On Contextcast: A Context-aware Communication Mechanism

Lars Geiger, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems (IPVS), Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
phone: +49-711-7816-428 fax: +49-711-7816-424
{geiger, duerr, rothermel}@ipvs.uni-stuttgart.de

***Abstract*—The dissemination of messages according to clients' contexts (i.e., location and other attributes) opens up new possibilities in context-aware systems. While geocast or content-based publish/subscribe forward messages according to client location or attributes, respectively, neither uses a combination of the two. In this paper, we present this new communication paradigm and the challenges it poses. We also extend concepts from publish/subscribe networks to efficiently deal with highly dynamic user location to lower update rates by approximating the user's location. This reduces update rates by between 25% and 90%, depending on the granularity of the approximation.**

I. Introduction

USING user contexts to address and disseminate messages is a powerful new communication paradigm. Instead of explicitly addressing via, for instance, IP addresses, contextcast uses indirect addressing via context attributes. This enables applications like the distribution of tourist or event information. In this paper, we introduce a system for context-based addressing and routing of messages, called CONTEXTCAST. With it, an announcement for an art gallery could be sent according to a users's *interest* in modern art and the proximity to the exhibition. One can also imagine the dissemination of event information using an event's location and an *age* attribute to inform people under a certain age of a city's night life while people over a certain age might be more interested in the classical sights like churches or museums. Other sample applications of contextcast include the selective distribution of product information (advertisements), context-aware personal communication services (e.g., context-aware instant messaging or semantic email addressing [1]), or context-aware warning services, for instance, for people with allergies.

An efficient network for forwarding messages is required for large-scale contextcast scenarios. The basic idea of using attributes for addressing in contextcast is similar to content-based publish/subscribe systems, which have been thoroughly researched in the past few years. A straight-forward idea to implement contextcast, based on a publish/subscribe network, is to let users subscribe to messages based on their context and use content-based routing to forward messages to users with matching context. However, as one contribution of this paper, we show that contextcast has some fundamentally different properties, which prevent us from simply using existing content-based publish/subscribe networks for contextcast. Instead, we need novel context-based routing mechanisms. One reason is that even the very expressive class of publish/subscribe systems does not cope well with context information, in particular, location information as a very important type of context information. If a location were simply put into a subscription, any movement would cause an update of that subscription. Moreover, simply treating location as an ordinary pair of float values – as proposed by [2] – would make any two contexts differ at least in their location information. This would prevent a contextcast system from using an optimization like covering, which aims to reduce routing information in the overlay network for message distribution by aggregating similar or identical subscriptions. Such optimizations are essential for a scalable implementation; therefore, we argue that context attributes have to be treated specially to allow for efficient systems. In this paper, we present a contextcast overlay network including a first strategy to deal with highly dynamic location information by approximating user positions. Our evaluations show that this approach greatly reduces the number of context updates caused by movement of a mobile object by up to 91.5%.

The rest of this paper is structured as follows: Section II gives an overview of related work. After that, we introduce context-based routing, its semantics and our system model in Section III. Section IV shows the challenges resulting from differences to existing content-based routing mechanisms. An approach to reduce updates by approximating location information is the focus of Section V. The merit of our approach is shown in Section VI, where we present results obtained from a simulation with a prototypical implementation. In Section VII, we conclude the paper and give a short outlook on future work.

II. Related Work

The concept of a contextcast, which we introduce in this paper, is closely related to both publish/subscribe (or simply pub/sub) and geocast. From content-based pub/sub comes the usage of attribute/value pairs to address and route notifications while geocast addresses receivers in certain locations.

Numerous pub/sub systems have been researched and developed in recent years, e.g., REBECA [3] or SIENA [4]. Nevertheless, none of these systems includes any particular support to use the receiver's location in either subscriptions or notifications. The authors of [5] and [6] extended REBECA and SIENA respectively to support mobility but focused on problems arising from intermittent connectivity instead of location information for addressing. The authors in [5] also introduced "location-dependent subscriptions", which

Published in IEEE International Conference on Communications, 2009. ICC '09. Dresden, Germany, 14-18 June 2009.

http://dx.doi.org/10.1109/ICC.2009.5199239

allow a location attribute filter in subscriptions. However, they do not specify how they handle covering and merging subscriptions, which are an important aspect of REBECA. In addition, Chen et al. [2] and Cugola et al. [7] proposed to incorporate location information in publish/subscribe systems. Chen introduced a concept of spatial events but does not consider the properties of locations. Cugola also introduced locations into publish/subscribe, and argued the need to handle locations differently from simple attribute types. They recently revisited the topic in [8], where they introduced a context-aware publish/subscribe system. They extended the semantics of pub/sub to include publisher and subscriber context in addition to subscriptions and publications. The authors do not specify, however, how they intend to use this information for routing and ensure that the information is up-to-date. We consider these issues in this paper.

Existing geocast systems offer support for locations as addresses. Examples for such systems are GEO [9], which uses a geometric location model based on WGS84, [10], which uses a hierarchical symbolic location model, or "semantic geocast" [11], which uses a hybrid model. All these geocast systems use locations as the sole method of addressing and routing and do not consider additional context attributes. To extend any of these systems for context-aware communication would require integrating additional context information into the geographic routing structures, which is not a trivial task. Instead, we focus on the extension of pub/sub-like overlay networks with the capability to deal with location information.

An earlier work [12] used the term "ContextCast". However, this system only supports addressing and routing in a global context space $G \subset \mathbb{Z}^n$. Since this is an n-dimensional space, the addressing and routing can be seen as a variant of a geocast system. Additional types like *strings* or even semantic information like a *color* type and attribute are not provided and would have to be mapped to separate (numeric) dimensions in G. While this is possible for strings, it is unclear how more complex types like an enumeration *color* could be mapped onto these numeric dimensions.

III. CONTEXTCAST

In this Section, we introduce context-aware communication, the semantics of matching messages and contexts, and our general system model.

A. Context-aware Communication

1) User Contexts: The contexts in CONTEXTCAST represent users or entities participating in the system. Let k represent a particular context. We assume that it consists of a number of context attributes α, one of which is the entity's location. Since we focus on the efficient handling of a *location* attribute in such a system, we distinguish between the location attribute α_{loc} and the other context attributes α_i. The context attributes α are tuples *(type, name, value)*. The location attribute α_{loc} is given as a geometric location based on WGS84, with a type of "WGS84". For the other context attributes α_i, we currently support simple types like *integer*, *float*, and *string*. However, the system can easily be extended to support other types, e.g., an enumeration *color* or a *class* attribute based on a class hierarchy. See Figure 1 for an example of such a context.

$$
\begin{array}{rrcl}
k\text{:} & \text{WGS84: location} & = & 48.12\text{N}, 9.10\text{E} \\
& \text{string: class} & = & \text{pedestrian} \\
& \text{string: interest} & = & \text{"jazz"} \\
& \text{string: interest} & = & \text{"modern art"} \\
& \text{int: age} & = & 33 \\
& & \vdots &
\end{array}
$$

Figure 1. Example of a context k

$$
\begin{array}{rrcl}
m\text{:} & \text{WGS84: location} & \in & 48.0\text{N} - 48.4\text{N}, 9.0\text{E} - 9.2\text{E} \\
& \text{string: interest} & = & \text{"jazz"} \\
& \text{int: age} & > & 30 \\
& \text{payload} & = & [\text{concert details}]
\end{array}
$$

Figure 2. Example of a message m

2) Contextcast Messages: The contextcast messages address the contexts and thus the corresponding entities in the system. They consist of a number of constraints on context attributes. Formally, let m represent one particular message. Then m consists of a number of attribute filters ϕ. (Again, for clarity, we distinguish between the constraint on location, which we call target location ϕ_{loc}, and other attribute filters ϕ_k.) The attribute filters ϕ are tuples *(type, name, operator, value)*, where operator_ϕ can be any binary operator that is defined on the attribute type and evaluates to {true, false}. The filters determine the set of receivers by matching a message's attribute filters and the attributes of the users' contexts. The matching between messages and contexts is explained in detail in the next section. In addition to these attribute filters, a message also carries a payload. Figure 2 shows a sample message.

B. Matching Messages and Contexts

The matching of attribute filters and attributes is defined by the predicate $\text{AFMatch}(\phi, \alpha)$: An attribute filter ϕ matches an attribute α, i.e. $\text{AFMatch}(\phi, \alpha)$ evaluates to true, *iff*

$$
\begin{aligned}
\text{type}_\phi = \text{type}_\alpha \wedge \text{name}_\phi = \text{name}_\alpha \\
\wedge\, \text{operator}_\phi(\text{value}_\phi, \text{value}_\alpha)
\end{aligned}
\tag{1}
$$

Whether a message with several attribute filters matches a user context and must be delivered to that user is defined by the predicate $\text{Match}(m, k)$: Let k be a user context with attributes α_i and let m be a message with attribute filters ϕ_j. Then k matches m, i.e. $\text{Match}(m, k) = \textit{true}$, *iff*

$$
\text{for each } \phi_j \; \exists \alpha_i : \text{AFMatch}(\phi_j, \alpha_i) = \textit{true} \tag{2}
$$

Thus, attribute filters in messages are evaluated as a conjunction of predicates, while a disjunction can be expressed by sending multiple messages.

C. System Model

The purpose of our CONTEXTCAST system is to forward messages from the senders to *all* receivers with a matching context, i.e., avoid situations where a client with a matching context does not receive a message (false negatives). (Though

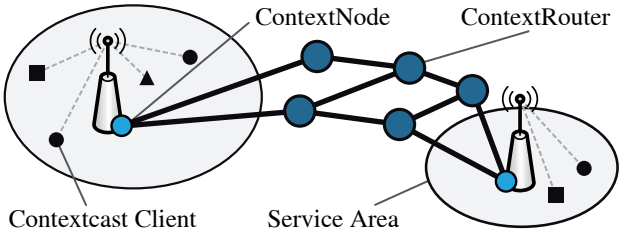

Figure 3. The CONTEXTCAST system

this seems to be most natural for applications, it may be possible to improve the system even further if we accept a certain amount of false negatives to reduce update rates. However, this is out of the scope of this paper.)

To achieve a scalable system, we use a distributed approach with routers forwarding the messages to the users with a matching context. This resembles a distributed publish/subscribe system, where a network of brokers disseminates notifications to all interested clients.

Since it is very unlikely that context-based addressing will be supported by the Internet routing infrastructure on the network layer in the near future, we imagine an overlay network for this purpose. It is formed by infrastructure nodes, which could be provided and maintained by ISPs. See Figure 3 for an overview of the overlay and its components (in addition to *location* the clients have a context attribute *interest* that is depicted in the figure by the shape of the clients).

In our system, the overlay nodes basically have two roles: routing and access. We assume the users of our system are mobile and equipped with a positioning sensor.

When the clients connect to an access node (or ContextNodes), they transmit their context information to it. The access nodes propagate this information into the network of contextcast routers (or ContextRouters), which build routing tables based on where clients with certain contexts are located. We call the propagation of context information an "update" and the contexts, which the routers store, the state information that is maintained in the overlay.

The access nodes cover a certain service area and all clients joining the system select an access node according to their own location and these service areas. The service areas can be assigned following different criteria. Since we assume a system of wired or wireless LANs containing the clients, it seems natural to assign service areas coinciding with the area covered by the LAN, i.e., a context node's service area might be as small as a floor of a building covered by a wireless LAN or as big as a set of LANs, e.g., a campus network of a university.

Users can also roam between different service areas and, after roaming, re-attach to a new access node. This information is again propagated in the network and the routers adjust their forwarding tables.

IV. COMPARISON OF CONTEXTCAST AND PUB/SUB

Pub/sub and contextcast, as we describe it here, share some similarities. Therefore, using content-based pub/sub overlay networks as a basis for distributing contextcast messages seems reasonable. However, as we show in this section, contextcast poses a number of new challenges that prevent us from simply using pub/sub.

At first glance, contextcast and pub/sub seem similar: contexts appear to be equal to subscriptions and contextcast messages equal to notifications. However, the semantics of both systems differ: in contextcast messages select the receivers via attribute filters, while in a pub/sub system the receivers select the messages via their subscription.

As we can see from this description, there is a fundamental difference between the two paradigms: In content-based publish/subscribe systems, brokers check whether the content of the published message is subsumed by the subscriptions and forward notifications accordingly. The receivers' subscriptions therefore select the interesting notifications. In contrast, contextcast routers check whether the context of the client is subsumed by the published message to make their routing decision. In this case, the messages (and hence the senders) select the receivers. The matching semantic is therefore *reversed* between the two systems. This has an impact on several aspects of the system.

First, of course, the local matching in the nodes as part of the forwarding decision has to be adapted. The basic, unoptimized implementation for this, however, is straight-forward. Second, it also affects the reduction of state in the network. Pub/sub systems increase scalability by techniques like covering to represent several similar subscriptions by the most general one; only general information covering the more specific information has to be propagated far into the network, which reduces the induced overhead. This works very well for subscriptions with value ranges. For subscriptions with point values, covering is only possible when either both subscriptions are equal or when one is more specific than the other. Unfortunately, covering in this form offers virtually no benefit in contextcast. On the one hand, since contexts are point values, we are limited to sets of identical attributes and values when trying to cover two contexts. On the other hand, also covering contexts where all other attributes are equal is hardly possible, since usually at least the location attributes are different between two contexts. Consider for instance two WGS84 coordinates of two users. These coordinates will hardly be the same for two different users. Therefore, without special treatment, both user contexts must be propaged separately into the overlay network without the possibility for covering.

Another possibility to aggregate contexts and thus reduce state information in the network is a method similar to merging in publish/subscribe. Merging replaces a set of similar subscriptions by a single, newly generated one, which contains all the information of the individual contexts. Then this merged subscription can be propagated in the network for building routing tables.

Unfortunately, a more complex aggregation largely depends on the characteristics of the contexts and on the different attribute types. *Location*, for instance, has to be aggregated differently than enumeration types like *interest*. Also, aggregation usually causes a loss of information, which in turn increases the number of messages that are transmitted without a matching receiver. These *false positives* cause additional forwarding,

processing and storage load in the network. An aggregation of user contexts must therefore find a good trade-off between the reduction of contexts and the introduction of false positives into the system.

The need for an efficient reduction of updates and state in a contextcast system is even more evident when we consider the update rates of clients: Contextcast must be able to cope with high update rates, since, for example, modern GPS receivers provide update rates of up to 4Hz. If this resulted in the same update rate in the overlay, scalability would be severely limited. In Section V, we therefore propose an approach to approximate location information in an effort to lower the update rate caused by moving clients.

Another difference between contextcast and pub/sub concerns how contexts and subscriptions are entered into the respective systems: A pub/sub system has a pattern of "subscribe, unsubscribe, subscribe, unsubscribe, ..." and there is usually little or no correlation between any two subscriptions. Consider the example of a stock market subscription, with a filter on the stock's value rising above a certain amount. If the filter matches a notification, the stock might be sold and the subscription is no longer relevant. It could be replaced by a subscription for different stocks, by a totally different subscription, or not at all. In contrast, most attributes in a user context are rather static (like the attributes *gender* or *age*) and mostly change gradually (*age* or *location* are examples for that). This could be exploited in different ways: First, we can use an "*update*" operation to update only a small part of an existing context in the system instead of revoking the complete old context and re-installing the new version. This reduces the size of the updates. Second, we can also use a prediction function to further reduce the number of state updates in the network. Then update messages would only have to be sent when the prediction does differ from the actual context values.

In this paper, we propose a technique for reducing updates by using approximated location information in user contexts. More general methods of aggregating contexts and applying predictions to context values are part of our future work.

V. Location Approximation

To improve the scalability of the system, we strive to limit the amount of context data transferred and stored in the network. The reduction of updates requires a trade off against routing accuracy: On the one hand, if we reduce updates to zero, i.e. contexts are only stored at access nodes but not propagated into the overlay network, the only way to forward a message to all addressed receivers is to broadcast it. This causes many false positives to be forwarded because of the missing information for in-network filtering. On the other hand, if we have very accurate and up-to-date information in the network and propagate every update to every router (i.e. global knowledge at every router), the system can make very precise routing decisions without creating false positives. Unfortunately it would generate an update for every little change in context.

Location is surely one of the most dynamic context attributes. Thus, movement is the reason for many of the context updates in the system. To avoid updates, we propose an approximation of *locations* and only update contexts for larger changes of *location*. (Of course, changes in other attributes still cause updates; however, in this paper we only focus on updates caused by movement.)

As we have shown in the system model, each access node has an assigned service area. Hence, a simple yet effective solution to approximate location information is to identify an entity's location with the service area of its current access node. In this case, we can forego the context updates caused by movement, as long as an entity only moves within the service area. If it leaves the area, it needs to re-attach to a different access node anyway to inform the system that messages matching its context should now be forwarded to this access node. This reduces updates caused by movement to the entering and leaving of an access node's service area, instead of every few meters.

This change also requires us to adapt how we match messages: Since contexts no longer contain exact positions, we cannot test for the inclusion of a context in the target location any more. Instead, we need to test whether the target location intersects with a context's approximated location (i.e. the access node's service area):

$$\text{WGS84: location} \cap \langle \text{target location} \rangle ,$$

where the operator $\cap$ is defined as $\text{loc}_1 \cap \text{loc}_2 = \text{true}$, *iff* $\text{loc}_1 \cap \text{loc}_2 \neq \emptyset$.

While this optimization reduces the updates caused by movement, it increases the number of false positives: We now have to consider all contexts for forwarding in which the *location* intersects with the target location, simply because the user might be in that overlapping area. However, since the context information might need to be propagated far into the network, whereas messages are directed towards matching contexts, a context update usually causes more traffic in the network than a single message. We evaluate this trade-off between updates and messages in the next section.

VI. Evaluation

To evaluate the performance of our approach, we have implemented a prototype to determine the savings in messages of the location aggregation we described in the previous section. We simulated an overlay network of 1000 ContextRouters with 2000, 5000, and 10000 mobile clients. Since we only consider the effects of the *location* attribute, we fix the other context attributes and simulate only clients with these attributes. (This also means that we do not simulate any updates caused by changes in attributes other than *location*.)

The routers were connected according to a Waxman model [13]. Simulation time of each run was $900\,\text{s}$ and the clients followed an incrementally changed random motion. We simulated clients with a low average speed typical for pedestrians. The simulated system covered a total area of $10000\,\text{units} \times 10000\,\text{units}$, which we then divided up into 400 squares (resulting in an edge length of $500\,\text{units}$ per square). On average, 60% were then selected as access networks and each was assigned a router within its area as access node.

Each of the clients sent random messages with an average rate of one message every $18\,\text{s}$; the target location of these

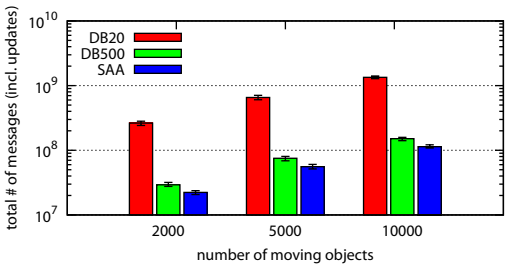

Figure 4. Message load for Contextcast

messages were random squares with an edge length of 0.5 to 1.5 times the edge length of an access network square (500 units), uniformly distributed over the total simulation area. The other attribute filters were set to the fixed attributes, therefore only the *location* was relevant for determining matching receivers.

In this system, we then simulated our approach approximating user location by the access nodes' service areas (SAA, Service Area Approximation) and compared this to an approach propagating more accurate position information, i.e. we sent a position update once the client moved more than 20 units (DB20), accounting for the typical accuracy of GPS. We also compared it to an approach with a distance threshold of 500 units (DB500). This served as a reference with a *location* uncertainty comparable to the service area approximation (if the distance threshold were significantly higher than the size of an access network, it would replicate the behavior of SAA). In all approaches the clients also sent an update message when both entering and leaving an access network.

We ran each experiment 10 times, with different network topologies, client movement, and contextcast messages. In each run, we measured the number of forwarded updates and contextcast messages between neighboring routers and calculated the average number of total overlay messages.

As can be seen from Figure 4, SAA reduces the average total number of messages by 91.5% compared to DB20. This effect is largely the result of a decrease in updates by 91.6% on average, while the number of messages increases between 66.4% for 2000 clients and 16.2% for 10000 clients because of the false positives (the amount of false positives is higher for 2000 clients because of the lower density of clients). However, since the number of forwarded updates is significantly higher than the number of forwarded messages (in the order of 10^3 for DB20 and still 10^2 for DB500), the reduction of updates greatly outweighs the false positives introduced by the approximated *location*. SAA also reduces updates by 24.9% compared to DB500, with similar location uncertainty. The reason for this reduction is the fact that even with a 500 unit threshold, the clients must still register and unregister with the access nodes, in addition to sending a *location* update once they move more than 500 units.

These results show that the reduction of update messages greatly reduces message load in the system. As the message rate increases, though, the effect of false positives also increases until finally the false positives outweigh the savings in updates. However, the simulation shows that – depending on the size of the system – the message rate would have to increase by 10^2 or more to outweigh the savings achieved by location approximation. This would mean more than 5 messages every second from each client, which is unreasonably high in the scenarios we envision for a contextcast.

VII. Conclusions

Using user contexts to address and route messages opens up interesting new applications. In this paper we identified the new challenges for contextcast, which are not addressed by existing publish/subscribe overlay networks. Moreover, we proposed a first optimized contextcast overlay network that efficiently deals with highly dynamic location information by approximating individual user positions to reduce the need for frequent position updates.

This approximation of location information also alleviates the problem that any two contexts differ at least in the *location* attribute and thus paves the way for aggregating similar context information. Since an access network might contain many objects with similar contexts, the aggregation of context information could greatly reduce the need for propagating many individual context updates into the network. Instead only few aggregated updates have to be sent. We are currently working on ways to exploit these similarities in the aggregation of context data and only propagate context for groups of users, which should further improve the scalability of routing tables and thus the whole system.

References

[1] M. Kassoff, C. Petrie, L.-M. Zen, and M. Genesereth, "Semantic email addressing: Sending email to people, not strings," in *AAAI 2006 Fall Symposium on Integrating Reasoning into Everyday Applications.*, 2006.

[2] X. Chen, Y. Chen, and F. Rao, "An efficient spatial publish/subscribe system for intelligent location-based services," in *DEBS '03: Proceedings of the 2nd international workshop on Distributed event-based systems*. New York, NY, USA: ACM, 2003, pp. 1–6.

[3] G. Mühl, "Large-Scale Content-Based Publish/Subscribe Systems," Ph.D. dissertation, TU Darmstadt, 2002.

[4] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Interfaces and Algorithms for a Wide-Area Event Notification Service," University of Colorado, Tech. Rep. CU-CS-888-99, 2000.

[5] L. Fiege, F. C. Gärtner, O. Kasten, and A. Zeidler, "Supporting Mobility in Content-Based Publish/Subscribe Middleware," in *Middleware*, ser. Lecture Notes in Computer Science, M. Endler and D. C. Schmidt, Eds., vol. 2672. Springer, 2003, pp. 103–122.

[6] M. Caporuscio, P. Inverardi, and P. Pelliccione, "Formal Analysis of Clients Mobility in the Siena Publish/Subscribe Middleware," mar 2002.

[7] G. Cugola and J. E. M. de Cote, "On introducing location awareness in publish-subscribe middleware," in *ICDCSW '05: Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 377–382.

[8] G. Cugola and M. Migliavacca, "On context-aware publish-subscribe," Fast abstract on DEBS'08, 2008.

[9] T. Imielinski and J. C. Navas, "GPS-based geographic addressing, routing, and resource discovery," *Commun. ACM*, vol. 42, no. 4, pp. 86–92, 1999.

[10] F. Dürr, C. Becker, and K. Rothermel, "An Overlay Network for Forwarding Symbolically Addressed Geocast Messages," in *Proceedings of the 15th International Conference on Computer Communications and Networks (ICCCN'06)*, oct 2006, pp. 427–434.

[11] J. Roth, "Semantic Geocast Using a Self-Organizing Infrastructure," in *Innovative Internet Community Systems*, ser. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, 2003, pp. 216–228.

[12] D. Heutelbeck, "Context Spaces — Self-Structuring Distributed Networks for Contextual Messaging and Resource Discovery," *Lecture Notes in Computer Science*, vol. 2519, pp. 248–265, 2002.

[13] B. M. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 9, pp. 1617–1622, 1988.