Topic-Based Resource Allocation for Real-Time Publish/Subscribe Communication Systems

(Invited Paper)

Jian Li*, XiaoQiang Ji*, Xue Liu†, JianGuo Yao‡, Sathish Gopalakrishnan§, Fei Hu*
*School of Software, Shanghai Jiao Tong University, China
†School of Computer Science, McGill University, Canada
‡School of Astronautics, Northwestern Polytechnical University, China
§Department of Electrical and Computer Engineering, University of British Columbia, Canada

***Abstract*—We address the issue of supporting real-time message dissemination in a publish/subscribe system. The asynchronous operation and connection brokering approach embodied in the publish/subscribe paradigm aid scalability and support varied topologies. These advantages, however, create challenges for providing predictable performance for real-time applications. We propose an efficient design for message queuing and forwarding by brokers in a publish/subscribe system. We differentiate messages by topics and not just be publishers or subscribers.**

Our real-time publish/subscribe design is analogous to the design of packet routers for high-speed networks. We manage input and output message queues per-client and per-topic. This approach facilitates a multiple-input multiple-output architecture and provides performance isolation for different topics. This approach differs from conventional system architectures that either use first-in first-out ordering of messages or employ per-client (not per-topic) prioritization. Our framework can provide deterministic upper bounds on delay for periodic and general (σ, ρ)-bounded real-time message flow in a publish/subscribe system with low overhead.

I. Introduction

The publish/subscribe (P/S) messaging paradigm enables the development of scalable and distributed information processing systems. For some time-critical distributed systems [1], [2], [3], [4], [5], [6], the design and implementation of a real-time publish/subscribe mechanism is a central aspect of the systems ability to provide data within the required temporal constraints. Scalability is achieved by employing brokers that connect publishers to subscribers and limit the need for many direct connections. In this article, we describe the design of a real-time message transfer scheme for P/S brokers with a view to supporting scalable real-time P/S systems. Our work supports bounded delays for messages, thereby increasing the predictability of the system.

A P/S system is comprised of publishers, subscribers and brokers (Section 2 describes the paradigm in greater detail). Publishers provide information on a *topic*; they advertise the topic and publish *notification messages* relevant to that topic. Subscribers specify their interest(s) and receive relevant information when it appears. Brokers mediate between publishers and subscribers by selecting the appropriate subscribers for each published notification. Publishers are loosely coupled to subscribers, and need not know of their existence. Publishers and subscribers focus on the topics of interest and can be ignorant of system topology. The key advantage of P/S systems is the indirect, asynchronous and anonymous communication paradigm that loosely couples the publishers and subscribers. This, however, results in unpredictable communication performance for real-time streams. In addition many publishers [subscribers] are allowed to write [read] information on the same topic. This many-to-many communication manner increases the difficulty in evaluating and measuring the performance of real-time data streams. Nevertheless, a real-time message transfer in a P/S system is needed for many applications including algorithmic trading, market data processing, fraud detection, intrusion detection, traffic surveillance and air traffic control.

Currently, the Data Distribution Service (DDS) [7] and the Java Messaging Service (JMS) [8] are commonly used application programming interfaces for P/S systems. The differences in the APIs are not critical to real-time performance; the design and implementation of the underlying system affects the quality of service. Real-Time Innovations's DDS [9] implementation maps topics to IP multicast groups so that messages can be distributed efficiently to subscribers. Clearly, multicasts use bandwidth more effectively than other schemes, which deliver the messages to different subscribers by managing a set of point-to-point connections (over TCP, HTTP or RMI). Point-to-point connections are used by, for example, openJMS [10] and ActiveMQ [11].

DDS does achieve high throughput and low latency, with claims of being 25 times faster than SUN's JMS implementation; this high performance system is, however, based on best-effort multicasting that does not scale easily to a large number of participants.

Several research efforts have focused on QoS and real-time support for P/S systems [12], [13], [14], [15], [16]. In their position paper [17], Araújo and Rodrigues proposed additional primitives for QoS-aware P/S systems. None of these approaches are known to provide guarantees for real-time message streams. Specifically, message scheduling needs greater attention to attain predictable performance for real-time streams (Section 3 elaborates on related work and the motivation for this work.).

The main contribution of this article is the presentation of a

novel real-time message scheduling and forwarding framework for the P/S system (Section 4). This framework provides a many-to-many message scheduling method for P/S system that leverages the multi-input-multi-output (MIMO) switch design strategy. In this sense, it is similar to work on switch design and scheduling for high-speed networks as exemplified by the *iSLIP* scheduler [18] that achieves 100% throughput (but has very poor deterministic performance) and switch schedulers for real-time applications [19], except that we develop mechanisms for a broker to distribute messages in a P/S system. The MIMO approach that we suggest employs input and output matrices to schedule messages and achieve real-time message dissemination for a P/S system. This method has an extremely small overhead and can provide the deterministic service delay guarantee. An interesting and useful aspect of this scheme is the use of weighted round-robin (WRR) scheduling to achieve topic-based importance so that it allocates resource to accessing clients in dynamic fashion (Section 5). We provide both theoretical bounds on messaging delay at the broker (Section 6) and evaluate the performance of the proposed scheduling scheme via the simulation on the TrueTime: a network and embedded control system simulator(Section 7).

Our approach differs from other P/S implementations as a result of these two core principles: a) we emulate a MIMO design (similar to a network switch), and b) we employ topic prioritization to achieve differentiated QoS.

II. Model for Publish/Subscribe Systems

Any P/S system (Figure 1) has three types of participants: publishers, subscribers and brokers. A P/S system implements group communication through five primitives: *subscribe, advertise, publish, unsubscribe and unadvertise* [20].

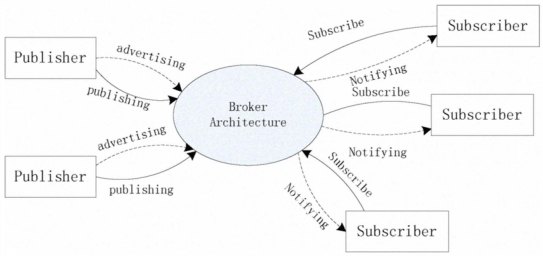

Fig. 1. Publish/Subscribe architecture

Publishers *advertise* the type of information that they would *publish*. Participants may *subscribe* to specific information types. A subscription remains in effect until it is canceled by a call to *unsubscribe*. An advertisement remains in effect until it is revoked by an unadvertise.

In the P/S model, subscribers typically receive only a subset of the total messages published by the anonymous message dissemination channel. The process of selecting messages for reception and processing is called *filtering*. There are two common forms of filtering: *topic-based* and *content-based*. In a topic-based P/S system, messages are published to *topics*. The publisher is responsible for defining the classes of messages to which subscribers can subscribe. Subscribers in a topic-based system will receive all messages published to the specific topics, and all subscribers to a topic will receive the same messages. Content-based P/S can be considered as a filtering function on top of a topic-based P/S system, that is, messages are only delivered to a subscriber if the attributes or content of those messages match the constraints defined by the subscriber. In this paper, we are solely interested in how to establish a real-time scheduling and message delivery framework for P/S systems. For the remainder of the discussion, we assume a topic-based P/S system only.

In order to simplify the problem, we target centralized P/S systems where publishers post messages to an intermediary broker and subscribers register subscriptions with that broker. The broker normally implements store-and-forward functionality to transmit messages from publishers to subscribers, and performs the filtering function.

We use the term *client* to represent the physical connection from both publishers and subscribers to the broker infrastructure. Observe that a client possesses only one physical access connection, and clients use the access points to advertise topics and subsequently to publish multiple messages on topics previously advertised. Thus, an advertisement expresses the client's intent to publish messages on a particular topic. The client also uses the access points to subscribe for notifications of interest. The broker uses the access points deliver any notifications of interest to clients.

III. Background and Motivation

Araújo and Rodrigues [17] summarized the main advantages of the Publish/Subscribe model: it decouples publishers and subscribers in several dimensions. Space decoupling captures the fact that interacting parties do not need to know each other. Time decoupling captures the fact that parties do not need to be actively participating in the interaction at the same time. Flow decoupling captures the asynchrony of the model. QoS decoupling captures the separation of QoS parameters from the type or content of events.

The asynchrony of P/S system can be implemented by means of a centralized architecture like IBM MQSeries [21] and Oracle Advanced Queuing [22]. The asynchrony can also be implemented by a distributed architecture like TIBCO Rendezvous [23] that uses a decentralized approach in which no process acts as a bottleneck or a single point of failure. Large scale infrastructure networks with either centralized or distributed brokers has been proposed in [24], [25], [20]. The real-time QoS in P/S paradigm have been studied in the past [17], [14], [16], [13], [25], [20], [26], mainly with a focus on application-level requirements or developing communication middleware and toolkits.

There is a need for real-time P/S mechanisms that leverage low-level message scheduling and forwarding methods based on message's topic importance so as to provide deterministic real-time guarantees for P/S system. In this sense, the broker in a Publish/Subscribe system, which is responsible for transferring information between publishers and subscribers, is akin

to a packet router (switch). We use this intuition in developing an efficient broker for P/S systems that can satisfy real-time requirement. We briefly mention router designs to develop the context for our work.

Packet routers (switches) can be designed to provide deterministic delays on a per-hop and end-to-end basis for real-time systems [19]. However, those results can not be employed directly in P/S system, since they only handle the message forwarding from one input to one output. Nevertheless, the abstractions of stream scheduling are germane to our real-time P/S design. Primarily, a switch has many input ports and many output ports, as well as different architectures [27], [18] and scheduling policies [28], [29], [30] have been used to transfer packets from input port to output port. This model is applicable to a broker in a P/S system because the broker has to direct messages from publishers to subscribers. Specifically, the virtual output queueing architecture [18] offers good router performance with limited overhead; in this approach, the router maintains a virtual output queue (VOQ) for each output at the input queue and uses a scheduler to transfer packets from a VOQ to an output port. This model can be re-purposed to provide fair and predictable scheduling in a P/S system.

Conventional middleware solutions for P/S systems do not take into account the resource partitioning. When a P/S service is collocated with other services, the system utilization is consistently high and delays are excessive. In our design, we assume that the P/S service is allocated a portion R of the total computational resource. As long as this allocation is protected, we are able to provide deterministic guarantees on performance.

In the next section, we describe our topic-based real-time message scheduling method for dynamic distributed real-time P/S system. This scheduler can handle messages in every input queue with different priorities and avoid the head-of-line (HOL) blocking [18]. Our analysis will show that this topic-based scheduling has desirable resource allocation properties and provides real-time guarantees.

IV. Design of a Real-Time Publish/Subscribe System

The message scheduling and forwarding policy employed by a broker is central to the design of a predictable P/S system. In a topic-based P/S system, this poses multiple challenges:

- Fair scheduling among different publishers for the same topic;
- Fair scheduling among different subscribers to the same topic;
- Appropriate prioritizing among different topics.

A P/S system needs to support a many-to-many group communication model. One topic can have many publishers and many subscribers simultaneously. We, therefore, leverage a MIMO design for our real-time P/S middleware. In our architecture, we maintain an input queue for every client (a client may host multiple publishers) and each input queue is associated with virtual output queues for each topic. This choice makes our architecture similar to the hardware design for a high-performance router.

A. The TWR^2R^2 Scheduling for Input Queue Management

1) Topic-Based Input Management: Our P/S middleware uses a Topic-based Weighted Round Robin (TWRR) message scheduling approach, which allocates the system serving time according to the topic importance. This is unlike traditional round-robin schemes that prioritize specific clients. The rationale for our choice is that it is often that topics have inherent value, and all publishers/subscribers for that topic need to be considered accordingly. The challenge posed by this prioritization is that multiple publishers and subscribers might exist for a topic and resources need to be multiplexed between these participants. Every topic is assigned a weight according to its importance, i.e., the more important a topic the greater its weight.

We use a weight matrix W to maintain the topic weights (Figure 2), where ω_k $(k = 1, \ldots, NT)$ denotes the weight of topic k and NT is the number of topics in the P/S system.

$$W = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{NT} \end{bmatrix}$$

Fig. 2. Topic weight matrix

Like most round-robin strategies, we schedule messages in time cycles. Each cycle is divided into cell times, and messages are relayed during a cell time. $TWRR$ uses the traditional WRR scheduler to accommodate every topic during each cycle (Figure 3). This ensures that each topic is served without excessive delay. (We assume the overhead of round robin operation is negligible.)

```
Supposing that there is a Topic set S = {S0, S1, ..., SNT-1};
W(Sk) indicates the weight of Si;
k indicates the server selected last time, and k is initialized with -1;
cw is the current weight in scheduling, and cw is initialized with zero;
max(S) is the maximum weight of all the topics in S;
gcd(S) is the greatest common divisor of all topic weights in S;

while (true) {
    k = (k + 1) mod NT;
    if (k == 0) {
        cw = cw - gcd(S);
        if (cw <= 0) {
            cw = max(S);
        }
    }
    if (W(Sk) >= cw)
        return Sk;
}
```

Fig. 3. Pseudocode of weighted round-robin scheduling

Suppose a broker manages the dissemination of messages belonging to three topics A, B and C, and the topics are assigned weights of 4, 3 and 2, respectively. Then, each cycle

is divided into multiple cell times according to topic weights; for example, $AABABCABC$ (Figure 5).

2) Matrix-Based Round Robin for Publishers: Each topic may have multiple publishers. The simple $TWRR$ schedule assigns cell times to topics but these cell times need to be multiplexed among the different publishers for that topic. To this end, a cell time is divided into smaller quanta. In a time quantum, the broker transfers one message from an input queue (a publisher) to multiple virtual output queues (that correspond to subscribers) using a round robin policy.

We use a topic-based input matrix ($MatrixIN$) to schedule messages from different publishers within the cell time allocated to a given topic. $MatrixIN$ is an $NC \times NT$ matrix where NC is the number of clients and NT is the number of topics. The elements of $MatrixIN = (in_{i,k})_{i=1,\ldots,NC;k=1,\ldots,NT}$ can only be 1 and 0, and if client i is maintaining a publisher on topic k then $in_{i,k} = 1$, otherwise $in_{i,k} = 0$.

An example of an input matrix is shown in Figure 4. Each row represents a client and each column represents a topic. For instance, the first row [1 0 1] means Client 1 is maintaining publishers for topics A and C.

$$MatrixIN = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Fig. 4. Topic-based input matrix

During a cell time allocated to Topic A, for example, the broker will check the input queues of clients 1, 2 and 4, and dequeue one message relevant to Topic A from every input queues of client 1, 2 and 4. This approach improves throughput considerably when compared to a traditional FIFO queue that may also experience HOL blocking.

3) Illustration of the TWR^2R^2 Scheduling Approach: The input management scheduler divides one system cycle time into cell times by $TWRR$ scheme, and divides a cell time into quanta by a simple RR scheme; we abbreviate this to TWR^2R^2 scheduling. As shown in Figure 5, a cell time is allocated to every topic, and a quantum is allocated to every input queue. To facilitate scheduling, a TWR^2R^2 broker transfer messages in fixed-size fragments that fit one quantum.

Note that we leveraged the rational design of MIMO in our P/S system design, but MIMO is able to transfer the messages in parallel based on switch hardware design, which is not realizable in P/S middleware. For this reason, we divided the cell time (T_{cell}) into quanta ($T_{quantum}$), and a quantum is allocated to each client that maitains publisher on specific topic. Accordingly, multiple messages from different clients are handled within a single cell time, and a (virtually) parallel scheme is achieved. The length of cell time (T_{cell}) depends on the number of clients. The worst-case analysis and its deterministic real-time guarantee will be discussed in the next section.

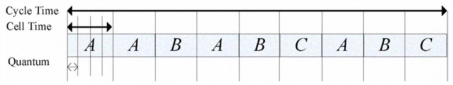

Fig. 5. Topic based TWR^2R^2 time scheduling

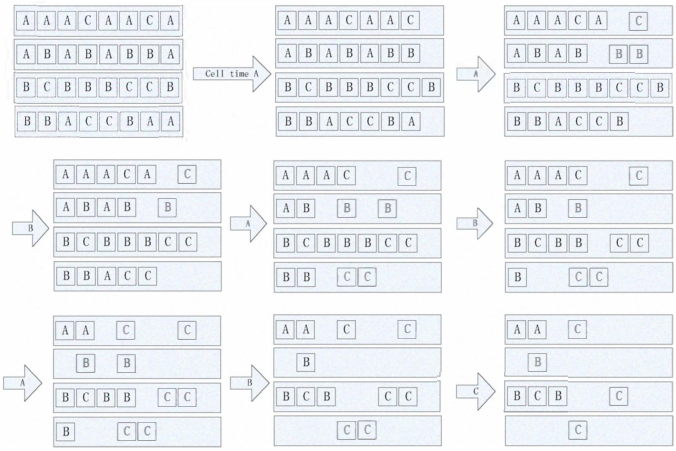

Fig. 6. Illustration of topic-based input queue scheduling routine

Figure 6 illustrates a scheduling routine under the TWR^2R^2 policy. This illustration is for the case with three topics A, B, and C with weights $W = [4\ 3\ 2]$. $MatrixIn$ is defined in Figure 4, then the cell time allocation pattern is shown in Figure 6. The first cell time is assigned to topic A, so the broker checks the input queue of clients 1, 2 and 4 because the first column of $MatrixIN$ is [1 1 0 1]. The broker dequeues and forwards the first message related to topic A in each input queue. The second cell time is also assigned to topic A, and the same procedure is followed. The third cell time is assigned to topic B, and the broker checks the input queue of clients 2, 3 and 4, since the second column of $MatrixIn$ is defined as [0 1 1 1]; The broker dequeues and forwards the first message related to topic B in the input queue of clients 2, 3 and 4. This procedure is repeated until the end of the cycle time, and another cycle starts. In the Figure 6, we assume no new messages arrive and all messages stay on their initial allocation only in order to illustrate and demonstrate the difference between TWR^2R^2 and FIFO. In a real P/S system, new messages will be enqueued, and all messages are moved toward the head of queue automatically after any dequeuing.

B. Topic-Based Output Management

The broker should deliver messages efficiently and fairly to all subscribers. Traditional P/S systems manage a group of TCP, HTTP or RMI connections to deliver the same message using point-to-point data transfer, which must significantly degrades bandwidth utilization (as mentioned in Section I). An efficient message delivery scheme should forward a message to all destinations with a single multicast or dispatch the same message through the different channels simultaneously. This kind of scheme is already employed in middleware such as RTI's DDS [9], broadcast communication over wireless

channels [31], [32], [33], in web server architectures [34], and avionics data buses [35].

In this section, we make use of a matrix $MatrixOUT$ to manage the message delivery to output ports so that the broker will forward a message, simultaneously, to all subscribers for a specific topic.

$$MatrixOUT = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Fig. 7. Topic-based output matrix

Figure 7 is an example of $MatrixOUT$; each row represents the subscription status of a client and each column represents a topic. In the example, Client 1 has subscribed to topic A and C. Meanwhile, topic A has clients 1, 3 and 4 as subscribers.

Notice that primitive actions of P/S system, such as subscribe, advertise, publish, unsubscribe and unadvertise, will cause the modification of $MatrixIN$, $MatrixOUT$ and W. Maintaining these matrices, however, imposes little overhead. In addition, the scalability is high since $MatrixIN$, $MatrixOUT$ and W are protocol independent.

V. Resource allocation based upon TWR^2R^2 scheduling

TWR^2R^2 scheduling for message dissemination provides a topic-based resource allocation. Recall that the matrix W denotes the number of cell times allocated to each topic in a cycle time. Let R denote the total available system resource, then topic k is allocated r_k portion of R, where

$$r_k = \frac{\omega_k}{\sum_{k=1}^{NT} \omega_k}. \tag{1}$$

$T_{cycle} = \sum_{k=1}^{NT} \omega_k \times T_{cell}$ represents the length of a cycle time; T_{cell} represent the maximum value of cell time (maximum value implies that all clients are publishing the specific topic). Accordingly, a topic k is allocated ω_k cell times by the broker; the total time apportioned to topic k in one cycle time is denoted by T_{topic_k}.

$$T_{topic_k} = r_k \times T_{cycle} = \omega_k \times T_{cell}. \tag{2}$$

(1) and (2) demonstrate that TWR^2R^2 message scheduler allocates an r_k-proportion of the resource to topic k, which is equal to ω_k units of cell time every cycle time.

The TWR^2R^2 scheduler allocates resources to topics, and clients share the resource in a dynamic manner. Every client obtains access to the system in accordance with a matrix Δ_{client}, which is deduced from the message scheduling management matrices introduced in the previous section.

$$\Delta_{client} = MatrixIN \times W = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{NC} \end{bmatrix}. \tag{3}$$

For every topic that a client i publishes, it is served for δ_i quanta per cycle time, as shown in (4).

$$T_{Client_i} = \theta_i \times T_{cycle} = \frac{\delta_i \times T_{quantum}}{gcd(W)}; \tag{4}$$

Client i is assigned θ_i portion of the whole system resource, as shown in (5).

$$\theta_i = \frac{\delta_i}{\sum_{i=1}^{NC} \delta_i}. \tag{5}$$

Equation (5) highlights the property that every client is not allocated a fixed fraction of the system resource; this is in contrast to traditional RR and WRR schemes. The TWR^2R^2 scheduling method allocates resources to clients based on the topics that a client publishes information for, and the weight associated with those topics. Primitive actions of P/S system will lead to a modification of $MatrixIN$ and W, which will result in a change in the resource allocated to the clients. This property makes TWR^2R^2 scheduling suitable for systems with dynamic requirements.

VI. Real-time Performance of TWR^2R^2 Scheduling

Prior results on the schedulability bound [36], [37] and on delay bounds [19] for WRR still hold for the message forwarding under TWR^2R^2 scheduling. The difference is that the TWR^2R^2 policy provides resource allocation and real-time performance guarantees for topics rather than for individual clients.

We consider the (σ, ρ)-bounded arrival flow as a general source model [38], [39]. The cumulative arrival curve is upper bounded by (σ, ρ), where ρ stands for the average arrival rate and σ stands for the burst size. If $F(t)$ denotes the arrival curve, then the arrival workload at any interval $[s, t]$ is upper bounded by:

$$F(t) - F(s) \leq \sigma + \rho(t - s) \qquad \forall 0 \leq s \leq t. \tag{6}$$

This (σ, ρ)-bounded source generates n units of workload in a time length no smaller than $l = \left(\frac{n-\sigma}{\rho}\right)^+$ [39], where the '+' symbol means the value is 0 if it is negative.

Periodic or sporadic flows can be considered as a special cases of (σ, ρ)-bounded flow with $\sigma = 1$ and $\rho = 1/P$, where P denotes the period.

For a topic k that needs to send E messages every T time units, if ω_k and T_{cycle} are given, condition (7) holds [40]:

$$\omega_k = \left\lceil \frac{E}{\lfloor T/T_{cycle} \rfloor} \right\rceil. \tag{7}$$

Condition (7) is useful only when $T \geq T_{cycle}$. It, therefore, provides a coarse delay estimate. It is known that the WRR allocates the cell time in a more regular fashion; in what follows we will present a more precise delay bound for TWR^2R^2 scheduling approach.

A. Estimating Cell Time Allocations

The matrix $W = [\omega_1;\ \omega_2; \ldots; \omega_{NT}]$ maintains the weights of topics. We assume that topics in W are in non-increasing weight order; as a result $\omega_m \geq \omega_k$ iff $m < k$. Topic k should utilize ω_k cell times in one cycle time allocated by $TWRR$. The cell time allocation pattern for topic k, in the worst case, is derived in following lemmas.

Lemma 1 (First cell time)**.** *If the current time is s and messages are scheduled using the TWR^2R^2 scheduling method (Figure 3), the next cell time will be allocated to topic k no later than*

$$t^{(1)} = s + \eta \times T_{cell}, \tag{8}$$

where

$$\eta = (NT - 1) + \frac{\sum_{m \in 1,\ldots,k-1} (\omega_m - \omega_k)}{gcd(W)}, \tag{9}$$

T_{cell} is the cell time length, NT is the number of topics, ω_k is the weight of topic k.

Proof:

According to the TWR^2R^2 method (Figure 3), one cycle time is divided into $\nu = T_{cycle}/gcd(W)$ cell times. In the worst-case, s coincides with the end of the last service time of topic k in a cycle time. We calculate the next cell time allocated to topic k. After s, TWR^2R^2 allocates $NT - k - 1$ cell times to topics from $k+1$ to NT. Then another cycle time begins.

Afterwards, the TWR^2R^2 scheduler allocates the cell times to the topics which have the largest weight first. D_k (see (10)*) is the total number of cell times devoted to topics before cw (current weight, which is a temporal variable for WRR method defined in Figure 3).*

$$D_k = \sum_{m \in 1,\ldots,k-1} (\omega_m - \omega_k)\,/gcd(W). \tag{10}$$

When cw is set to ω_k, TWR^2R^2 allocates $(k-1)$ cell times to every topic from 1 to $(k-1)$ because cw is no larger than their weight value. Then, the next cell time will be allocated to topic k. In all, from s the first cell time allocated to topic k is after $(D_k + k - 1 + NT - k)$ cell times, and the lemma is proved. ∎

Lemma 2 (Second cell time)**.** *If the current time is s and messages are scheduled using the TWR^2R^2 scheduling method (Figure 3), the second time a cell will be alloted to topic k is no later than*

$$t^{(2)} = t^{(1)} + \left(k - 1 + \sum_{g=k+1}^{NT} \Phi(\omega_g, \omega_k) \right) \times T_{cell}. \tag{11}$$

$$\Phi(\omega_g, \omega_k) = \begin{cases} 1 & \text{if} \quad \omega_g \geq \omega_k \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Proof: Note that the first cell time is allocated to topic k at time $t^{(1)}$. Afterwards, cw remains ω_k and TWR^2R^2 allocates cell time to topics g ($g = k+1, \ldots, NT$) if they have larger or equal weight value than ω_k. $\sum_{g=k+1}^{NT} \Phi(\omega_g, \omega_k)$ demonstrates this number of cell times. Then, cw is decreased by $gcd(w)$, and the TWR^2R^2 turns back to serve the topics with smaller index than k. Because the matrix W is arranged in non-increasing order of topic weights, all topics with smaller index than k will be allocated one cell time. So another $k-1$ cell time is allocated. Consequently, the sum of cell time is $\left(k - 1 + \sum_{g=k+1}^{NT} \Phi(\omega_g, \omega_k)\right)$ after $t^{(1)}$. Finally, the lemma is pursued. ∎

Lemma 3 (General cell time pattern)**.** *From any time s, and in the worst-case, the n_{th} cell time allocated to topic k is at no later than $t^{(n)}$ where $n = 2, \ldots, \omega_k/gcd(W)$,*

$$t^{(n)} = t^{(n-1)} + \Bigg(k - 1 + \sum_{g=k+1}^{NT} \Phi\Big(\omega_g, \omega_k - (n-2)gcd(W)\Big) \Bigg) \times T_{cell}. \tag{13}$$

Proof: Note that (11) is a special case of (13) with $n = 2$. After the first cell time is alloted to topic k, the TWR^2R^2 scheduler will check a round for every topic and allocate a cell time to it if its weight value is bigger than or equal to cw; cw is decreased by $gcd(W)$ after every round. The number of topics that have the weight value no smaller than cw before the n^{th} cell time alloted to topic k is $k - 1 + \sum_{g=k+1}^{NT} \Phi\Big(\omega_g, \omega_k - (n-2)gcd(W)\Big)$. This completes the lemma. ∎

We now pose an overload constraint for the publisher that is writing on a specific topic with a (σ, ρ)-bounded flow. The constraint is intuitive: the write rate of a publisher (ρ_{ik}) should not exceed the broker's service rate for that given topic.

Overload Constraint: Assume client i has a writer on topic k, which is a (σ_{ik}, ρ_{ik})-bounded flow f_{ik}. It does not create an overloaded if $\rho_{ik} < \omega_k / \sum_{k=1}^{NT} \omega_k \times R$.

Theorem 1. *Given a client i which maintains a writer on the topic k, and the writer can be modeled as a (σ, ρ)-bounded flow, the worst-case response time is bounded from above by:*

$$max_{n=1,\ldots,\omega_k/gcd(W)} \left(t^{(n)} + T_{cell} - \left(\frac{n - \sigma_{ik}}{\rho_{ik}} \right)^+ \right). \tag{14}$$

Proof: Based on the previous lemmas, $t^{(n)}$ denotes the start time of n_{th} cell time allocated to topic k. Then n^{th} message will be forwarded in the next cell time, so $t^{(n)} + T_{cell}$ is the worst-case served time for n^{th} message. $\left(\frac{n-\sigma_{ik}}{\rho_{ik}}\right)^+$ denotes the arrival time of n^{th} message from client i. The difference between the arrival time and service time is the

response time. The TWR^2R^2 scheduler allocates the cell time repeatedly in every ν, in which topic k is allocated $\omega_k/gcd(W)$ cell times. Thus the proof is complete. ■

Theorem 2. *Given a periodic flow, which can be modeled as $(1, \rho)$-bounded flow, and the arrival rate $\rho < \frac{\omega_k}{\sum_{k=1}^{NT}\omega_k} R$. Then the worst-case delay in one broker D is*

$$D = \left(NT + \frac{\sum_{m \in 1,\ldots,k-1} (\omega_m - \omega_k)}{gcd(W)} \right) \times T_{cell}. \quad (15)$$

Proof: Because the workload flow is periodic, and because its arrival rate is smaller than the service rate of the broker on that topic, the worst-case response time occurs for serving the burst. Assume the burst is loaded at time s, then first serving time of the first message is show in formula (14) when $n = 1$. Accordingly, the worst-case delay can be obtained as:

$$t^{(1)} + T_{cell} - s, \quad (16)$$

where $t^{(1)}$ is already calculated in Lemma 1. We substitute Formula (8) into Formula (16), then the theorem is proven. ■

As in Theorem 2, the per-hop upper bound on delay for a periodic flow depends on the number of topics, the topic weights, and the length of cell time (T_{cell}). The maximum value for T_{cell} is $NC \times T_{quantum}$, where NC is the number of client and $T_{quantum}$ is the maximum time for forwarding a message in the broker. $T_{quantum}$ is directly decided by the broker's computational resource portioned from the server, that is, the higher the broker's resource, the shorter the $T_{quantum}$. Normally, $T_{quantum} = B_{max}/R$, where B_{max} is the maximum message size and R is the broker's resource. Our analysis demonstrates the attributes of dynamic resource allocation as well as predictable real-time message forwarding behavior.

The analysis techniques for bounding delay can be employed to determine whether the current portioned resource for a P/S system can satisfy the application's deadline requirement. This also enables admission control for a real-time P/S system. Further, the end-to-end delay for a periodic flow can be bounded by a sum of delays along every broker in a path.

Note that the costs for worst-case delay analysis and schedulability determination are much heavier than the real-time message forwarding errand in the P/S system. In order to achieve the predictable real-time P/S performance, two methods can be employed for different cases. One method accomplishes the analysis only once which is suitable for the case where all potential subscribers are known in advance. The other method uses the resource partition (mentioned in section 3) which allocates the system resource to the schedulability determination process and P/S message forwarding process separately. The time for schedulability determination is omitted since P/S system need only the deterministic guarantee for the accepted stream after the admission control.

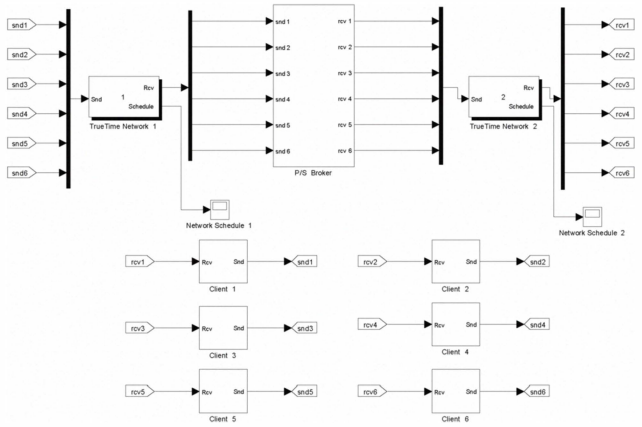

Fig. 8. Simulation RTPS in TrueTime

VII. Evaluation and Simulation

The TWR^2R^2 real-time message scheduling policy for RTPS system can provide predictable forwarding performance and allocate the system resource dynamically to the access clients. A protected system resource is allocated for message forwarding so that the deterministic performance can be guaranteed at anytime. This method differs from the traditional P/S system, which may achieve high performance with low workload but the performance turns unpredictable with high workload. In addition, we simulate TWR^2R^2 scheduling approach directly on top of different data link layers rather than aforementioned schemes in section I.

A. Simulation with TrueTime Simulator

We evaluate the RTPS performance via a real-time networked and embedded control system simulator, called TrueTime (http://www.control.lth.se/truetime/). TrueTime (TT) is event-driven simulator based on Matlab/Simulink, that makes it suitable for asynchronous p/s system simulation and provides various network modules including Ethernet, CAN, Full Duplex, TDMA, FDMA and switched ethernet networks, 802.11b WLAN and 802.15.4 ZigBee, etc.

Figure 8 illustrates the implementation of our RTPS in TT. Each client is a network node that maintains a TT kernel with the same task priority. The TrueTime network modules emulate the medium access and packet transmission from the clients to the P/S broker (via 'snd' and 'rcv' ports). TWR^2R^2 scheduling approach is implemented in P/S broker to handle packet queues, and the propagation delay is ignored. The simulation scenario is with 6 clients, 2 TT Network modules and 1 P/S Broker; the broker maintains 20 topics, and every five topics are assigned with a weight among $[10, 7, 4, 1]$; clients send messages based on subscribed topics randomly with maximum data rate of 10M bits/s and minimum frame size 512 bits; the sending time interval meets the evenly distribution between 0 to 1ms; Our P/S broker relay every message in 0.0512ms, and 0.001ms elapses for searching a specific topic in a client input queue; along the simulation

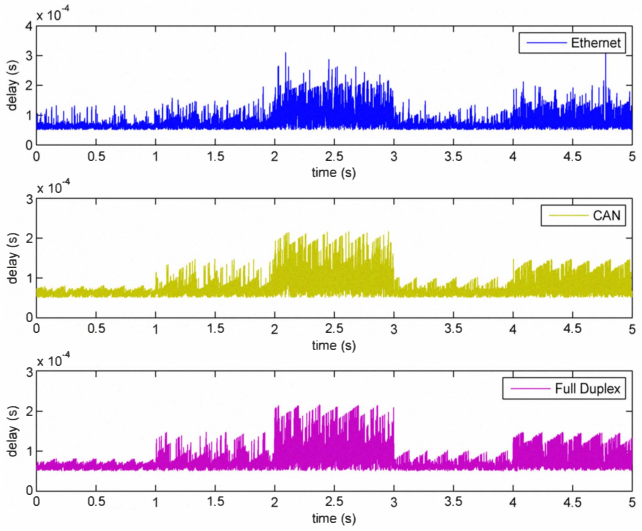

Fig. 9. RTPS performance on Three different Network Modules

time, the publishing and subscribing situation of the clients change in *one* second.

B. RTPS performance on different link layer

We first illustrate the schedule performance of TWR^2R^2 on top of different link layers, such as Ethernet(CSMA/CD), CAN(CSMA/AMP) and Switched Ethernet (Full Duplex). It is known that the real-time networking system can perform at most as well as that its underlayer transmission protocol does. See in the Figure 9, RTPS delay has the similar change trend, but the maximum delay of our RTPS on Switched Ethernet and CAN protocol (0.2160ms) is much smaller than that in Ethernet (0.3120ms). This effect is because of their predictable media access protocols [35].

C. Dynamic Resource Allocation of RTPS

We then simulate the dynamic resource allocation property of TWR^2R^2 approach according to the publishing and subscribing situation. To this end, MaxtrixIn and MatrixOUT change every second, and the transit transmission delay for every client as shown in Figure 10. Note that maximum transmission delay of the clients vary in every second according to the different allocated resource proportion. Because of the space limit, we only demonstrate the precise allocated resource, resulted average delay of Client 1 as well as its upper bound delay obtained by equation (15).

Client1	0∼1s	1∼2s	2∼3s	3∼4s	4∼5s
$\theta_1(\%)$	30.30	19.39	8.48	24.85	13.94
Avg Dly	0.0640	0.0778	0.1077	0.0711	0.0977
Max Dly	0.1320	0.1950	0.3600	0.1530	0.3120

TABLE I
ALLOCATED RESOURCE PORTION (%) AND AVERAGE, MAXIMUM DELAY OF CLIENT1 (MS)

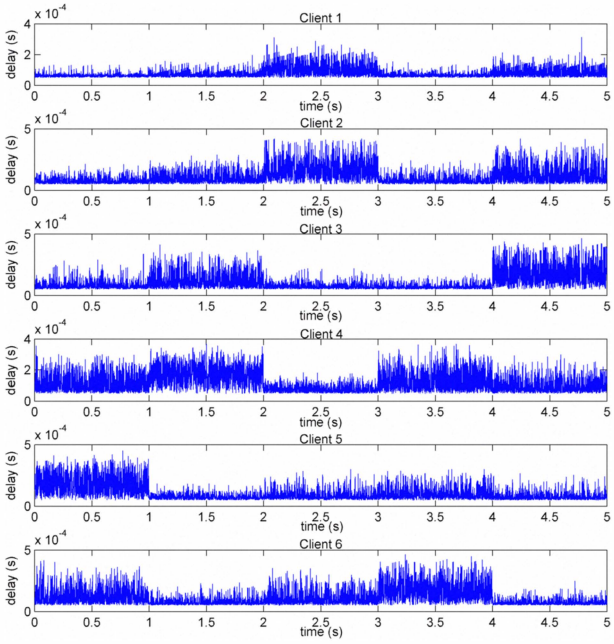

Fig. 10. Dynamic Resource Allocation by RTPS scheme

VIII. CONCLUSIONS AND FUTURE WORK

Real-time publish/subscribe systems are increasingly important in a variety of contexts. *The architecture of these systems needs considerable attention and the implementation of P/S systems needs to be coupled with the design of network infrastructure that can provide predictable data transport.* We have proposed an efficient many-to-many publish/subscribe message scheduling paradigm for asynchronous and indirect message delivery. The paradigm we suggest schedules messages with a topic-based message queuing method, and provides a predictable upper bounded delay. Specifically, we have addressed the problem in a centralized architecture where all publishers and subscribers connect to one broker. The broker is analogous to a packet router and we leverage that model in our work. In addition, our resource allocation mechanism adapts to changes in membership and workload experienced by the P/S system. Apart from extending the architecture to a decentralized setting, our future work will employ probabilistic techniques to analyze resource allocation and predict schedulability; probabilistic analysis can improve resource use and accommodate knowledge concerning system dynamics. We also note that our architecture, while guaranteeing real-time performance at the broker, does not currently consider delays incurred in the transfer of data over network connections. To ensure end-to-end performance guarantees, we have implicitly assumed networking services that provide (reasonable) upper bounds on delays, and that the entire system can then be analyzed as a multi-stage real-time system.

REFERENCES

[1] GemStone, "Case study 3 - market data," http://developer.gemstone.com/display/gfedev/Case+Studies.
[2] CISCO, "Trading floor architecture."

[3] R. Tactical Communications Group, "Rti selected for tactical communications manager product," http://www.rti.com/docs/TCG.pdf.

[4] PrismTech, "Prismtech announces next generation opensplice data distribution service."

[5] J. Yao, X. Liu, X. Chen, X. Wang, and J. Li, "Online decentralized adaptive optimal controller design of cpu utilization for distributed real-time embedded systems," *in Proceedings of the 2010 American Control Conference (ACC'10), Baltimore, MD*, 2010.

[6] J. Yao, X. Liu, and X. Zhu, "Reduced dimension control based on online recursive principal component analysis," *in Proceedings of the 2009 American Control Conference (ACC'09), St. Louis, MO*, 2009.

[7] OMG, *Data Distribution Service (DDS) Specifications.*

[8] SUN, *Java(TM) Message Service Specification Final Release 1.1.*

[9] *RTI DDS.* http://www.rti.com.

[10] *OpenJMS,*, http://openjms.sourceforge.net/index.html.

[11] Apache, *activeMQ; http://activemq.apache.org/.*

[12] F. Araújo and L. Rodrigues, ""quality of service in indirect communication systems"," *"Fourth European Research Seminar on Advances in Distributed Systems (ERSADS'01)"*, may 2001.

[13] S. Behnel, L. Fiege, and G. Muhl, ""on quality-of-service and publish-subscribe"," *icdcsw*, vol. 0, p. 20, 2006.

[14] S. S. S. T. P. Angelo CORSARO, Leonardo QUERZONI and A. VIRGILLITO, *"Quality of Service in Publish/Subscribe Middleware"*. IOS Press,, 2003.

[15] G. Deng, M. Xiong, A. Gokhale, and G. Edwards, ""evaluating real-time publish/subscribe service integration approaches in qos-enabled component middleware"," in *ISORC '07*, 2007, pp. 222–227.

[16] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "Overqos: An overlay based architecture for enhancing internet qos," March 2004.

[17] F. Araújo and L. Rodrigues, "On qos-aware publish-subscribe," in *Proceedings of the International Workshop on Distributed Event-Based Systems.* Vienna, Austria: IEEE, Jul. 2002, pp. 511–515.

[18] N. McKeown, "The islip scheduling algorithm for input-queued switches," *Networking, IEEE/ACM Transactions on*, vol. 7, no. 2, pp. 188–201, Apr 1999.

[19] Q. Wang, S. Gopalakrishnan, X. Liu, and L. Sha, ""a switch design for real-time industrial networks"," in *RTAS'08*, 2008, pp. 367–376.

[20] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[21] R. Lewis, "Advanced messaging applications with msmq and mqseries. que." 1999.

[22] Oracle, "Oracle9i application developer's guide - advanced queuing. oracle." 2002.

[23] TIBCO., "Tib/rendezvous (white paper)." 1999.

[24] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489–1499, Oct 2002.

[25] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[26] R. Rajkumar, M. Gagliardi, and L. Sha, ""the real-time publisher/subscriber inter-process communication model for distributed real-time systems: design and implementation"," in *RTAS'95*, 1995, p. 66.

[27] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 35, no. 12, pp. 1347–1356, Dec 1987.

[28] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[29] J. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," Mar. 1996, pp. 120–128.

[30] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *SIGCOMM '95*, 1995, pp. 231–242.

[31] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Tech. Jrnl*, pp. 41–59, Autumn 1996.

[32] D. Gesbert, M. Kountouris, R. W. Heath, C. byoung Chae, and T. Sälzer, "From single user to multiuser communications: Shifting the mimo paradigm," in *IEEE Sig. Proc. Magazine*, 2007.

[33] S. Cui, A. Goldsmith, and A. Bahai, "Energy-efficiency of mimo and cooperative mimo techniques in sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 6, pp. 1089–1098, Aug. 2004.

[34] N. Gandhi, D. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh, "Mimo control of an apache web server: modeling and controller design," *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, pp. 4922–4927 vol.6, 2002.

[35] "Arinc 664, aircraft data network, part 1: Systems concepts and overview," 2002.

[36] J. Wu, J.-C. Liu, and W. Zhao, ""utilization-bound based schedulability analysis of weighted round robin schedulers"," in *RTSS '07*, 2007, pp. 435–446.

[37] ——, "On schedulability bounds of static priority schedulers," *RTAS'05*, pp. 529–540, 2005.

[38] C.-S. Chang, *Performance Guarantees in Communication Networks.* Springer-Verlag, 2000.

[39] J. L. Boudec, P. Thiran, and N. Calculus, "A theory of deterministic queuing systems for the internet," 2050. [Online]. Available: citeseer.ist.psu.edu/leboudec03theory.html

[40] J. W. S. Liu, *Real Time System.* Prentice Hall, 2000.