $\mathbf{v_i}$.

We can consider attributes to be members of finite set domains. In this view, we can represent an instance of an attribute as a member of a set instance if its value is 1, and not a member of the instance if its value is 0. In this notation, we can apply set comparison operations within $\mathbf{G^g}$. Operations selecting for **subset**, **superset**, **not subset**, and **not superset** can be constructed as follows:

- For each modeled set element $\mathbf{e_i}$ of set **e** in the **RecAMD** generate an $\mathbf{eSub}_i$, $\mathbf{eSup_i}$, $\mathbf{eNSub_i}$, and $\mathbf{eNSup_i}$ attribute in the **MesAMD** model to represent selection sets **eSub**, **eSup**, **eNSub**, and **eNSup**.

- Subset and Superset can be selected over **e** by:

$$\prod_i (eSub_i \le e_i) \wedge (eSup_i \ge e_i)$$

where once again the multiplication operand over **i** represents the Boolean AND operation.

- Exclusion ranges of the set with Not Subset and Not Superset operations can be generated as:

$$\sum_i (eNSub_i > e_i) \vee (eNSup_i < e_i)$$

Selection on negation of subset or superset is equivalent to open range selection for scalars. Hence, we refer to these as open-range operations.

All of the scalar range and set selection terms for $\mathbf{G^g}$ can be compounded in a single $\mathbf{G^g}$ by conjunction of the parts presented above. It should be noted, however, that the open range selector on scalars, and the exclusion range selection on set representations generate deeply embedded disjunctions in $\mathbf{G^g}$. Hence, generation of open-range selectors in $\mathbf{G^g}$ will create an exponential number of disjunctions in the number of set and scalar Receiver attributes. This will result in an exponential number of publish/subscribe filters for an applied $\boldsymbol{\sigma_G}^{\mathbf{g}}$. For efficiency concerns, open range selection and set exclusion operations will not be practical. However, automatically generated closed range selectors and closed set (subset and superset selection) set operations **are** efficient in $\boldsymbol{\sigma_G}^{\mathbf{g}}$ implemented on Siena-like algorithms, generating only a one filter with a number of conjunctive terms that is linear in the number of receiver attribute variables.

- We have shown that legal, efficiently routed selection expressions contains the class of expressions selecting a single-closed range for each receiver attribute.

3.4.2 Complexity Analysis of $\mathbf{G^g}$

We have demonstrated the construction of a $\mathbf{G^g}$ for the Siena attribute-based publish/subscribe system which allows us to perform Site-Selective Messaging. We now consider the size of the Siena filters constructed against parameters of a Receiver Site attribute model. Given that a **RecAMD** has **n** scalar attributes, and **s** disjoint sets of **m** potential members each, the generic **G** will contain $\mathbf{2^{n+sm}}$ disjunctive terms, each containing **2n+2sm** conjunctions. Implemented in Siena, this requires $\mathbf{2^{n+sm}}$ filters with **2n+2sm** constraints per filter at each receiver site. Figure 1(a) graphs the number of available attributes given a limitation in the number of filters allowed at each receiver. The exponential filter cost means that to obtain 10 attributes at each receiver, we would require more than 1,000 filters at each receiver. It is unlikely that $\mathbf{G^g}$ supporting open range operations would find practical application in an implementation of Site-Selective messaging over publish/subscribe unless the mechanism were extremely efficient for routing over large sets of filters, and subscriptions and unsubscriptions could be processed very quickly. If we restrict our selection expressions to closed ranges, then the total cost in Siena is **1** filter containing **n+sm** conjunctions. The requirement of a single filter makes the method extremely efficient. Figure 1(b) graphs the number of available attributes versus the number of available comparison operations allowed within the selection filter, assuming $\mathbf{G^g}$ only supports closed-range selection criteria.

3.4.3 Specified G

If a user anticipates Site Selectors of a given form $\boldsymbol{\sigma_E^x}$, then a specific $\boldsymbol{\sigma_G^x}$ can be created and instantiated at all receivers ahead of the Site Selection request. Recall that the requirement for generation of $\boldsymbol{\sigma_G^x}$ from $\boldsymbol{\sigma_E^x}$ is:

$$s \in \sigma_E^x(RR_s) = m \in \sigma_G^x(MR_m)$$

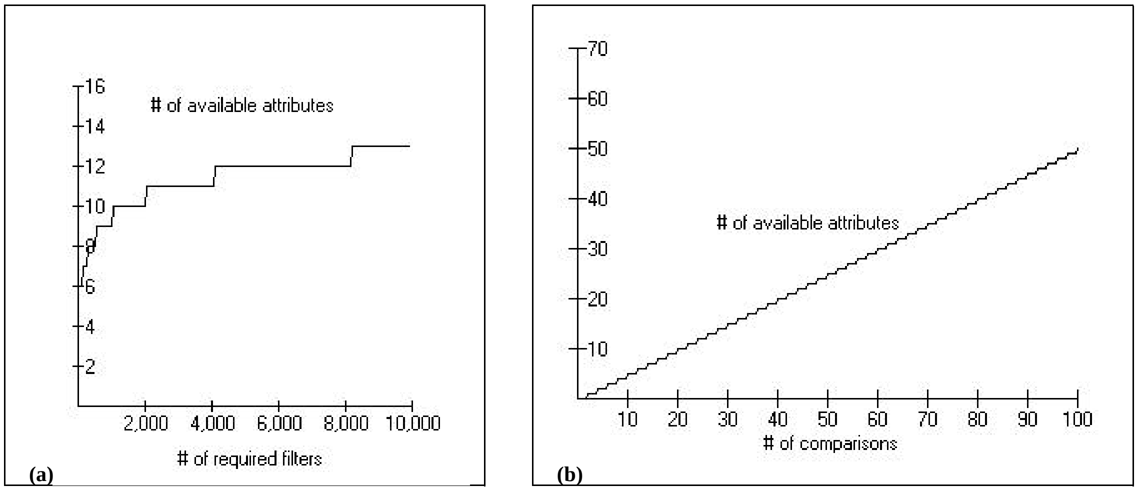

FIGURE 1. (a) a graph of the number of available attributes given a number of required filters, when open-range selection criteria are supplied by G^g. (b) a graph of the number of available attributes given a number of allowed comparison operations in the filter for a G^g with only support for closed-range selection criteria.

which implies that:

$$E^X\big|_{MR_m}(RR_s) = G^X\big|_{RR_s}(MR_m)$$

Therefore, for any Boolean expression $\mathbf{E^x}$ (obeying the restrictions of Section 3.2) to be constructed on elements of **MR** and evaluated an **RR**, an equivalently evaluating **G** can be constructed on elements of **RR** and evaluated on an **MR**. The procedure is as follows:

1. For each constant value in $\mathbf{G^x}$ generate a **MesAMD** attribute to contain that value.

2. Replace all constants of $\mathbf{E^x}$ with the generated **MesAMD** attribute. The new expression is $\mathbf{G^x}$

4 Summary

This paper demonstrated a relational model for Site-Select Messaging, a powerful technique for selecting sites to receive messages. We have demonstrated a relational algebra model for this messaging paradigm that is equivalent to the SELECT operation of traditional relational algebra for databases. We then showed how this messaging paradigm could be efficiently implemented on peer-to-peer attribute-based, content-driven publish/subscribe systems. Hence, efficient peer-to-peer routing of Site Select Messages is possible.

Furthermore, we show that the combination of a generic, $\mathbf{G^g}$, selection expression and declared specific selection expressions $\mathbf{G^x}$ can be automatically generated from receiver site attribute models, and specific Site Selection expressions, respectively. The former case covers a large class of Site Selection expressions that can be applied without declaration and broadcast of the Site-Select expression. The later allows broadcast of a Site-Select expression once, with subsequent publish/subscribe-efficient routing of Site-Select messages.

We demonstrated the class of Selection Expressions that can be legally represented on an underlying publish/subscribe mechanism. We showed the subclass that can be efficiently implemented on a publish/subscribe algorithm. We showed that the resulting legal set of selection expressions is extensive and that the efficient subset contains the class of expressions selecting a single-closed range for each receiver attribute.

For cases where we do not wish to broadcast an $\mathbf{E^x}$ prior to its use in Site-Select, or where we wish to select on an $\mathbf{E^x}$ containing inefficient disjunctions, we recommend the following approach: Site-Select for an efficient superset of the desired sites, and then locally test conditions to arrive at the desired exact set.

5 Conclusion

In general, Site Selective command is far more efficient than a broadcast-based query to identify site characteristics. As a multicast event protocol, it delivers events only to those sites that match message criteria, and does so with publish/subscribe efficiency. The combination of scalability, efficiency, and utility of the paradigm provides significant advancement for large-scale control of distributed components.

Future work might consider direct implementation of Site-Select Messaging, without an underlying publish-subscribe framework, utilizing the existing theoretical knowledge of efficient peer-to-peer publish/subscribe routing. The technique would establish a framework for the Issue/Reveal paradigm, a variant of the Publish/Subscribe concept. Our goal would be to establish its expressability efficiency based on the routing-lattice theory that supports efficient publish/subscribe. Furthermore, hierarchical implementations of Issue/Reveal might find applications in command and control hierarchies running efficient network-level multicast protocols.

6 Related Work

Brian Gerkey and Maja Mataric directly apply this technique, independently developed, in the Murdoch [2] system. Murdoch focuses on the control of Heterogeneous Agents, entirely analogous to distributed system components. In their approach, subject-based publish/subscribe makes the routing mechanism aware of the capabilities and current conditions of agents. Controllers publish commands to qualifying agents via messages with subjects qualifying the required agent attributes. The paper described attribute conjunction in the selection of agents to receive a command. It also discusses the changing of agent subscriptions to represent run-time agent conditions. The paper concludes by utilizing the above general mechanism to implement a task allocation system.

It is not surprising that this technique has been independently developed by others seeking to control heterogeneous systems with abstract commands. This is precisely the domain for which the technique is most useful, command and control based on a commonly shared attribute model. Differences between Site-Select Messaging, and the Murdoch system include valued attributes in Site-Select Messaging, value range selection, finite valued set evaluation, a formal discussion of the requirements and constraints of the relationship between publish/subscribe and Site-Select Messaging, pre-generation of generic subscription functions (which are considered implicit in Murdoch) for arbitrary attribute models, run-time modification of attribute models, run-time propagation of new efficient subscription functions for otherwise inefficient selection expressions, and an efficiency and expressiveness analysis of the former topics.

CMIS [3] and Gerel [4] incorporate the concept of selection queries for distributed resources. However, in these systems, the selection of sites is a query of resources, with the results being a list of all applicable resources, returned to the querier. From there, the querier can apply commands to the list of generated objects. Clearly, this does not scale and is not the same concept as applied in Site-Select Messaging and Murdoch.

7 Sources

[1] Elmasri, R. Navathe, S. Fundamentals of Database Systems, Second Edition. Addison-Wesley. Menlo Park, California, 1994.

[2] Gerkey, Brian P. Mataric, Maja J. ‘Murdoch.’ *Proceedings of the fourth international conference on Autonomous agents June 2000*. ACM Press.

[3] Pavlou, G. Liotta, A. Abbi, P. Ceri, S. ‘CMIS/P++: Extensions to CMIS/P for Increased Expressiveness and Efficiency in the Manipulation of Managed Information.’ IEEE Network.September/October 1999. IEEE Press.

[4] Wei, J. Endler, M. ‘Programming generic dynamic reconfigurations for distributed applications.’ *1992 International Workshop on Configurable Distributed Systems.* Page(s): 68 -79. IEEE Press.

[5] Knight, J. Heimbigner, D. Wolf, A. Carzaniga, A. Hill, J. Devanbu, P. Gertz, M. ‘The Willow Survivability Architecture.’ UVA Tech report number here

[6] Valetto, G. Kaiser, G. ‘Combining Mobile Agents and Process-based Coordination to Achieve Software Adaptation.’ (Not yet published.)

[7] Kramer, J. ‘Configuration programming- a framework for the development of distributable systems.’ *IEEE International COnference on Computer Systems and Software Engineering. Tel Aviv, Israel, May 1990.* IEEE Press.

[8] Carzaniga, A. Rosenblum, D. S. Wolf, A.L. “Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service”. *Nineteenth ACM Symposium on Principles of Distributed Computing (PODC2000), Portland OR. July, 2000.* ACM Press.