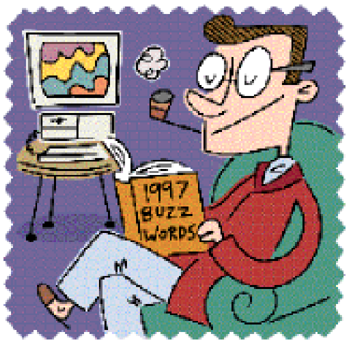
Articles & White Papers

# ON MIDDLEWARE
**By Max Dolgicer**

## With no clients and no servers, just what is publish/subscribe?




**Remember those days just a few years ago when the phrase client/server computing was nothing more than a fashionable buzzword? Well, today client/server computing is becoming more of the rule than the exception. Regardless of whether distributed client/server computing is good, bad or indifferent, the innovation contrary to other things in the world, such as technology, stocks, real estate, inflation, world affairs and so on, is something that moves in one direction only -- forward. As a result, for some, client/server has become an outdated term, as well as an outdated model.**

The moral of this example is that new distributed computing models quickly become both attractive and viable. Peer-to-peer, workflow and publish/subscribe are just a few examples of the new models that facilitate the development of new styles of distributed applications. In this column, the publish/subscribe model will be focused on for the following reasons: the publish/subscribe model is still widely misunderstood by many application developers; there are hundreds of production applications, including trading applications (speaking of mission-critical applications!), running today using this model; and lastly, we believe that the publish/subscribe model has tremendous potential for becoming a widely spread model to develop event-driven, distributed and mission-critical applications.

### Defining publish/subscribe

Today, there is a strong disagreement about the definition, interpretation and required components of the publish/subscribe model, even among people whose claim to fame is understanding this model. So, just what is the publish/subscribe model?

As the name suggests, the publish/subscribe model is publishers and subscribers. The publishers publish information and the subscribers subscribe to and consume information. There are no clients or servers. It also should be noted that the publisher can play the role of subscriber and vice versa. The roles of publishers and subscribers are determined merely by application designers, and are dependent on the nature of the application.

**Publish/subscribe development**

Historically, publish/subscribe has evolved from the real world of trading applications, and not from the research and development of universities. As the example below indicates, trading applications have a tremendous applicability to the publish/subscribe model, and this is why, even before word of the publish/subscribe model got out, the technology had been proven and tested in real commercial environments. Rendezvous from Tibco Inc. (formerly Teknekron Software Systems), Palo Alto, Calif., for example, was extracted in part from trading applications before becoming a general purpose publish/subscribe product. As a result of this applicability, the lion's share of applications running in different trading houses worldwide are using publish/subscribe technology, a substantial number of which are using Tibco's publish/subscribe trading applications.

Consider this example. An application on a trading floor publishes information pertinent to changes in stock prices. In network terms, this application simply either broadcasts or multicasts stock pricing information on the network. Broadcasting means that the information will be transmitted to every machine situated on the same network segment. Multicasting means that rather than transmitting the information to everyone, the information is transmitted only to a selected list of machines. Subscribers then subscribe to and consume that information.

Continuing the trading floor example, when a trader executes a trade, an event is generated, which in turn can trigger other applications to act upon it, thus affecting the overall state of the business. These are the events that enable individual applications to communicate with each other and to be "logically" integrated (that is, having a direct effect on each other) without being "physically" connected (that is, tightly coupled to each other).

For example, using the publish/subscribe mechanism, a global risk management application can receive realtime notification when a trading application books a trade worth $100 million without having any direct communication to it. Ultimately, the communication among publishers and subscribers is anonymous and decoupled -- consumers need not know from where or how data is being produced. The anonymous communication is very practical and efficient, no direct connection-oriented communication between different applications is required and communication overhead incurred by direct communication is prevented.

Subscribers can also subscribe to, or register interest on specific topics. For instance, in the trading floor example, there are three types of subscribers: one interested in receiving pricing information about all technology stocks; a second interested only in receiving information about IBM and Microsoft stock prices; and the third interested in the Microsoft stock price only if the value of the stock exceeds $120. (See Fig. 1.) The publish/subscribe model allows different applications to communicate with each type of subscriber without actually being connected to that subscriber.

Consider a virtual message bus (some people call this an information bus) that "wires" different corporate departments, connecting and integrating a variety of applications by allowing publishers and subscribers to communicate with each other by attaching themselves to the message bus. Imagine that a message bus "runs" across disparate lines of businesses, thus facilitating the integration of applications that were not capable of sharing information before.

When trade takes place, for example, distribution of the message (or information) pertinent to that trade to other applications, such as risk management, order matching and settlement, is essential. All other applications receive the trade information by subscribing to the information using one of the publish/subscribe APIs of the publish/subscribe product in use. For instance, if a risk management application is only interested in receiving messages about trades exceeding $100 million, it will automatically receive trades matching only this criteria. A publish/subscribe engine will not just keep track and match the publishers with the subscribers and deliver the message(s) to the subscriber(s), it will also make sure that only messages matching the subscriber’s criteria are delivered.

Fortunately, this is not a dream, it is reality. A publish/subscribe model is neither "slideware" nor "vaporware"-- it exists in the real world and provides features that no other distributed models are capable of providing. Commercial publish/subscribe middleware products on the market today include Tibco’s Rendezvous; SuiteDome from SuiteSoftware, Anaheim, Calif., which supports publish/subscribe capabilities as part of the company’s robust, object-oriented middleware product that also includes support for features such as Kerberos security, message queuing, automatic failover, a distributed object repository and a distributed database, among others; Message Distribution Systems from ISIS Distributed Systems Inc., Marlborough, Mass., which is a wholly owned subsidiary of Stratus Computer Inc., also of Marlborough, Mass.; and Sunnyvale, Calif.-based BEA Systems Inc.’s Tuxedo, a full-fledged, distributed TP monitor to which publish/subscribe capabilities have been added in the Tuxedo 6.0 Release. For those that still suffer from the common "Not Invented Here" syndrome, a publish/subscribe model, which is layered on the top of fast performing message-oriented middleware, could be developed inhouse.

**What are the pitfalls?**

The publish/subscribe model requires a paradigm shift. Designing and developing applications using publish/subscribe is nonconventional. It requires application designers and developers to come together and understand how to design event-driven applications. (An event is an occurrence that can be processed by an application. In a publish/subscribe model, an event generated by one application can be processed by another application providing that the latter registered an interest in such an event. In the risk management example above, for instance, the application will be notified automatically when a trade exceeding $100 million takes place. Subsequently, the risk management application will process such an event by alerting the company’s management about excessive trades.)

This model also assumes the presence of communication protocols (like TCP/IP) that are capable of supporting broadcasting and multicasting functionality. In reality, many corporations are still using heterogeneous communication protocols that are not capable of supporting such functionality. It is not very feasible to implement a publish/subscribe mechanism without built-in broadcast/multicast functionality. An attempt to establish point-to-point connections between the publisher and many subscribers will simply not scale.

Only a few publish/subscribe products are available on the market. Assuming that for whatever reason (for example, technology, money, politics and so on), none of the available products will fit, it becomes inevitable that corporate developers will have to consider inhouse development of a publish/subscribe model. As mentioned above, the publish/subscribe mechanism can be effectively developed on top of message-oriented middleware (MOM) systems. This is feasible; however, a few requirements are mandatory at this point: (1) the MOM product has to be extremely fast; (2) distribution of messages to process groups (one-to-many) should be included; and (3) the MOM product should understand events and be event driven.

Lastly, most of the products on the market will require custom programming to extend the publish/subscribe model to the mainframe. While this is good news to many consulting firms, it might not be as good to corporations. Since substantial amounts of corporate data reside on the mainframe, and mainframes are again playing strategic roles in corporate information systems, the integration of publish/subscribe with the mainframe is important. In fact, in our view, the mainframe should be treated in this case as just another platform, meaning that the same functionality and the same features of publish/subscribe should also be available there.

**Publish/subscribe potential**

In the publish/subscribe model, all interactions between different applications are event driven. Event-driven applications are capable of directly modeling the business. For this reason alone, we believe that event-driven applications, as opposed to standard client/server demand-driven applications, will be pervasive. This is especially true for businesses where the ability to react quickly to changes requires important information to be distributed to multiple data consumers in realtime. Publish/subscribe allows applications to publish events on the network, and other applications to be triggered and automatically invoked based on these events.

However, the publish/subscribe model is not just another method of program-to-program communication. In fact, there is no direct communication between any programs. All communication between programs is being performed via a sophisticated communication engine of publish/subscribe, which is capable of distributing and delivering information from one publisher to many subscribers without requiring each subscriber to establish an expensive network connection with the publisher.

Another advantage of the publish/subscribe model is that it is dynamic in a sense that new publishers and subscribers can be introduced without any reconfiguration. Subscribers can choose the subject matter (topics) that they are interested in at runtime. Conversely, a publisher can define a set of new subjects and immediately begin publishing information on these subjects. Again, this is accomplished by the ability of a publish/subscribe engine to keep track of the subject matters, match publishers with subscribers, as well as provide a robust communication platform capable of distributing messages across the network.

In sum, publish/subscribe is powerful middleware that provides unprecedented flexibility in the development of new distributed applications, as well as in the integration between new and existing applications. 1

Source: International Systems GroupFig. 1The publish/subscribe model allows different applications to communicate with each type of subscriber without actually being connected to that subscriber.