第 XXIV 部

Integrated Distributed Environment with Overlay Network

第 24 部
Integrated Distributed Environment with Overlay Network

第 1 章 IDEON: Integrated Distributed Environment with Overlay Network

1.1 Toward Unrestrained and Imaginative Rendezvous, Location and Routing

IDEON (Integrated Distributed Environment with Overlay Network) is a working group that pursues autonomy in the designs of distributed systems.

We believe that network designs should encourage self-generation of activities which utilize resources spread over different locations (hence, *integrated distributed environment*) by allowing spontaneous creation of layers of abstract network over the IP layer (hence, *with overlay network*).

Putting more stress on autonomy changes how the three ingredients of communication are performed:

1. Rendezvous (or how to identify the peer)
   The word "rendezvous" means a prearranged meeting place. In computer communications, such a meeting place can be a name space or a space for identifiers. Rendezvous performed under autonomy may allow spontaneous naming and resolution among the participating nodes themselves.
2. Location (or how to locate the peer)
   This is to locate the node that represents the identifier. The node is typically identified by the identifier in the lower layer of the network. Autonomous location may involve identifying the closest copy of information among redundant copies spread over the network with help from participating nodes in vicinity.
3. Routing (or how to reach the peer)
   This is to traverse the topology of the network so that a message can reach the peer. Location and routing may be done in the same procedure because it may be necessary to traverse the topology of the network to locate the peer. Autonomous routing may involve creation of topology in an ad-hoc manner.

We would like to propose alternative networking designs so that each of these can be performed in unrestrained and imaginative ways.

By "unrestrained and imaginative" we mean that no restraint should be made by the network as to which object can become the target for communication, without intervention of any authorities or privileged intermediate nodes, and that new ways of communication can be developed by the creativities of the participants of the network.

Since autonomy implies that there is no authority to guarantee the truthfulness of information (or that such an authority is weak), trust becomes an important issue.

1.2 Summary of the Research

In the IDEON working group we produced four major research outputs. Youki Kadobayashi extends Kademlia algorithm, Kenji Saito works on trust management and its applications, Takaaki Ishida made a simulator for spatially sticky information distribution algorithm for RF communication, and Yusuke Doi introduced bloom filters to enable peer group management on DHT.

The research by Youki Kadobayashi contributes for heterogeneous structures of DHT networking. Youki's extension on Kademlia algorithm enables fair-sharing between nodes that have different capabilities like connection bandwidth, processor power, and storage capacity.

To accomplish high-availability and robustness at the same time, heterogeneous networking is a rational approach. With the Youki-Kademlia we can integrate various devices like refrigerator,

HDD video recorder and many more massive always-on devices into the DHT network. Integrated with greater devices like server computers, these devices may introduce a good amount of proximity and redundancy.

This research is described in detail in chapter "Achieving Heterogeneity and Fairness in Kademlia."

The research by Kenji Saito is on trust management. Kenji is trying to apply the i-WAT system to many applications including decentralized systems like Jabber (coupled with PGP identity), and web-based systems for WIDE project research community or for readers of a book.

Spontaneous contribution helps decentralized system to work correctly. And the research is a building block that could be applied by many decentralized systems like DHT, file sharing/caching, and sensor networks. The next step would be integrating between one or more DHT algorithms and barter transactions that i-WAT expects.

This research is described in detail in chapter "i-WAT: The Internet WAT System."

The research of Takaaki Ishida is unique. Takaaki's work realizes spatially sticky information between RF-transmitter devices using a very simple algorithm. A simulator is made to investigate how the algorithm works on random traffic patterns. His vision of applications includes adhoc advertisement and event notification in specific locations.

This research is described in detail in chapter "Design of Content Cruising System."

The research by Yusuke Doi enables peer groups managed on a DHT network to be used as the foundation of fine-grained node discovery. The work introduces a protocol to take intersection between sets managed on different nodes. It uses bloom filters effectively and reduces network traffic at risk of false positives.

With such an effective way to get intersection, a properly managed index of resources on DHT can be made into an effective resource discovery mechanism. For example, intersection of peer groups (resource groups of peer nodes) gathered on peer's characteristic is fine-grained set of peers that have the characteristics.

This research is described in detail in chapter "Peer Group Rendezvous with DHT."

Another work jointly done by those members is about pointer on an inter-overlay network infrastructure. The pointer notation is called Uniform Rendezvous Pointer (in short, URP). The most significant difference between regular URL/URI and URP is that URP does not just identify a resource. Instead, URP is a notation how a client should try to search among overlays and how the result would be verified and treated. Because resources on some overlays like Freenet and Gnutella is unstable and has no identity, neither just an identifier nor locator would suffice in inter-overlays. URP would be an efficient glue between an i-WAT overlay network and other collaborative overlay networks.

This research is described in detail in chapter "Uniform Rendezvous Pointer."

1.3 Open Issues

To realize the communication model IDEON-wg proposes, we still have many open issues to be solved. To foresee the next steps of our research, we describe those issues here.

1.3.1 Measurement and Visualization

Measurement is always needed to keep a system well-formed and stable. For example, active DDoS should be detected quickly after the attack begun and blocked to prevent other normal traffic to fail. Visualization is also needed to make measurement understandable by human administrators.

But, measurement is always difficult for true distributed systems. IDEON expects all the systems have enough freeness. To ensure freeness, no central management point should be exist in the system.

Two evident examples are closely related with IDEON activity. One is DHT and the other is

i-WAT. Those two are both true distributed, without authority nor checkpoint to pass.

A DHT system may have some shared seeding nodes. Like root servers of DNS, the system may mandate a node to contact some set of nodes before joining into the DHT network. With these seeding nodes, the system can take some sort of statistics and control. But this kind of checkpoint may put bottleneck on the system and most systems would take different approach. Possible approaches for DHT may be like random-sampling, automatic aggregation like aguri, and active event transmission over the overlays.

For i-WAT and other PGP-like systems, measurement is a more difficult issue. Communication model of those systems is truly ad-hoc peer-to-peer and no one other than the two communicating now can tell when and how the communication occurs. Difficulties may arise more if there is no daemonic node to process request except a user's jabber client (i-WAT case). In this case, there is no way to take measurement if the user is not on the net.

1.3.2 Multi Thin vs. Single Ultimate

We must investigate and select among the two evident architectures, multi thin (MT) overlay network and single ultimate (SU) overlay network. The goodness of MT overlay network is its extendability. For each new application and objectives, one can create its own overlay network that has tailored naming and routing system most suitable for the application.

At the same time, MT overlay introduces complexity. If there is no interconnection between each MT overlay that would be inconvenient. The resources on the overlay cannot be accessed from outside of the overlay. To overcome its restriction, one or some set of syndication mechanism would be needed.

Pros and cons of SU overlay network is just vice versa. SU overlay introduces widely-applicable naming and routing framework, thus most of application would be okay under the framework. But at the same time, a user who wants application beyond the framework needs to modify the framework or to create a new and suitable framework for the application. One outstanding example of SU overlay network is Project JXTA (see `http://www.jxta.org/`).

1.3.3 Update and Transition Process

Architectures of overlay networks should include remote update functionality. An overlay network is actually a coordination of programs on many nodes. This is totally distributed architecture thus update of the programs is not easy as update of server centric service. Moreover, transition from old network to new network would be a challenge like re-numbering of IP-network or transition from IPv4-only world to dual stacked world with IPv6.

(on later version more discussion would be here.)

1.4 Integration with (PG)^{3}A Working Group

(PG)^{3}A stands for *PGP/GnuPG Applications*, and the working group has been studying new trust mechanisms for distributed autonomous systems based on peer-to-peer authentications.

Since its activities are technically no longer bounded to usage of PGP, and the subject matter is closely related to many research topics in IDEON, the two working groups are going to be united to form a single group to pursue autonomy in distributed environments.

This section describes the past activities of (PG)^{3}A and the prospects of activities in IDEON in the future with respect to trust in distributed autonomous systems.

1.4.1 Brief History of (PG)^{3}A

(PG)^{3}A was established in October 2002. While its goal was deployment of PGP through development of PGP/GnuPG applications, its activities were focused on development of a new trust mechanism based on peer-to-peer authentication,

using a sort of currency system to facilitate exchanges.

The following experiments have been conducted by (PG)^{3}A.

1. Evaluation of a free currency system as an application of PGP (WIDE Camp March 2003)

   i-WAT was introduced in this experiment. The objective was to evaluate the Jabber-based *i*-WAT system. Only seven participants could experience actual transactions, and the important finding was that even people with sufficient knowledge of PGP found the key exchange cumbersome. That led to the introduction of OMELETS below.

2. Verification of a cooperation mechanism using a free currency (WIDE Camp September 2003)

   OMELETS (Open, Modular and Extensible LETS[1]) was introduced in this experiment. A currency system called *WIDE Hour* was implemented as a web application using OMELETS. 166 WIDE members have been using WIDE Hour, and we are beginning to learn a lot about free currency exchange through the experiences.

1.4.2 Future of IDEON and Trust

Recent interests in *distributed algorithmic mechanism design* (DAMD) shows that researchers of distributed systems are beginning to pay more attention to incentives for cooperation and fairness of sharing resources.

DAMD is a unified effort between economics (*mechanism design* part) and computer science (*distributed algorithmic* part). It is characterized by presence of strategies and payments. Participants of a distributed autonomous system behave selfishly, choosing strategies that would maximize the payments they receive. Abstraction of strategies and payments is one of the key issues of a DAMD design.

Since our interests are in autonomy, we are most interested when those strategies and payments are about barter relationships, connecting directly the parties of exchanges. The medium of such exchanges can be seen as a barter currency. One direction that IDEON can pursue is application of barter currencies to formation of distributed autonomous systems.

第 2 章 Achieving Heterogeneity and Fairness in Kademlia

Abstract

Attempts to construct overlay networks in ubiquitous networked environments will face challenges of both performance and resource heterogeneity. We describe enhancements to Kademlia, a peer-to-peer system which has provable consistency and performance in a fault-prone environment. Our enhancements can accommodate nodes with arbitrary degree of heterogeneity, while at the same time ensuring fairness, without adding much overhead. A probabilistic mechanism to check parameter integrity is introduced, which in turn is exploited to probabilistically check compliance of nodes to advertised parameters. To demonstrate the feasibility of this approach, we have implemented a prototype and then evaluated its ability to accommodate arbitrary degree of heterogeneity.

2.1 Introduction

Many DHT algorithms[171, 237, 271] assume that all participating nodes have comparable amount of resources, and that they can accommodate the same request rate. In other words, all nodes probabilistically receive uniform load if nodes are uniformly distributed across given ID space and if requested IDs are also uniformly distributed. Nodes may occasionally receive higher load as a result of designated-node election algorithm, as in *k*-bucket node selection of

1 LETS (Local Exchange Trading System) is a form of community currency.

Kademlia[171], but they are only probabilistic results. The underlying assumption of these DHT algorithms is that uniform load — in terms of bandwidth, storage, and computation — is generally acceptable.

Such an assumption becomes unacceptable, however, if one starts to consider nodes with fewer resources, e.g., PDAs and home appliances with embedded micro-nodes. Nodes may have different kinds of resources; for example, their storage may be either abundant or scarce, bandwidth may be either narrow-band or broadband, and their connectivity may be either intermittent or persistent. Furthermore, nodes may have different communication characteristics and they may also have different failure modes. In such heterogeneous environment, the underlying assumption of uniform capacity should be abandoned in favor of capacity-aware resource allocation.

Our motivation for this work is as follows. First, we are interested to address the following question: can heterogeneous nodes with different resources augment each other, in autonomous, mutually-serving overlay networks? More specifically, can we exploit characteristics of some micro-nodes in order to maintain overlay networks more reliably? For example, refrigerator can be *always-on* node of particular overlay with energy-saving, slow processors and with few amount of memory, while PDA is *intermittent* node with modest amount of storage; we are interested to see DHT algorithms that can leverage such situations.

Second, we are interested to evaluate the feasibility of extending existing DHT algorithms for heterogeneity. The virtue of existing DHT algorithms is simplicity; if the extended DHT algorithm is very complex, it becomes very difficult to communicate the idea and build further work on it. In the meantime, complex algorithm would require major modification to source code.

Third, we are interested to maintain fairness while at the same time accommodating heterogeneity. If we tolerate heterogeneity in DHT algorithms, it becomes easy to misbehave, e.g., by pretending as micro-node and still receive as much service as regular node. We are interested to explore design space where fairness can be ensured without losing simplicity and without adding much overhead. Note that we are only dealing with fairness in the sense that false advertisement of capability can be identified and greedy use of foreign node's resources can be alarmed; we do not deal with the problem of skewed distribution of request rate among similar nodes.

This report describes extensions to Kademlia that introduce *weight* as a new node attribute, thus facilitating coexistence of lightweight nodes and heavyweight nodes, while at the same time maintaining fairness by probabilistic detection of misbehaving nodes. We focus on performance heterogeneity and resource heterogeneity in this report; we do not consider heterogeneity in hardware/software architecture, assuming pervasiveness of today's general purpose processor architecture and operating system architecture.

We have chosen Kademlia as our base DHT algorithm because of three reasons. First, it offers tunable parameters, k and α, unlike other DHT algorithms. While it has not been pointed out in the original Kademlia paper, it is trivial to redefine k and α as node-local parameters. We consider to change these parameters depending on the weight of specific node.

Second, parallel RPC to α nodes (in FIND_NODE and FIND_VALUE RPCs) and k nodes (in STORE RPC) can be exploited to combine desirable properties of nearby heterogeneous nodes. For example, relatively slow, persistent node can be used to reliably maintain k-bucket, while fast and intermittent node can be used to quickly respond to FIND_NODE requests.

Third, knowledge of numerous neighbors can be exploited in order to efficiently identify misbehaving nodes without incurring much additional overhead. When the ID space consists of d bits, Kademlia maintains contact information of kd nodes in the k-bucket, whereas Chord

maintains finger table of d nodes.

Our contribution in this report is as follows. We show that Kademlia can accommodate arbitrary degree of heterogeneity by introducing *weight* as a new node attribute, by making k and α node-local parameters, and by modifying α-node election algorithm.

Furthermore, we describe extensions to maintain fairness in heterogeneous overlay so that nodes with different weight can be differentiated, while at the same time detecting unfair use of resources within the same weight. More specifically, we introduce two new RPCs, *check fairness* RPC and *check weight* RPC, in order to detect unfair escalation of k values and false advertisement of weight, respectively.

This report is organized as follows. Section 2.2 discusses several ways to introduce heterogeneity into DHT algorithms, and then describes our heterogeneity extension. Section 2.3 describes probabilistic mechanisms to assure correctness of weight and detect greedy use of foreign resources. The feasibility and effectiveness of our method are evaluated in Section 2.4. Finally, Section 2.6 concludes this report.

2.2 Heterogeneity in Kademlia

We simply deal with performance heterogeneity and resource heterogeneity by encoding them in single scalar value of *weight*. If single scalar value falls short of representing various combinations of performance heterogeneity and resource heterogeneity, one can arbitrarily add similar scalar values to represent them. Our intent in this paper is to concisely describe portions of base DHT algorithms that must be modified; expressiveness can be better addressed at the implementation level.

For the convenience of discussion, let smaller weight represent nodes with inferior performance and fewer resources, and let larger weight represent nodes with superior performance and richer resources.

One can conceive several ways to accommodate heterogeneity in ID space. Considering the binary tree representation of ID space, it can be extended either as a concatenation of multiple homogeneous trees, or as a single, uniformly heterogeneous tree. The former case can be implemented by using ID's several most-significant bits as weight; the latter case can be implemented either by using several least-significant bits as weight, or by defining a new attribute that is independently defined from ID.

We introduce a new node attribute, weight, since Kademlia does not offer explicit *join* or *move* operation. If we spare either MSBs or LSBs, a node becomes inaccessible if the node changes its weight. While we do not consider to change nodes' weight in this paper, it is desirable to leave such possibility. Also, if we spare MSBs, the uniformity of ID space will be lost. The use of LSBs will give us similar effect to the weight attribute as long as weight remains persistent.

Weight is propagated through RPCs by including sender's weight in every packet. Also, contact is extended to <node ID, IP, UDP port, weight>, whereas it has been <node ID, IP, UDP port> in the original.

The weight attribute can be used to modify FIND_NODE and FIND_VALUE RPCs as follows. In Kademlia, FIND RPCs to remote nodes return closest k nodes, and α nodes are selected from closest k nodes with consideration to round-trip time. We elaborate the selection algorithm of α nodes by considering weight in addition to round-trip time. During α iterations of node selection, random variable w is chosen from skewed distribution and one node with weight w is picked up with consideration to round-trip time, so that nodes with different weight receive FIND request at different request rate.

It should be noted that k and α are redefined as node-local parameters that are derived from weight, whereas they have been static, globally shared constant in the original Kademlia. In order to ensure fairness, the derivation table of k and α from weight must be the same across all participating nodes. In addition, k must be

explicitly communicated to peers by including k in FIND RPCs.

Lightweight node can reduce both communication and memory overhead by using smaller weight. Since k may become small, the node can maintain and receive shorter list of contacts. More capable nodes can advertise larger weight, so that they can maintain larger k-bucket for efficient lookup. Similarly, α can be controlled independently of k as long as $\alpha \leq k$.

The heterogeneity of k comes with some drawbacks, however. If an RPC responder maintains a smaller k-bucket, it will return distant nodes even if the RPC initiator maintains larger k-bucket. In order to minimize such situations, replies from nodes with larger k-bucket should be preferred; replies from nodes with smaller k-bucket should be used only in the event that other closest k nodes do not respond.

If adjacent k nodes are of smallest weight, the preferential treatment of heavyweight nodes do not work; FIND RPCs will distribute load among lightweight nodes.

2.3 Fairness in Kademlia

While *weight* can be used to accommodate nodes with orders of magnitude difference in capacity, a mere scalar value could be easily forged to limit arrival rate, while at the same time maximizing request rate. We introduce simple mechanisms to ensure fairness without adding much overhead. We attempt to ensure fairness by identifying false advertisement of weight, unacceptable degree of parallel queries, and unacceptable request rate.

False advertisement of weight can be identified as follows. Since weight of a given node should be the same across all foreign nodes, a node receiving RPC sends a *check weight* RPC to a random node it knows of, with original packet's sender and weight encoded. We can control the overhead of the check packet by generating packets on certain probability p. The recipient of *check weight* RPC checks if known weight of the node in question agrees with the original packet's weight; if those two values disagree, it signifies error to the sender.

Note that maintaining weight of known nodes is not a major overhead in any Kademlia implementation, since it keeps record of contacts for k-bucket, and it caches recently obtained contacts for subsequent FIND RPCs.

This system of probabilistically checking integrity of weight can be circumvented by misbehaving nodes, however. If a misbehaving node disconnects for a while, wait for its ID to expire from k-bucket and from cached contacts in foreign nodes, and then it joins again, its weight can be manipulated. This comes with penalty to the misbehaving node, however: it cannot use resources on the overlay for a while.

Another kind of manipulation attack would be to change its node ID and join again to the same overlay with different weight. This cannot be prevented in most DHT algorithms, since they do not require signature of nodes that persists across sessions. In fact, such signature is very difficult to obtain unless one can define a way to associate nodes with real-world entities of enough credibility. A common alternative approach would be to penalize newly joined nodes so that they cannot send requests for a while. We do not explore this issue further in this paper.

Once integrity of weight is ensured, it becomes possible to detect unacceptable degree of parallel queries that any given node is sending, since k is directly derived from weight. We consider to detect such an event without disclosing the target that node in question is seeking for. This can be achieved by introducing *check fairness* RPC, which is also probabilistically generated by recipient of FIND RPCs. Upon receipt of a FIND RPC, each recipient generates *check fairness* RPC at certain probability, then sends it to a node of distance k. Inside the *check fairness* RPC packet, (sender, target) tuple of the original FIND RPC is hashed to conceal the target. Upon receipt of the check packet, the receiving node can detect violation of k if the received hashed value matches

the hashed value of the last known (sender, target) tuple for the sender. The node replies (sender, target) tuple to the sender as a proof of possession.

There can be such circumstances that a node of distance k is included in the next iteration of FIND query for the same target. In this case, the sender of FIND RPCs will be incorrectly identified as a misbehaving node. In order to minimize such situations, the *check fairness* RPC should be sent immediately after receipt of FIND RPC. In addition, one can introduce delay before starting next iteration of FIND RPC, in order to assure phased execution of both RPCs.

The degree of parallelism can also be controlled by manipulating α, but such manipulation is difficult to detect unless α exceeds k. The realistic solution would be to keep k small so that the impact of manipulation could be minimized.

We also consider to rate-limit request depending on the weight of sending node. This can be implemented by keeping track of inter-arrival time of requests; such an extension is not a major overhead, since original Kademlia keeps track of round-trip time to each node of k-bucket.

2.4 Evaluation

We have implemented basic part of Kademlia in C, and then extended it for heterogeneity and fairness. We have found that such extension can be easily made. The original implementation consisted of 2855 lines of code, whereas the enhanced version consisted of 3512 lines of code. The additional code of 657 lines has been broken down in Table 2.1.

Next, we ran the algorithm through single workload generation program with different set of parameters, in order to evaluate their effect on both workload distribution and lookup efficiency. The workload generation program runs as an overlay node which initiates requests and responds to other node's requests. We used a test-bed environment which consisted of 4 PCs interconnected by a 100base-TX switch, each running 32 instances of the workload generation program. Each run of emulation consisted of 600 seconds; 12 runs with different random seeds are conducted for each set of parameters, in order to offset potential skew in randomness.

Table 2.1. Breakdown of additional code

Category	Lines
k and weight	39
Modified node selection	64
check weight RPC	85
check fairness RPC	102
Request rate check	51
Hash function	200
Miscellaneous	116
Total	657

We used the following workload: constant request rate, with uniformly distributed weight, and without any artificial delay or packet loss. In order to assure uniform distribution of weight, the number of nodes for each weight has been kept constant. In the α-node selection algorithm, the probability of choosing weight w, p_w, is exponentially proportional to its weight: $p_{w+1} = 2p_w$, and $\sum p_w = 1$.

Requests are generated as follows: STORE requests are generated to random target in the ID space, and both FIND_VALUE and FIND_NODE requests randomly re-used recent targets. These three kinds of requests have been sent alternatingly at every second.

We measured the number of FIND_NODE packets received, in 128-node emulations of $w < 8$ and $w < 4$, with parameter k ranging from 5 to 20. The box-and-whisker plots of $k = 5$ are presented in Fig. 2.1 and 2.2, while results for $k = 20$ are presented in Fig. 2.3 and 2.4. The α has been kept constant; $\alpha = 2$ in all emulations. We confirmed that highly skewed distribution of load among classes can be attained by increasing maximum weight from 4 to 8. The outliers of weight 0 are particularly visible in Fig. 2.1 and 2.2, since an initial node with weight 0 is designated as initial rendezvous node; it accepts initial FIND_NODE requests from the rest of the nodes.

Interestingly, the number of received FIND_NODE packets are linearly proportional

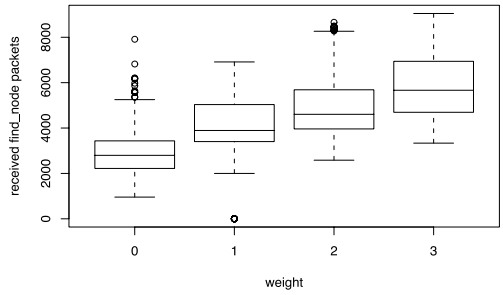

Fig. 2.1. Distribution of received FIND_NODE packets; $k = 5, w < 4$

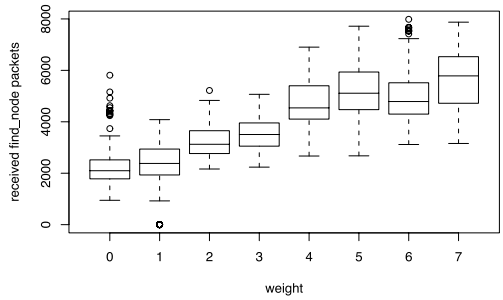

Fig. 2.2. Distribution of received FINDNODE packets; $k = 5, w < 8$

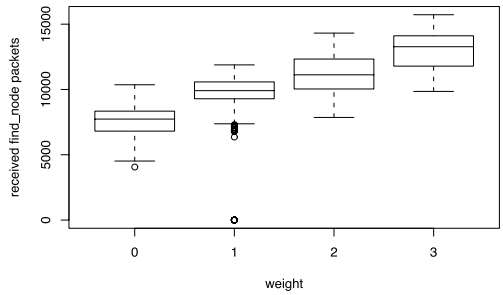

Fig. 2.3. Distribution of received FINDNODE packets; $k = 20, w < 4$

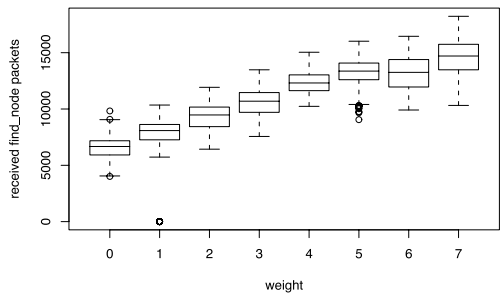

Fig. 2.4. Distribution of received FINDNODE packets; $k = 20, w < 8$

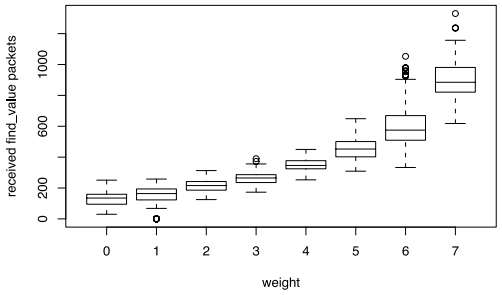

Fig. 2.5. Distribution of received FIND_VALUE packets; $k = 20$, $w < 8$

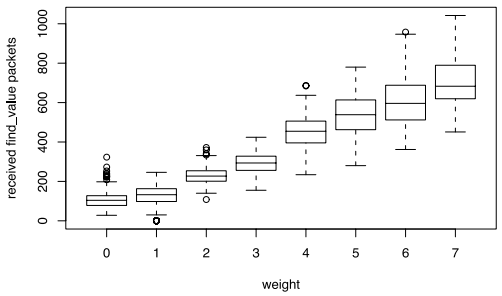

Fig. 2.6. Distribution of received FIND_VALUE packets; $k = 5$, $w < 8$

to weight, even though p_w has been made exponentially proportional to weight. This can be explained by the FIND_NODE algorithm of Kademlia; it eventually resends the FIND_NODE to all of the k closest nodes it has not yet queried, making α-node selection algorithm affect more weakly.

In contrast, the number of received FIND_VALUE packets exhibit exponential trend if k is large (Fig. 2.5); smaller k weakens this trend (Fig. 2.6). The exponential trend is presented more strongly in FIND_VALUE because it can immediately terminate once the RPC initiator obtains value from one of the responders. In other words, the k-node lookup for terminating the request, as employed in FIND_NODE, is not necessary as long as the value exists.

Next, we measured the effect of using different k values among classes. When we used $k = 5$ for $w < 4$ and $k = 20$ for $w \geq 4$, the number of FIND_NODE packets became twice as much as that of uniform $k = 20$ (Fig. 2.7). This can

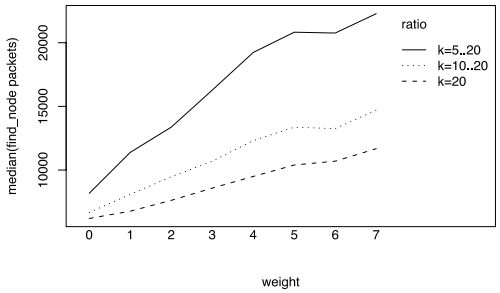

Fig. 2.7. Median of FIND_NODE packets for different k values

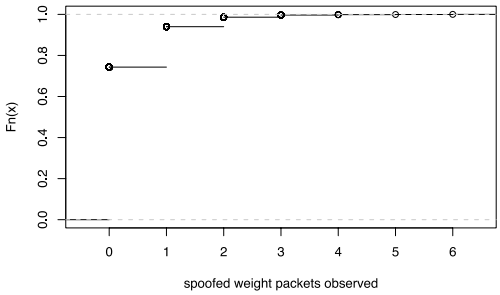

Fig. 2.8. Empirical CDF of spoofed weight packets observed at each node

be intuitively explained as follows: nodes with larger k-bucket require multiple replies from nodes with smaller k-bucket in order to find the closest k nodes. This implies that the use of smaller k value should be limited to small fraction of nodes; it can be useful when STORE requests to closest k nodes are frequent, or when the memory requirement of larger k-bucket cannot be accommodated.

Next, we measured the detection rate of packets with spoofed weight. In this test, one of the 128 nodes advertises false weight value for 180 seconds and other nodes attempt to detect spoofed weight by generating *check weight* packet at the fixed rate of 0.78%. The empirical CDF of spoof observations at each node are plotted in Fig. 2.8. Since spoofed packets are sent to random nodes in the k-bucket, the results were almost identical regardless of k value.

2.5 Related Work

Techniques exist to accommodate heterogeneity by creating two-tier structure on top of overlays: the *expressway* of eCAN[330] and *secondary overlay* in Brocade[329] are such examples. Such approaches quickly become incomprehensible if one attempts to construct multiple tiers of overlays. These techniques may find it difficult to accommodate nodes if they have orders of magnitude difference in bandwidth, processing power and memory, and if their performance heterogeneity and resource heterogeneity cannot be simply categorized into two groups.

2.6 Conclusion

Many DHT algorithms assume that each participating node can accommodate comparable amount of resource requirements. Such an assumption becomes unacceptable if one attempts to connect heterogeneous devices with single overlay.

We have shown that resource heterogeneity can be accommodated by making minor modification to certain class of DHT algorithm. We described extensions to Kademlia that introduce *weight* as a new node attribute. Required modifications to support *weight* are minimal: the α-node election algorithm and packet formats are enhanced to take *weight* into account, and two key parameters of Kademlia, k and α, are made node-local.

Furthermore, we have introduced two new RPCs, *check weight* and *check fairness* RPCs, in order to detect false advertisement of *weight* and unfair escalation of k values. These RPCs are probabilistically generated to detect misbehaving nodes, thereby ensuring fairness.

The proposed enhancements are incorporated into our Kademlia code, through which we have evaluated the feasibility of such extension. As a result, we have found that such extension can be easily made. Through emulations, we confirmed that load distribution can be controlled by weight and that false advertisement of weight can be identified.

第 3 章 *i*-WAT: The Internet WAT System — A Medium for Cooperation in Distributed Autonomous Systems —

Abstract

This report proposes a currency system called i-WAT[250] to promote sustainable barter relationships among subsystems of a distributed autonomous system. i-WAT itself is an autonomous system, without necessitating a central point of authority. It uses a digitally signed, electronic form of promissory note as the medium of exchange.

This report illustrates how it should assist building of distributed autonomous systems and their internetworking, and discusses, in particular, how trust can be maintained in such a currency system.

Among other means, the current design of i-WAT allows the notes to be transported over Jabber[186, 187] instant messaging protocol. A prototype of an i-WAT checkbook has been developed as a plug-in for a Jabber client. We are beginning to experiment on the actual usage of the currency system using the checkbook as well as provisional web applications.

3.1 Introduction

3.1.1 Peer-to-peer is a form of economy

My Japanese dictionary says that economy is "the act and process of production, distribution and consumption of goods and services which forms the bases of human communities, and the whole body of social relationships built upon such activities." It is not just about saving but about how finite resources are distributed in or among communities, which influences on how people interact with each other.

In this sense, economy and distributed autonomies, or so-called *peer-to-peer*, are closely related; in fact, peer-to-peer is a form of economy, in which distribution of resources is performed without the necessity for central coordination.

3.1.2 Economy in autonomous distributed systems

Recent interests in *distributed algorithmic mechanism design*[80], unified efforts between economics (*mechanism design* part) and computer science (*distributed algorithmic* part), shows that researchers of distributed systems are beginning to pay more attention to incentives for cooperation and fairness of sharing resources.

Distributed autonomous systems require coordination among nodes to achieve their goals or to satisfy their requirement specifications. Since each node may behave selfishly to maximize its benefit, sense of *incentive-compatibility* becomes important, which is a state of the collection of selfish behaviors of the nodes resulting in achievement of the goal as the whole. Relationships among nodes in such a system necessitate fair exchange of the computing resources. Media for barter relationships seems an essential part of such a design, which may take forms of points or barter currencies[250].

3.2 Peer-to-peer currency

3.2.1 Reason for peer-to-peer currency

Money plays an important role in economy. As a medium of exchange, it eliminates the need for *double coincidence of wants*, in which each of the two parties is willing to consume what the other is producing. Resources are more effectively distributed without such a need for coincidence.

Today, however, money looks more like a problem than a solution. Its another role, a medium for accumulating wealth, has resulted in scarcity of the media, dividing the world into haves and have-nots, the former having control over the latter.

To be independent from such control, and to achieve sustainable local economy even in presence of attacks or global/national depressions, alternative forms of money have been proposed and tested. Successes of such experiments include Wörgl[256] in 1932, in Comox Valley[258] in 1983 and in Ithaca[87] since 1991. The one in Comox

Valley promoted barter relationships.

Many of the successes are short-lived, however, because most designs of alternative money are dependent on the qualities of their administrations. Many experiments owe their successes to their first administrations which were more adequately motivated than later ones.

It would thus benefit the sustainability of economy if we could design a currency system without the necessity for central administration. As far as sustainable economy is concerned, currencies, too, need to be peer-to-peer.

3.2.2 Example of peer-to-peer currency

WAT System[316] is one of a few examples of existing peer-to-peer currency, in which a form of promissory note called *WAT note*, a physical sheet of paper, is used as the medium of exchange. Fig. 3.1 shows the three types of trade in WAT System:

1. Drawing trade
   A person in want of some goods or service becomes a debtor, and issues a WAT note. The debtor writes on the note the name of the provider of the goods or service, the amount of debt[2] and the debtor's signature. The debtor hands the note to the one who becomes the creditor, and in return obtains the goods or service.
2. Using trade
   The creditor can use it for another trading. To do so, he or she writes the name of the payee on the back of the note. The payee becomes the new creditor, repeating which the WAT note circulates among people. The length of the chain of creditors shows how much trust the note has gained.
3. Liquidating trade
   The WAT note is invalidated when it returns, as a result of a trade, to the debtor.

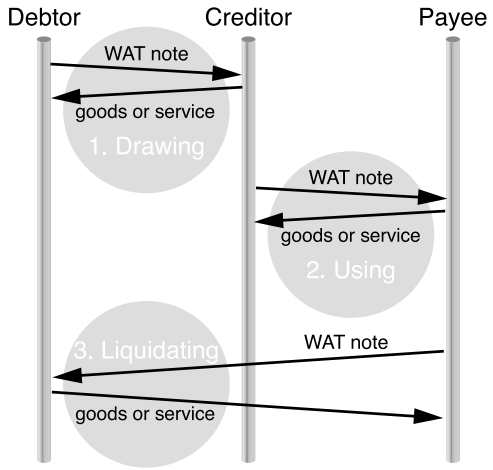

Fig. 3.1. Trading with a WAT note

WAT System is a *free* currency in the following ways:

1. Administration-*free*
   Anyone can spontaneously start WAT System with a sheet of paper if they follow a few rules.
2. Interference-*free*
   It is independent of national or global economy.
3. *Free* location
   Any WAT note is compatible with any other WAT notes in the world, and the currency system is globally operable (although within the limit where one's credit can be trusted).

3.3 *i*-WAT: the Internet WAT

3.3.1 Overview

We developed *i*-WAT[250] as an extension of WAT System so that it can be used on the Internet. It is intended to be used by people or by autonomous programs in distributed systems.

The medium of exchange in *i*-WAT is a message signed in OpenPGP[21], by which transferring the ownerships of electronically represented WAT notes is implemented. The exchanged messages are called *i-WAT messages*, and the note represented by the messages is called an *i-WAT note*.

Table 3.1 shows the types of *i*-WAT message.

All *i*-WAT messages are signed by its sender,

2 Typically in a unit called *WAT*, which represents cost of producing electricity from natural energy sources, but anyone can create their own units.

Table 3.1. *i*-WAT messages

No.	Message name	Function
1	*i*-WAT <draw>	Draws an *i*-WAT note.
2	*i*-WAT <use>	Uses an *i*-WAT note.
3	*i*-WAT <accept>	Confirms the readiness to accept the provided *i*-WAT note once its validity is verified.
4	*i*-WAT <reject>	Rejects an *i*-WAT note.
5	*i*-WAT <approve>	Guarantees the validity of an *i*-WAT note, and approves the transaction.
6	*i*-WAT <disapprove>	Denies an *i*-WAT transaction.

and are formatted in the canonical form of XML[17] which handles nested signatures well.

3.3.2 Protocol

The three types of trade are implemented as follows:

3.3.2.1 Drawing trade

1. The debtor sends *i*-WAT <draw> message which contains the e-mail addresses of the debtor and the creditor, an identification number and the amount of debt. This message becomes the original *i*-WAT note after the protocol is completed.
2. The creditor sends back the *i*-WAT <draw> message to the debtor. This is called *i*-WAT <accept> message.
3. The debtor sends an *i*-WAT <approve> message to the creditor.

3.3.2.2 Using trade

1. The creditor adds to the *i*-WAT note the e-mail address of the payee, and sends it to the payee as *i*-WAT <use> message. This becomes the valid *i*-WAT note after the protocol is completed.
2. The payee forwards the *i*-WAT <use> message to the debtor as an *i*-WAT <accept> message. If the creditor wants to use multiple *i*-WAT notes at once, the payee must forward all the *i*-WAT <use> messages to all the debtors.
3. The debtor verifies the validity of the note, and sends an *i*-WAT <approve> message to the creditor and payee, as well as all other debtors in case multiple *i*-WAT notes are used at once, in order to assure atomicity of the transaction; the notes will not be transferred to the payee unless all <approve> messages are collected.

3.3.2.3 Liquidating trade

1. Like using trade, the creditor sends an *i*-WAT <use> message to the payee.
2. If the payee equals the debtor, the debtor invalidates the *i*-WAT note as the debt is now liquidated. The debtor sends *i*-WAT <approve> message to the creditor.

Fig. 3.2 shows the most complicated case where a creditor uses multiple *i*-WAT notes issued by different debtors.

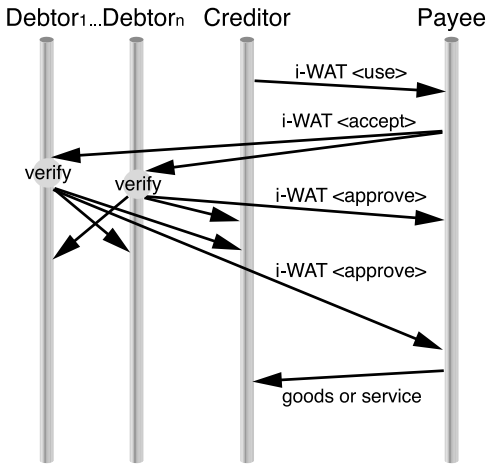

Fig. 3.2. Trading with *i*-WAT messages

3.4 Applications

3.4.1 DCR: Distributed Consumer Reports

DCR is an attempt to create a structure for administration-free exchange of information among consumers about the goods and services they use. Its challenges are to maintain liveness of and credibility to the circulated information, to which we approach by letting subscribers of reports make payments in *i*-WAT to the reporters, thereby offering the information providers motivation and responsibility.

The current design of DCR utilizes DHT (Distributed Hash Table)[3], but the specific algorithm to be used is still undecided. DCR makes multiple nodes responsible for a given item, by suffixing predefined salts (such as 0, 1 and 2) to the item's key[4], in order to avoid having a single point of failure.

Fig. 3.3 shows how DHT and *i*-WAT can be applied to making and circulating distributed consumer reports.

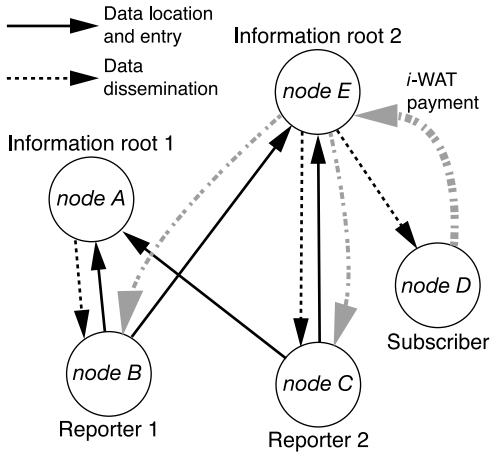

Fig. 3.3. Application of DHT and *i*-WAT to consumer reports

In the figure, nodes A and E are the *information roots* for a given item, based on the DHT algorithm in use. For a different item, a different set of nodes would be selected as its roots. The selection is made autonomously and deterministically (perhaps as part of location protocol) without imposing an expensive agreement protocol. Reports are compiled at the information roots based on the information gathered from the *reporters* (nodes B and C). A *subscriber* (node D) can obtain the reports from either of the information roots (decided by proximity), and pays an agreed amount in *i*-WAT to the one they choose to use. The paid amount is shared among the root and the reporters.

The reports can be either obtained by queries, or disseminated among registered users. An example of data dissemination over a DHT-based overlay network is found in [331].

3.4.2 Share-ik: an alternative copyright system

Our research group is developing an alternative copyright system named *Share-ik*[156] (Share intelligence and knowledge), which encourages derivation of new work from copyleft[86] materials.

Share-ik is essentially a set of rules close to the GNU General Public License[85], which defines a logical shared space from which public domain materials and their descenders may not leave. In order to apply this idea of "copyleft" to artistic creations such as music, Share-ik adds a rule for mutual evaluation: when new work is produced by reusing some existing work in the shared space, the reuser is asked to make a payment in *i*-WAT (in a unit called *share*) to the authors of the

3 In short, DHT provides a lookup service of <key, value> pairs. It provides an application-independent function which maps a key to the responsible node in the system. Typically, keys and node identifiers share the same n-bit name space, and are evenly distributed in the space using a one-way hash function. Given a key (and a message), a DHT system will find a corresponding node (*location*), and optionally routes the message to it (*routing*), which may return an application-specific value. Both location and routing are deterministic, and performed within a bounded number of overlay hops, which typically grows logarithmically to the number of participating nodes. This is more desirable for many applications than broadcasting search requests as seen in Gnutella[225]. Several DHT-based overlay network prototypes have been proposed, including Chord[271], CAN[237], Tapestry[328] and Pastry[248].

4 A technique introduced by Tapestry[328].

original work whom they appreciate. *i*-WAT protocol lets the original authors choose not to receive the payment for typically two reasons: 1) they do not agree with the new creation, or 2) a distributed auditing (section 3.6.3) shows that the credibility of the reuser is doubtful. The latter can be a result of their work not being appreciated by others in the community, leading to much less income in *share* than its outlay.

In this way, the original authors can know how much appreciation their works are receiving, and the reuser can know whether their creation was agreeable to the original authors, or if they are welcome in the community.

The result of the mutual evaluation is made publicly available within opus information documents formatted in the canonical form of XML, which allows partial content to be signed, so that the signatures of the authors of original work can be verified by other users.

3.4.3 SAFEE: SpAm-Free E-mail Exchange

SAFEE takes a psycho-economical approach to solve the problem of spam. Let us define spam as an e-mail message receiving which the receiver feels a loss rather than a benefit. Our solution is to counterbalance the loss with an *i*-WAT payment. The unit of payment in SAFEE is called *MU* (Mail Unit).

In SAFEE, there is one simple rule:

> The sender of a message must pay 1*MU* to each receiver of the message.

The assumption here is that if any two parties have a healthy relationship, on average the amount of messages exchanged between them is evenly balanced, and they should not feel loss (a difference within several messages is not a problem as *i*-WAT has a weak budgetary constraint). On the other hand, few people would reply to unsolicited e-mail, which means that the sender loses 1*MU* for each receiver of their message. The receivers gain 1*MU* each, which they can use for sending an e-mail message or exchange with other *i*-WAT currencies; the loss is compensated by the payment. Fig. 3.4 illustrates this idea of defusing spam.

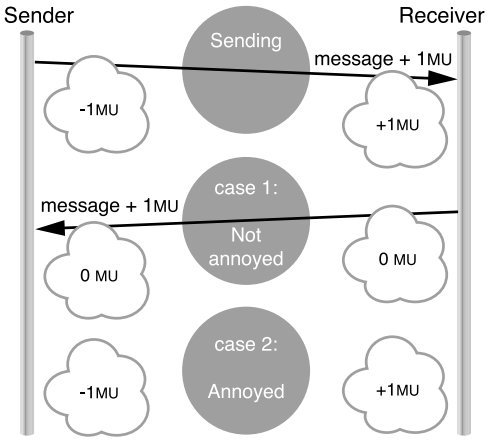

Fig. 3.4. Defusing spam in spam-free e-mail exchange

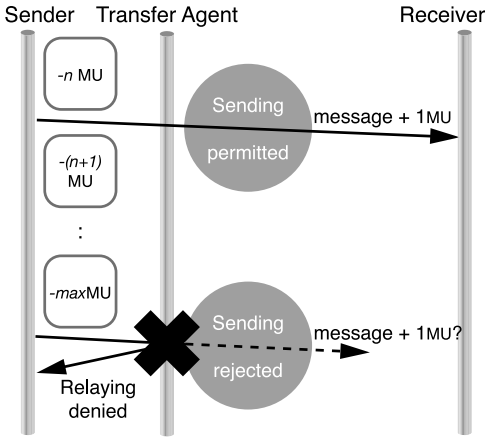

Fig. 3.5. Prevention of spam in spam-free e-mail exchange

This message transfer system can be augmented by a rule for the message transfer agents:

> A transfer agent must reject relaying a message if the negative balance of the sender's *MU* account is more than *max MU*.

where *max* is sufficiently large so that it does not prevent ordinary people from sending messages. A distributed auditing (section 3.6.3) can be used for checking the sender's account. Fig. 3.5 illustrates this idea of preventing spam.

Obvious problem with SAFEE is that since *i*-WAT is based on OpenPGP, it is easy to forge one's identity by creating key pairs for arbitrary

e-mail accounts. To handle such situations, the transfer agents can also check with a DCR query if the public key of the sender is backed up by trusted signatures. Also, the trust values of the senders obtained from a distributed auditing (section 3.6.3) may be used for prioritizing the transfer requests.

3.4.4 Other possibilities

Other, still highly conceptual applications of *i*-WAT include what we tentatively call ATP (Associating Transportations and Pedestrians). It is inspired by OSMOSE[169], a conceptual vehicle introduced by CITROËN at Paris Motor Show 2000. An OSMOSE vehicle can let pedestrians know its destination, and the driver can pick up a pedestrian or two who would like a ride.

Some apparent questions in designing an OSMOSE-like vehicle and its surrounding systems are trust and economy: what if the guest turns out to be a mugger, or driver a kidnapper? Is it morally or economically adequate to allow, quite literally, free ride? To answer these questions and provide solutions, there need to be mechanisms of authentication and showing appreciation, which should work regardless of locations (because vehicles travel). *i*-WAT seems to satisfy all these requirements.

Also being investigated is a free currency limited for educational purposes.

3.5 Internetworking with *i*-WAT

3.5.1 Incentives for internetworking

There needs to be reasons for nodes to participate in and cooperate across multiple peer-to-peer systems or even within a single peer-to-peer system. Although *i*-WAT may work as a medium of exchange, that alone does not mean that it will assist the designs of actual distributed autonomous systems. We need to provide a mechanism for incentives and fairness.

In a monetary economy, accumulation of bank notes forms an incentive. As we described earlier, this results in scarcity of the media, and does not promote a fair exchange. Therefore, instead of using accumulation of notes, we use accumulation of trust of the nodes to build an incentive-compatible mechanism. Our hypothesis is that trust can be expressed by how much transactions the node has successfully processed.

For example, the trust value of a node can be expressed in the following formula, which works as an incentive for the node to make transactions, and prompts the node for both using notes in possession and providing services for liquidating its debt, so that its income and outlay are balanced.

$$trust = \log \frac{income \times outlay}{|income - outlay| + 1}$$

This is the basic formula we use for now, a variation of which is discussed in section 3.7.2.

3.5.2 Example of exchange mechanism

The semantics of *i*-WAT inherited from WAT System allows the notes to be freely associated with values. Such values include *i*-WAT notes in different units in different currencies.

Fig. 3.6 shows how *i*-WAT notes in different currencies can be exchanged with one another.

The figure shows three communities, A, B and C, depicted as rings. These communities can be circles of people, rings of Chord[271] or some other forms of distributed hash tables, or

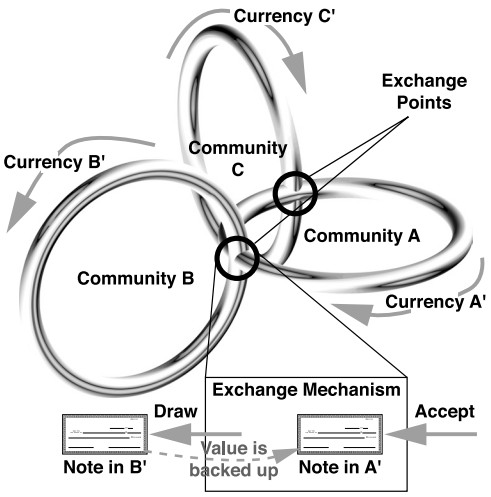

Fig. 3.6. Exchanging *i*-WAT notes among different currencies

just any groups of nodes in autonomous overlay networks. We assume that methods for message-routing exist among these communities. The communities use currencies A′, B′ and C′ respectively.

An entity belonging to both communities A and B can become an exchange point between currencies A′ and B′. Such an exchange point can take a note in A′, and draw a new note in B′. The value of the new note is backed up by the exchange point's possession of the original note in A′,

A node in community A can ask the exchange point to draw a note in B′ in return of a note in A′, The obtained note can be used to ask for some service in community B. *i*-WAT requires that the each end of a transaction must have the other's trusted public key. Those public keys can be signed by the exchange point.

The exchange points are motivated to collect the drawn notes and give up the original notes, as it will insure the increase of their trust values. They are also motivated to advertise their services.

If someone in community B wants to issue a note in currency C′, then they use the two exchange points in Fig. 3.6 to exchange a note in B′ to A′ and A′ to C′.

3.6 Discussions on trust

3.6.1 Embedded locality

In *i*-WAT, the debtors need to have the trusted public keys of all creditors appearing in the life-cycle of the notes they issued, because they are responsible for verifying all the transactions using the notes.

Some argues that it is unlikely to happen in real-life. Bu we believe that this defines the system's locality; the above condition can easily be satisfied in a small group of people closely working together, where every one can verify and sign each other's public key. The condition can also be satisfied, albeit marginally, according to the transitive nature of trust in PGP[1], because the both parties of a transaction must have each other's trusted public keys. While it is possible for an *i*-WAT note to travel across communities, the note would be more trusted if it stayed within one community. This is the reason for the need for exchange points described in section 3.5.2.

3.6.2 ID's accountability

The most frequent criticism against *i*-WAT is that it uses e-mail addresses (or whatever IDs coupled with PGP public keys) for identifying parties, which can be changed, reused or forged easily.

Suppose a PGP key-pair is generated for an imaginary person. Since the immediate parties of transactions must contain each other's trusted public key in *i*-WAT, the public key of the imaginary person must be signed by someone directly in acquaintance with the person: the creator of the person and thus, the forger. In order for the public key to be transitively trusted, the forger must be included in the chain of the creditors. If the imaginary person fails to liquidate the debt, as the forger intends, the forger him or herself must take over the debt being its immediate creditor. Therefore the cost of lying is considered high.

3.6.3 Distributed auditing

Since *i*-WAT is decentralized, calculation of the trust values of participating nodes needs to be achieved by collecting information from each transaction, and constructing an image of the node's account based on that information. There is no guarantee that the collected information is truthful if there are incentives for lying or colluding.

In order to tackle this problem, we first take a look at how a user's account information can be used for calculating their trust values. Table 3.2 shows how records of notes in a user's checkbook can be used in measuring their trusts.

We argue that the following statements are true:

1. There is no incentive to conceal the records of liquidated or used notes.
   The users would not claim less income and outlay in balance than there actually is because that would decrease their trust values.

Table 3.2. Meanings of *i*-WAT notes in the checkbook

Debtor of the note	Description	Role in trust value
This user	Liquidated	Balanced income and outlay
This user	Not liquidated	Negative balance or debit
Not this user	Possessed	Positive balance or credit
Not this user	Used	Balanced income and outlay

2. One cannot lie to have more debits by claiming to have drawn more notes, or more credit by claiming to possess more notes drawn by someone else, because they may be asked for a proof in an auditing process.
   If there is a proof, we believe that it should be considered a set of different problems: transactions without actual practice of bartering, and inflation of the value system which might result from such transactions. We believe there can be operational solutions for this sort of problems, and experimenting on solutions in the actual barter economies described later.
3. The only reasonable way to tell a lie is not to reveal the existence of debits or credits.
   As a countermeasure for this, we can apply the protocol for *fair sharing* described in [211].

Protocol to detect concealed debits (CD):

CD-1. At a random interval, to a randomly chosen user, one asks for a list of their possession of notes.

CD-2. For each note in the list, the one asks its debtor for the list of drawn notes.

CD-3. If the note in question is not included in the list, the debtor is lying about their debit.

Protocol to detect concealed credits (CC):

CC-1. At a random interval, to a randomly chosen debtor, one asks for a list of their drawn notes.

CC-2. For each note in the list, the one asks its current owner for the list of all notes they possess.

CC-3. If the note in question is not included in the list, the creditor is lying about their credit.

Note that in the above protocols, CD-1 and CC-2, as well as CD-2 and CC-1, are indistinguishable to the receiving end of the queries. Therefore there are disincentives to lie to the queries.

A debtor and the immediate creditor may have a reason to collude. They might lie that the transaction never took place. However, the relationship between a debtor and the immediate creditor is not symmetrical. It is riskier for the creditor because lying means denial of their crediting debt.

We are investigating further to make the distribute auditing an inexpensive process.

3.6.4 Sustainability

Trust is also dependent on sustainability of *i*-WAT itself. *i*-WAT inherits the polycentric nature of WAT System, and should be difficult to break.

In *i*-WAT, the debtor is responsible for guaranteeing that the circulated note is not a fraud. In this sense the debtor is privileged, but it is not a single point of failure because once the debtor fails the immediate creditor takes the debtor's role; the function of the debtor follows the chain of creditors.

Which means that the rational behavior is never to take an *i*-WAT note directly from a debtor. Wwhile it is very true, people do not always act rationally, especially in a small group of people where everyone knows each other well. There is some risk if a note travels across communities, but as described in section 3.5.2, a community member is never obliged to take notes issued by members of other communities.

3.7 Deployment and experiments

3.7.1 Jabber-based *i*-WAT

i-WAT needs a decentralized message transport to assure its sustainability. While a DHT-based overlay network is being investigated to provide such functionality, we need to verify the design of *i*-WAT by quickly deploying it even in absence of a desired infrastructure.

i-WAT allows the underlying carrier of messages to be existing e-mail or presence/instant messaging system. A prototype of an *i*-WAT checkbook has been developed as a plug-in for a Jabber[186, 187] client, which will be made available to public soon.

A pre-release version has been used by a small group, and there are some findings. In particular, we discovered that even people with sufficient knowledge of PGP found the key exchange cumbersome. Perhaps there needs to be a support for secure public key exchange as a subsystem of *i*-WAT.

3.7.2 OMELETS and WIDE Hour

With the distributed auditing described in section 3.6.3, we can treat the peer-to-peer currency system as if the account information comes from central sources.

There is an example of a barter currency with central authority called LETS (Local Exchange Trading System), which was first introduced in Comox Valley[258] in 1983.

We have developed OMELETS (Open, Modular and Extensible LETS), a collection of Java classes to implement LETS as a web application, in the hope that it becomes useful in verifying the designs of mechanisms using barter currencies.

An application of OMELETS have been developed for WIDE Project[318], a research project of distributed systems which has more than 700 active members. The barter currency for the project is called WIDE Hour[317] (the web site is for WIDE members only), based on the number of hours of labor for the project. The trust value is called *WIDE Power*, given by the following formula:

$$\begin{aligned} &\mathit{WIDE\ Power} \\ &= \log \frac{\mathit{income} \times \mathit{outlay}}{|\mathit{income} - \mathit{outlay}| + 1} - \mathit{penalty} \end{aligned}$$

where *penalty* is decided by the administration.

The maximum WIDE Hours each member can spend are limited to 24 WIDE Hours a day.

WIDE Hour was introduced in a four-day meeting in September 2003, and more than 600 transactions had been processed by the end of year 2003.

3.7.3 Internetworking between web and peer-to-peer barter currencies

i-WAT can also implement WIDE Hour, and we are planning to connect the two implementations together in March 2004. Fig. 3.7 illustrates how it should work.

WIDE Project has a strict notion of membership, but its activities often involve non-members. While it does not always make sense to provide non-members with accounts in OMELETS version of WIDE Hour, *i*-WAT notes in WIDE Hour can always be issued outside the project. By the internetworking mechanism, such notes can be made compatible with the OMELETS version of WIDE Hour.

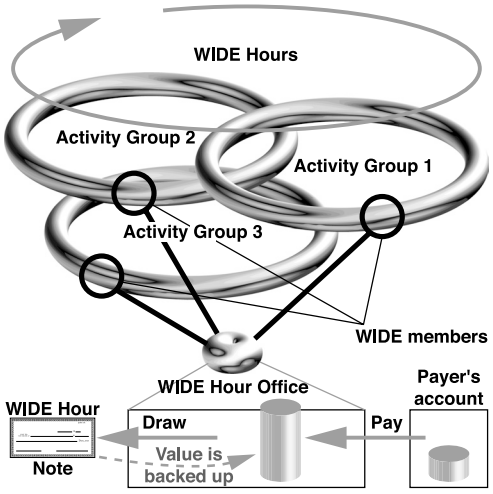

Fig. 3.7. Exchanging WIDE Hours outside the WIDE members

In the figure, there are two types of overlays. One is the overlays of activity groups, and the other is the overlay of OMELETS version of WIDE Hour. The former is peer-to-peer and the latter is a star network. The former overlays can interact with each other using *i*-WAT notes in WIDE Hour. A WIDE member can obtain such a note by asking WIDE Hour Office for an exchange.

It works just like the exchange point in Fig. 3.6, only that the office accepts payments in LETS, which backs up the value of the new *i*-WAT note.

3.7.4 MANA

MANA is another application of OMELETS, involving a book[203] equipped with an Auto-ID[8]-compliant 2.45 GHz RFID (Radio Frequency IDentification) tag.

The barter economy of MANA allows the users to obtain points by visiting certain locations where RFID readers are placed, and having their books identified by the readers. Obtained points can be used in a community residing on the web[179].

This is a large-scale experiment; about 10,000 copies of the book is expected to be circulated by March 2004, and we expect that about 2,000 people will participate in the experiment.

We plan to experiment on internetworking between web and peer-to-peer currencies using this barter economy, too.

3.8 Conclusions

This report proposed usage of a peer-to-peer currency system called *i*-WAT to promote sustainable barter relationships among subsystems of a distributed autonomous system.

i-WAT inherited its polycentric nature from WAT System, its predecessor in the physical world. It is carefully designed not to introduce any single point of failure. Trust is also maintained without necessitating a central authority.

A prototype of an *i*-WAT checkbook as well as provisional web applications has been developed. Experiments are ongoing.

第 4 章 Design of Content Cruising System

Abstract

Advancement in wireless networking and mobile computing technology has brought a new chance to communicate not only with a particular person but with general public who are in a specific area in real space. Geographically dependent contents such as advertisement to a specific region and traffic information are suitable for this kind of communication. If these contents are bound to appropriate time and place, the value of contents becomes higher for both senders and receivers.

The goal of our research is to realize a new content distribution model where contents will be gathered only into specific area autonomously without centralized management but with cooperation of mobile nodes. The use of this model will give users a freedom to send or leave contents about specific location and time, as well as giving users a chance to receive all the information bound to their locations.

The achievements of our previous research were published as a set of proceeding papers[136, 137]. In this report, we present the design of a simple but robust model to exchange contents among nodes within certain locations. We call the model "Content Cruising System (CSS)". CCS is a kind of peer-to-peer system which requires nodes to collaborate with one another using a commonly shared transmission algorithm.

4.1 Introduction

There are many kinds of information surrounding our lives which depend on geography. The information such as advertisements of local stores or traffic information is more appreciated when it is distributed to people living or visiting nearby

rather than to people in distant locations. Such contents will have higher value when they are shared among people in that specific time and place.

There exist distribution services of geographically-dependent contents to mobile nodes according to their locations. Such services are generally called "location-based services."

These services are divided roughly into two patterns of approaches.

One is to provide contents to wide areas, by utilizing cellular phones, for instance [139]. This approach requires costly equipments and presence of a central management, which limits the users' freedom of communication. Moreover, it makes the services not dependable, in a disastrous situation where the infrastructure is damaged, or when we are in some developing region without such an infrastructure in the first place.

The other is to provide contents to local areas, with a number of hotspots and divide each area in pieces [34]. The use of this kind of services is limited to places with hotspots. Moreover, it is difficult for content senders to appoint target areas minutely.

In order to solve such limitations, we have been conducting researches on building another structure to propagate contents depending on the locations of nodes.

In this report, we propose a simple but robust model to distribute contents used in real space, wireless communication environment. We call it "Content Cruising System" (CCS). The main aspect of proposed model is that it does not require centralized servers to distribute information, and it does not depend on types of network connections either. In this model, nodes themselves exchange contents with one another, without requiring the presence of content servers or network infrastructures.

4.2 Approach

We have been studying a structure of another content propagation method in the application layer, using "physical movements" of the mobile nodes. This approach is not always premised on routing on the network which has fixed durability. A content is broadly stored through a chain of accidental and temporary communication events performed when mobile nodes move close to one another (we define this as "Ad-hoc Communication"). The content has a chance to be forwarded further during some physical movement of a node, resulting in a distant unspecified node receiving contents. This is a mechanism similar to rumor, word-of-mouth communication in actual societies, so transmitted contents are spread disorderly through multi-hop communication. However, such information flooding can be suppressed to fixed geographical areas by using each node's location information (which obtained from commonly used device such as a GPS receiver), and we hope that this mechanism will establish some new communication models depending on the locations of people. If it is possible to make contents to remain in a specific area, the mobile nodes which exist in the specific place will transmit these contents to one another. Furthermore, since this communication model is independent from each node's network-reachability, the node that newly enters into the area will receive the remaining content, so the content-exchanging among the nodes will be kept in the specific area now and in the future. This can also be a robust information system for guiding people to some destinations, or seeking a missing person in a disastrous situation where usual means for communication are no longer reliable.

The content transfer model performed on the Internet so far is the movement of the content between the nodes fixed on the network. Opposed to this model, the content transfer model we are proposing is so to speak a "locally circularizing model", because the node itself moves and gathers the content which is fixed locally, and leaves the content to an area to attain the content transfer among unspecified nodes. Although we do not specify the format of contents in our system, we

assume the contents to be small in size, such as voice messages or pictured maps, in the real-world applications.

4.3 Related Work

"Geocast"[205] is well-known as a method of location-based communication on the Internet. This method is realized by adding geographic information to the internet protocol. Ad-hoc network can be formed using one of many routing protocols which are different from the internet protocol. In MANET [194], the way of location based routing, or multicasting a packet to two or more nodes which exist in a specified geographic area on the Ad-hoc network is examined[157, 166]. These may be suitable for our system as lower layer protocols in the future.

In a sense, our aim is to develop a kind of distributed content storage system in real space. In the field of overlay network, there are many proposed ways for designing distributed storage system[39, 163, 271, 328]. As a commonly-observed feature, these mechanisms use distributed hash table as a file name space, and provide distributed data storage and ways for searching data effectively. These approaches are efficient in terms of decentralization of contents, although our mechanism requires that the contents are weighed by their physical locations. Above all, "Freenet"[39] has interesting mechanisms for content distribution: anonymity and encryption of the contents. We have similar requirements on relaying contents in real space.

4.4 System Design

4.4.1 Overview

Based on the concept described earlier, we designed a mobile peer-to-peer location based content distribution model called "Content Cruising System" (CCS), and devised a transmission algorithm based on location information when exchanging contents among mobile nodes in a real space wireless ad-hoc communication.

CCS sets its goal on providing highly robust and simple method for delivering contents to unspecified number of users being close in the real world. In order to achieve this goal, it utilizes excess computing resources in each node, as well as excess network resources. Contents are autonomously converged into targeted areas and kept in those areas. Therefore, CCS will not limit the size of the contents and communication method below the application layer.

CCS is operated by cooperation of servant applications which have same function installed on mobile nodes, and the thrown-in contents circulate through "store and forward catenation" with neither continuous formation nor negotiation among the nodes.

The Objective of this system is to build a general scheme listed below.

- Location based transmission: contents are transferred and sustained in a specified area autonomously.
- Location-aware: contents are found by people in a relevant location
- Context based selection: unnecessary contents are eliminated based on the user's context.

4.4.2 Analysis of Communication Environment

We are assuming there are some specific conditions in ad-hoc communication, compared to usual communication environments. Therefore we will model the communication environment and settle the conditions to perform the content transferring described in the foregoing paragraph.

Hardware Requirements

Our target machines have functional capability shown in Table 4.1.

Functional Requirements

Besides a computing power, each node has at least the following four functions:

- Receive the content
- Transmit the content

Table 4.1. Hardware Requirements

Functional capability	Assumed hardware
communicate with neighbor nodes	IrDA, Bluetooth, Wireless-LAN,etc.
store data	HDD, RAM, Silicondisc, etc.
sense self status	GPS, Clock, and other sensors.

- Display (or play) the content
- Store the contents

Prerequisites of Communication

With the kind of ad-hoc communication mentioned above in mind, we set the following prerequisites of communication:

- No specific management serves.
- Each node is independent and not controlled mutually.
- Autokinetic communication by cell broadcast.
- The content sender does not have knowledge of the receiver's situations or location.
- Limit on each node's disk space.

4.4.3 Requirements

According to the conditions of ad-hoc communication, we assume there are no negotiations among the mobile nodes for transmitting contents which are once thrown-in. Therefore, it is needed to satisfy some functional requirements in order to realize the structure of CCS.

The required functions and our solution are described below.

Unique Content ID

To avoid duplication of the contents, unique identifiers will be needed for each content. In creating a unique identifier, we plan to use a kind of one way hash functions such as SHA1[204] or MD5[246] based on the combination of the MAC address and content generation time.

Context Awareness

Because each node is independent, it is impossible to take any action without some kind of trigger. By configuring the trigger as a common requirement among servants, collaboration as a whole will become possible. As an item in the common requirements, we decide to use the each node's context (including its location and time), which is acquired by their own sensors as a trigger for each node's action (sending, displaying, or accumulating).

On that basis, the sender adds meta data to the content header called COMPASS (COmmon Message PASsing Sheet), which includes the expected destination point and time in order to share ways of handling the contents in all participating nodes. Receiving nodes compare their own context with the COMPASS, and use it as a trigger to decide their actions.

Selection of Contents

Since each node has limited storage capacity, a mechanism is needed for automatically deselecting contents that do not match the user's context. We consulted the LRU algorithm[278] that data will be eliminated from the domain where operating frequency is low when cache is full. Moreover, each content has its own predefined duration.

Weighting Content and Transmission algorithm

To avoid meaningless flooding of contents, the sender must define "the condition in which the content becomes most valuable" and transfer it to the other nodes. Using as a location-based service, the contents become most valuable when they reach their destination points. Therefore, the value of each content is defined as a unit in "Centripetal Force" from the destination point.

4.4.4 Configuration of Transmission algorithm

If each of the contents has a specific target

region, the *Centripetal Force* should be maximized only in that area which has a specific shape (orthogon, circle, oval, etc.). However this method is not suitable for our system because each content's *Centripetal Force* is apt to be conflicted in overlapping portions where many con-tents are thrown-in. Our system aims to cover any places as long as there are mo-bile nodes. It also has to deal with numerous contents as well.

In order to avoid the confliction, *Centripetal Force* should be calculated relatively between each node's present location point and the content's destination point rather than between those areas, because "point" is more innumerable than "area". In addition, our system aims to leave the chance for each content to be transmitted widely by physical movements of participating nodes, in contrast to the other aim to sustain contents in certain range. Therefore we thought the *Centripetal Force* should not be regulated in obvious range by the system.

Based on above reasons, we chose exponential decrease as a formula to calculate the *Centripetal Force* among the other monotone decrease function to start with.

The *Centripetal Force* is calculated as follows:

$$F(d) = ae^{-kd}$$

"d" is the distance from the destination point, "a" is the value at the destination, "k" is the parameter which define the dependence with distance.

Every content is affected by this Centripetal Force, and every node which possesses the content does the same actions according to Centripetal Force felt in the location.

As previously mentioned, Centripetal Force is a reference to measure the importance of the contents in that location. The receiving ratio at each point should be similar to the Centripetal Force respectively. Hence, CCS uses the reciprocal of the exponential function above as the time interval for the next transmission to take place.

The relation between the Centripetal Force and interval times of sending is shown in Fig. 4.1.

4.4.5 Framework of Servant Application

In the CCS environment, all nodes have the same unsophisticated functions performing common behaviors. The functions of the servant are comprised from the following four modules. Their relation is shown in Fig. 4.2.

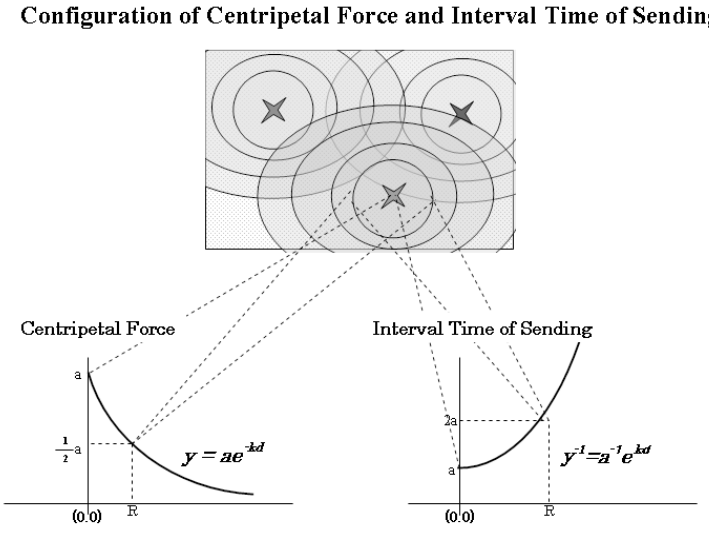

Fig. 4.1. The relation between the Centripetal Force and interval times of sending

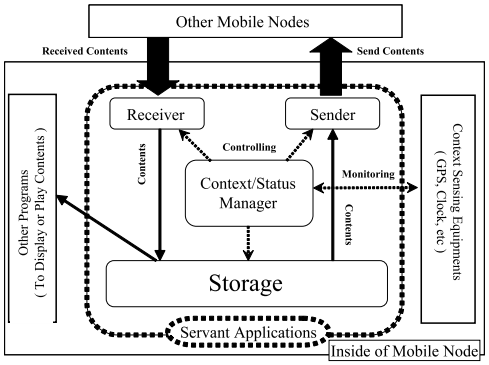

Fig. 4.2. Relation of servant modules

Sender Module

This module sends contents which are stored in the Storage module to the neighbor nodes according to the direction by the Context Manager. Contents are broadcasted to all nodes in its radio signal range.

Receiver Module

This module receives content from the other nodes, and put it into Storage module.

Storage Module

This module stores the received content in FIFO order, because the more adequate the content is, the more frequently it is received, according to the transmission algorithm of CCS. The content which is staying longer than its predefined duration is deleted by Context manager's command.

Before storing each content, the storage module compares the Content ID of the new content with those of the old ones which the node has ever received. If duplication is detected, the storage module discards the old content.

Context/Status Manager

This module get information of the condition of the node (defined as"Context") from its sensor (GPS, clock, etc.) continually. It compares the context of the node with the COMPASS of the content, and calculates the Centripetal Force to prompt each module to function adequately in order for the content to be distributed/circularized in the specific area.

4.5 Future Prospects

Though only the information distribution model to many and unspecified people is dealt with in this report, it is also possible to carry out a specific partner and specific content switching by the secrecy of a content and a user's authentication using Hashed-Key technology[204, 246].

This mechanism would be needed to avoid Denial-of-Service such as inserting a large number of junk files[39].

Moreover, in the present circumstances, mediator is premised to transmit every content it received, but the contents should be weeded out according to a certain demand of users for them, as it is the case for rumors.

In addition, we substituted transmitting frequency for *Centripetal Force* this time, but in case it is required engineering design about an efficient resource management for the problem of a transmission bandwidth and that of node's electric power.

CCS makes various distribution models possible by giving transmission rules to a COMPASS. Besides the distribution model treated as an example in this report, the various distribution models with the combination of different rules (such as a content-distribution using a mediation person's or their transportation devices' characteristic, content-distribution using the reservation function by time, and so on) should be considered.

4.6 Summary

The purpose of our research is to realize the contents distribution system which is decentralized in form that mobile nodes distributed in various places sharing the functions to store and to distribute the contents.

In this report, we proposed the design of Content Cruising System to realize the form of a decentralized content-transmission system in real space. This system brings about the mechanism in which contents are transmitted

autonomously, by adopting a simple algorithm in ad-hoc communication. The catenation of content-transmission in a specific area provides for location based communication that does not require a specific server or a specific infrastructure.

We are now continuing to experiment using our original simulator to estimate the efficiency of our algorithm. The achievements of our previous research were published as a set of proceeding papers[136, 137]. We will examine the optimal combination for various situations in our future experiments.

第 5 章 Peer Group Rendezvous Using Intersection among Peer Groups on DHT

5.1 Background

5.1.1 Motivation and the problem

Peer group is a strong and flexible structure to enable coordination between applications, server-client, and peers in networks. The idea of peer group is originally introduced in IS-IS toolkit[14], but on the Internet we are still unable to use the concept in large area, large group, and inter-administrative domain.

A peer means an equivalent node. In this report, the term equivalent is loosely as 'to share one or more common characteristics' and a peer group is a group of nodes that share such characteristics. For example, group of printer and group of nodes in a room are both peer group.

A contemporary middleware, Project JXTA[273] uses the idea of peer group as its foundation. Although JXTA's peer group is secure and capable to act in ad-hoc environment, some investigation[96] tells current JXTA implementation has limitation on data transfer. Thus it may have performance bottleneck, mainly due to its complexity.

This report's contribution is in *rendezvous process* using concept of peer group. Rendezvous is the process to find nodes to communicate. It includes service based, interface based, time based, and free keyword based rendezvous.

Due to rise of number of Internet nodes and variety of node characteristics, rendezvous in large scale is important and should not be overlooked. We have been making rendezvous with DNS, search engine, and some of service location or directory service. But they seldom scale, or just give us simple name binding like DNS, or just cover major content like search engine.

Fig. 5.1 describes segment map of such rendezvous technology.

The x-axis is local-global attribute. In local

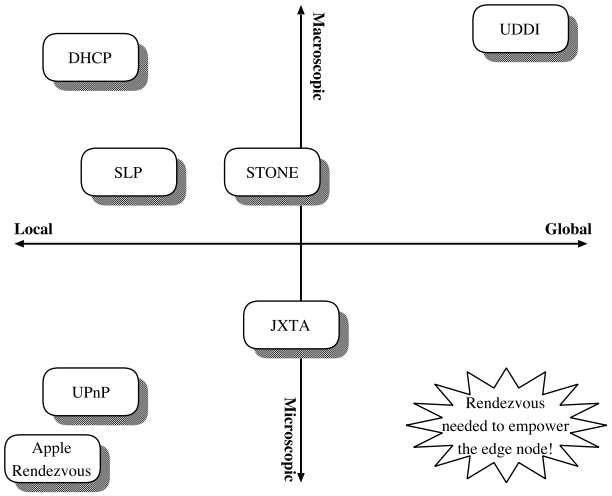

Fig. 5.1. Segment map of rendezvous technology

rendezvous, candidates of rendezvous are limited to nodes in the same domain with rendezvous user. The domain means local segment or local administrative domain.

The y-axis in the figure represents micro-macro attribute. Macroscopic rendezvous mean server/service centric rendezvous. Usually, server/service does not plug and unplug as regular client does, or many nodes could form a service for redundancy. Target of macroscopic rendezvous is service, not a node.

Microscopic rendezvous is a node centric method. What a user wants is the printer in that room, the sensor node on the coffee pot there, or just the PC ahead of the user.

As the map tells, we have global-scale technology like UDDI and microscopic technology like Service Location Protocol (SLP)[93], Universal Plug and Play (UPnP)[303], and Rendezvous technology by Apple Computer Inc.

The map also tells there is no major rendezvous technology to accomplish global-microscopic rendezvous. Actually, to empower the edge nodes in ubiquitous computing environment, technology in the segment is needed. The next section describes why global-microscopic rendezvous is needed.

5.1.2 Rise of edge

Legacy network communication is based on virtual-virtual communication. A manager puts their content on a virtual space (e.g. a web server) and virtualized people (physical location is not relevant in communication) through a WWW browser watch them.

Rise of edge is both capacity and capability improvements on the edge nodes placed in user's space, not in server room. Especially in semantics of ubiquitous computing, edge nodes like displays, sensors, and web-cams have interface between real space circumstances. They even have ability to control real space attributes like lighting controller, audio/video system, or air-conditioner. These nodes are bridge between physical world and virtual world.

With the power of the bridges, we are able to feel and control physical from virtual communication. To enable the power to the most extent, a user needs the ability to select a set of nodes out of trillions of edge nodes, to accomplish user's objectives.

For example, a user may want to find specific temperature sensor in a room at distant location and foreign administrative domain. If local rendezvous system is used, the user's query does not go over one administrative domain.

Thus, global-microscopic rendezvous is needed. Legacy and recent ubiquitous systems like STONE[188] tend to focus on micro-local rendezvous. Their focus is in single domain like a university campus. Actually, we need to interconnect very far edge nodes as same as local edge nodes.

5.1.3 Rendezvous

In this report, the author defines *rendezvous* as the following:

Definition of rendezvous:
The process to find a set of nodes k out of universal set U. Node characteristics list c distinguishes k from $U - k$.

The definition tells that rendezvous process occurs around node characteristics. Node characteristics include interface, class, location, time-location, and keyword. In practice, characteristics must be represented as some data, like arbitrary string, value, object, or document. In this report, the author assumes that a characteristic is represented as a key string. For example, printer should be represented as "printer". Namespace management of the key string is not discussed in this report and the author assumes everyone shares a de-facto dictionary space.

Interface based rendezvous is the process to find nodes with data I/O capability of certain type. Exported interface object class definition of remote method invocation would be a good characteristic for the rendezvous. Prim-

itive type of such rendezvous is MIME file type like `text/html` or `image/jpeg`, but MIME file type alone does not tell how the I/O will be handled.

Characteristics of class based rendezvous is node service class like "Printer" or product class like "LBP-..." Those values will be set in the factory.

Location would be another characteristic. For example, free string like "Building #60 Third Floor" and lat-long combination like "N140.330E38.994" would be good candidate representing location characteristics. Natural extension to location-based rendezvous is time-location represents an activity in certain place. A conference would be represented as "2003/10/01 10:00-16:00 @ Conference Room B"

In addition, free keyword based rendezvous should be available to accommodate new and innovative application. Like file share rendezvous, streaming radio listener rendezvous, and so on.

5.1.4 Related Research

Work of Reynolds and Vahdat[238] makes efficient keyword searching over DHT. Their method to produce intersection of keyword-bounded table using bloom filter is close to method described in this report.

Contribution of this report's approach on bloom filter technology is to control false positive rate. A false positive is not failure in rendezvous case described in section 5.2. It results in excessive traffic only. Thus, certain amount of false positives is acceptable and this approach does not recheck result by transferring actual data between DHT nodes.

In addition, as this approach does not need transfer of actual content, parallel query among three or more target keyword/peer group is done in simple way. Query of $A \cap B \cap C$ could be separated to query of $X = A \cap B$ and $Y = A \cap C$, on a node that holds set A. Query for X and Y is independent and simple parallel strategy works.

The proposed rendezvous method is something similar to INS/Twine[9]. The major difference is handling of complex queries. In their work, a resource is described in tree and the tree is split into *strands*. The strands resolved in parallel thus a complex query make a fair amount of traffic. In this report's method, an infrastructure node takes care to make efficient intersection between each peer group (corresponds to each strands) and traffic between each infrastructure node would be reduced.

5.2 Rendezvous with peer groups

There are many characteristics to be a rendezvous point (key) as section 5.1.3 describes. To support all of them, infrastructure systems should have free characteristics notation. In addition, a set of node should be able to be bound on each characteristic. A set of nodes for a characteristic consists of a peer group.

Thus, peer group concept is a natural solution to support such a free rendezvous, as peer group concept itself does not introduce any model like service class, interface, and location.

In this section, the author proposes a model of peer group with i3. An extension is proposed to enable flexible set operation on the infrastructure.

5.2.1 Internet Indirection Infrastructure

Internet Indirection Infrastructure[270] is an infrastructure to enable indirect communication among Internet nodes. The base of the infrastructure is DHT like Chord[271]. And each hashed key represents a "indirect socket" for one-way communication.

A listener node that wants to listen to certain key inserts the node's contact like IP address/port pair on the key. A sender node that wants to send a message to the key injects the message into a node of the infrastructure. With algorithm of DHT, the message is routed to the DHT node that is authoritative on the key. Finally, the authoritative DHT node sends the message to the listener node.

There may be more than one listener node at the same key. In the case the key is multicast key

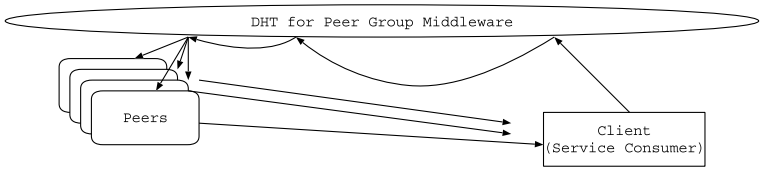

Fig. 5.2. A model of peer group rendezvous

and the message sent to the key duplicated and forwarded to all the listener nodes.

5.2.2 Proposed peer group rendezvous with DHT

On rendezvous process, a seeker node (client) wants to find a set of nodes with specific set of characteristics. After finding, the client node would want to send query to the set to find best node to use.

One characteristic for one peer group approach is natural approach to make such a rendezvous. The client wants complex resource and issues complex query like `X and Y and Z`. Each element like X in the query must be atomic, and atomic element in this case is a characteristic.

In this report, the author proposes a limited set operation, get intersection, as operator to make a complex query. Complex queries may be processed in various ways. In this case, each element is list of node identifier or contact. Handling each list as set and applying set operation is simple and effective way to solve the query. For example, the query `X and Y and Z` is converted into $X \cap Y \cap Z$.

The Fig. 5.2 represents simplest peer group rendezvous. The seeker node gets c because the infrastructure knows X, Y, and Z with DHT, and can get intersection among them. Actually, the infrastructure forwards the given message to peers in the intersection, and those peers can contact the client to serve.

5.3 Simple set operation on peer group

To enable flexible rendezvous mentioned above, the author proposes a method to make get-interaction operation over DHT nodes. The operation protocol accepts allowable errors as tradeoff and reduces traffic needed for the operation.

5.3.1 Get-intersection operation and bloom filter

This method contributes to traffic reduction on get-intersection ($X \cap Y$) operation. For rendezvous purpose, make smaller and finer grained set as k is needed. Thus, intersection is more important than union.

There are local and remote peer groups to be handled. Get-intersection operation is performed on a node in the DHT. Request for the operation is routed to DHT node that holds one of groups under the operation. The selected group is on the node and this is the local peer group. Others are remote peer groups because they are on remote nodes over DHT in most cases.

Sending and fetching whole peer group data to get intersection among local and remote peer group costs too much. With get-intersection operation, the entries wanted are in the local peer group. Thus, the local node does not need whole remote group. It just needs to know if each entry is in the remote group.

The operation is accomplished using exchange of bloom filter[18], without sending any of entries in the group. Bloom filter is a method to find if a data is in the data set or not, with possibility of false positives.

5.3.2 Bloom filter and false positives

A bloom filter is array of bits created with arbitrary length L, and a set of entry is registered to the filter. With a filter with the registered set, an entry can be tested against the filter. The test returns positive or negative as result.

Bloom filter test suffers from false-positives. If a test returns negative result, there is no chance to have the tested entry in the registered set. On the other hand, positive result of a test does not guarantee the entry in the registered set, due to hash value conflict used in the filter algorithm. False positive rate (FPR) is the probability to get positive result on testing an entry not in the registered set.

In case of false positives, excessive traffic is produced. The infrastructure forwards the message sent from client to peers that belongs to intersection among groups in query. In the case, a false positive means a message forwarded to an unexpected peer.

Low traffic is outcome of bloom filter based get-intersection operation. Thus, expected false positive rate should be balanced against message size. Detailed analysis of excessive traffic due to false positives is described in section 5.4.2.

The following equations tell the false positive rate under certain condition[18].

- $f = s^p$ and $L = \frac{p \cdot M}{-(\log s)}$
  - f: False positive rate (FPR)
  - L: length of the filter
  - M: number of elements in candidate set. The candidate set is set of elements that may be tested against the filter.
  - p: phase, number of independent hash function used to select a bit in the filter.
  - s: saturation. rate of 1 (true) among all filter bits in case of all of candidate elements registered. The literature[18] tells a filter is most effective if $s = 0.5$

Thus, to define appropriate length of L in the best efficiency ($s = 0.5$), we need to know required FPR (f or p as s is static) and M.

5.3.3 Protocol of get-intersection operation

Get-intersection operation for the query $A \cap B$ is processed between a server (SA) that holds set A and a server (SB) that holds set B is as the following. A query is processed on the node that holds the first group in the query. In this case, set A is the local peer group and set B is the remote peer group, and SA is local node and SB is remote node. Note that SA can find address of SB from B's hash value with DHT mechanism.

1. at SA: sends number of entries of set A ($|A|$), and required phase p, to SB.
2. at SB: calculate bloom filter of B from $M = |A| + |B|$ and given p. Return it to SA.
3. at SA: tests each entry of set A on the filter, and assume entries with positive result as the intersection.

The trick in the protocol is control of FPR. SA sends number of entries to SB. Thus, SB can use $|A| + |B|$ as parameter M.

5.4 Proof of concept

This section describes a prototype to investigate correctness of concept. In addition, basic analysis of the proposed model and set operation is described to support the investigation.

5.4.1 The prototype

The base of the prototype is Chord-like DHT with message forwarding on and get-intersection add-on. It is written in about 3500 lines of Python code. It uses TCP for all communication. A Chord node uses only 159 finger connections per node at most, thus the prototype keeps connection open for each finger node. The query resolution process is iterative. If the answering node knows better DHT node to ask, it returns redirection message. The query issuer retries on the new node and repeats the process until the query resolves. The wire-format of protocol is serialized form of Python language object for now. Thus object serialization overhead exists.

Each entry in the DHT is an array. Inserting something on the entry appends the value on the array. The message-forwarding module to listener node works if the listener node inserts data with *handler format*.

The format consists of format ID string and endpoint tuple. Endpoint tuple is host endpoint ID like DNS name or IPv4/v6 address, and port

number with 32-bit integer.

A client resolver may send get-intersection query. Query has a target hash ID, list of ID, and requested p as the payload. The query is routed to authoritative ode of target hash ID. The node tries the get-intersection protocol described in section 5.3.3 for each ID in the ID list, and return composed result (list of handlers) to the client resolver as the requested intersection.

5.4.2 Evaluation of get-intersection operation

In this section, the author evaluates traffic amount related to the method proposed.

Filter length As this method involves an integer (size of local set), an identifier (remote set identifier), and bloom filter to find intersection, the length of the filter itself dominates the whole traffic volume.

Optimum filter length itself can be derived as we get M and p. According to section 5.3.2, optimum L against M is shown in Fig. 5.3.

In the figure, there are three lines corresponds to less than 1 percent ($p = 6,\ f = 7.8e{-}3$) false positive rate (FPR) case, less than 0.1 percent ($p = 10$, $f = 9.8e{-}4$) FPR case, and less than 0.01 percent($p = 14$, $f = 6.1e{-}5$) FPR case. Two horizontal lines are 512 bytes (guaranteed payload size of single UDP datagram) and 1460 bytes (common size of largest datagram in Ethernet, without fragmentation). According to the equation and the figure, a 512-byte UDP message can carry at least filter for about 200 candidate elements with 0.012 ($200 \times 6.1e{-}5$) expected false positives or at most filter for about 440 candidate elements with 3.4 ($440 \times 7.8e{-}3$) expected false positives.

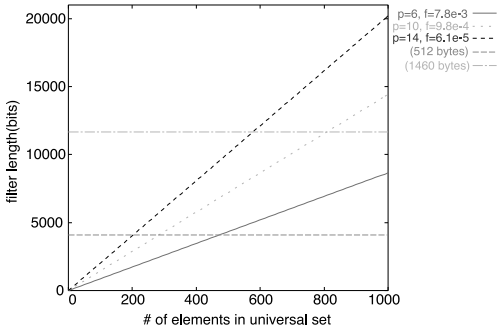

Fig. 5.3. Relation of filter length and number of elements in candidate set

Another evaluation is on M and size balance between left group and right group of the intersection operator. As equation in section 5.3.2 tells, the filter length L is derived from p and M. But there is no consideration on the balance of left/right. If left group has 999 records and right group only have 1 record, sending the whole right group back to the left is cheaper than sending back the filter with $M = 1000$, spends 181 bytes per p, if the record is small enough.

Let length of a record be r, and the following equation tells when filter costs more than sending back the whole group.

$$m = p \cdot \frac{1}{-(\log 0.5) \cdot r}$$

m is efficiency ratio of the bloom filter. If m is 1, the filter transfer and whole group transfer are of the same cost. In contrast, if m is 0.01, the filter transfer costs as much as one percent of whole group transfer. For example, if $r = 160$, one phase is worth 0.9 percent of whole group. If the right group only have 5 percent of whole set, sending bloom filter with $p \geq 6$ costs more than sending back whole group.

Excessive traffic amount due to false positives: As an intersection of groups is target of a message sent from a client, a false positive results in an excessive message forwarded. As the peer selected by false positive may just ignore the message, drawback of false positive is just the message transfer cost.

Let the message size be m and size of local peer group be M_l. Then, expected excessive traffic caused by a get-intersection operation (T_e) is $T_e = s^p \cdot M_l \cdot m$. At the same time, L is decided on p and M. Thus, as m, M_l, and M given, a node can find the best p with the lowest transfer amount ($L + T_e$).

The protocol proposed in the section 5.3.3

would be extended to let a node to find the best p. To make that, the required values must be on the node. As a local node sends parameters to remote node, the remote node is the appropriate node to find the best p. To help it, m should be given along with M_l to the remote node.

Traffic volume and latency in practice: Traffic volume and number of messages are measured with the prototype, described in section 5.4.1. Test preparation procedure is as the following. First, it sets up the DHT nodes and waits 30 seconds for the finger information to be initialized. After the boot, a set of random keys and values are created, series of combinations of $(key, value)$ are selected out of the keys and the values, and the combinations are inserted to the DHT nodes. In the following test scenario, there are 500 peers and 100 groups. And a peer joins 10 groups at random.

In the prototype, a value corresponds to a peer and a key corresponds to a group. A key holds one or more values as a group holds one or more peers. A combination of $(key, value)$ corresponds to a peer (bound to the value) joins to a group (bound to the key).

A test client makes get-intersection in succession among two and four groups. The operation to get all the groups to make intersection on the client side, get-all operation in short, at the same condition is also measured for comparison.

Table 5.1 describes traffic volume involved in get intersection operation (get-i) among two/four groups, and the operation among four groups per operation (average of 1000 operation), against

Table 5.1. Traffic Volume among the Whole DHT(bytes)

#Groups	Two		Four	
	get i	get all	get i	get all
$N = 5$	599	1177	983	2376
$N = 8$	622	1491	1275	2887
$N = 10$	812	1760	1413	3483

Table 5.2. Number of messages among the whole DHT

#Groups	Two		Four	
	get i	get all	get i	get all
$N = 5$	11.3	11.5	20.8	23.4
$N = 8$	12.8	12.5	24.1	25.2
$N = 10$	16.0	16.0	24.4	31.6

5 nodes, 8 nodes, and 10 nodes. For comparison, result of get operation on all the groups (get-all) is also shown.

The table tells that (1) the system looks to scale better than $O(N)$ because sum of bytes on $N = 10$ is less than double of sum of $N = 5$, (2) get-intersection consume less traffic. Especially for four group case get-intersection consumes 42 per cent of get-all, in contrast to two group case that consumes 45 per cent of get-all.

The table 5.2 tells average number of message exchange per operation for two groups intersection, four groups intersection and both get-all on the same condition above. We can estimate expected latency of each operation by expected RTT among each nodes and the client. If we estimate average RTT as 200 milliseconds, get-intersection among ten nodes ($N = 10$) and four groups (#Groups=four) may consume more than 3 seconds. Parallel query would help to make it shorter.

5.5 Discussions

5.5.1 Naming of groups

In this report there is no assumption and definition on name of group. This will cause confusion on name space.

For example, consider printer group. Due to latitude of group name, you can make printer group in English, Greek, Dutch, or any other language with every kind of encoding like iso-2022, iso-8859, Unicode, and others. More confusing example is many encoding for one language. In Japanese, there are iso-2022-jp (JIS), EUC-JP, Shift-JIS, and Unicode. DHT distinguishes every

group named identically and encoded differently as independent groups.

The author considers two approaches to ease the confusion.

The first approach is to define precise naming convention by some de-facto or standardization process. This is simple and robust approach but some application does not wait until such namespace is built.

The second approach is some sort of collaborative filtering to find similar groups, with or without external knowledge. If DHT node can assume group α and β is similar, it can automatically include reference to the other group when answering request. Then, client could select larger group or just ask both (or all, if there are more than three). More advanced approach includes automatic integration of very similar group. If group β is similar with α, DHT infrastructure can eliminate β and all request on the group is forwarded to α.

Another consideration is name conflict. If many incompatible instant messaging application, online game and other application use the same name like "Beginner's Room" or "Help Desk," name conflict leads up to disaster.

A solution is to mandate namespace tag in the group name. A new application which does not rely on other application's namespace must put a pseudo-random data and/or time-stamp ahead of the name. For example, "104956914.f96f;Beginner's Room" for the application that owns prefix "104956914.f96f;"

5.5.2 Handling huge group

DHT is a novel method to realize fully-decentralized data table. Actually, characteristics of DHT are similar to hash table. It has similar limitation on data handling with regular hash table.

The peer group rendezvous method proposed by this report assigns a hash entry for a group. The hash entry keeps a set of peers as single list. Thus, this method does not scale against group size.

The scalability against huge group may be beyond scope of DHT. Thus the method must be integrated with other huge list handling method. An approach for it is to set upper limit of number of nodes on each group. The next node after the limit would replace a node in the group with LRU if possible. Another approach is making subgroup of huge group, with finer grained granularity. For example, "color printer" and "monochrome printer" would be a good set of candidates as subgroup of "printer" group. Let "printer" group be abandoned and move all the printers into appropriate subgroup. The abandoned super group only includes reference to those subgroups.

5.5.3 Security consideration

DHT needs to be secured to be used for real application. As the proposed method depends heavily on DHT, threats on DHT are threats on the method.

Apparently, at least the following three denials of service threats exist.

1. Spamming on DHT entries
2. Malicious DHT node
3. Adding self to unrelated group to gain information

If a peer group got DoS attack, the group become useless because only junk would be found, or more simple case no peer takes care of the group, during rendezvous process.

Details of each attack and defense are beyond the scope of this report and not described here.

5.6 Conclusion

This report proposes a design of a simple peer group rendezvous process. Introducing intersection between peer groups, the method can support flexible resource finding and rendezvous.

Proposed rendezvous method is based on distributed hash table (DHT) technology. Therefore, the sets contain peer group membership information is completely distributed and cost of communication to make set operation

(get-intersection) is the key factor of the method's efficiency.

To reduce the cost, this report also proposes get-intersection protocol that sends a bloom filter to find out the intersection. The length of bloom filter is controlled by the whole size of those two groups to get intersection. Thus, there is no need to verify the result as long as the user accepts the false positives under certain rate.

The author also points out some difficulties to deploy proposed peer group rendezvous method. Security, naming, and group size issue should be solved as the next step.

第 6 章　Uniform Rendezvous Pointer
オーバーレイネットワーク間相互参照のための識別子空間

Abstract

As the first step towards multi thin overlays network architecture, we propose URP (Uniform Rendezvous Pointer). URP is the identifier space to point resources among different overlay networks. We discuss why we need URP and required functionalities for it. Especially, we evaluate some topics like policies, ability of verification, and so on.

The most significant difference between regular URL/URI is that URP does not just identify a resource. Instead, URP is notation how a client should try search among overlay and how the result would be verified and treated. Because resources on some overlay like Freenet and Gnutella are unstable and have no identity, just identifier nor locator would sufficient in inter-overlay.

URP would be efficient glue between i-WAT overlay network and other collaborative overlay network. With URP, overlay users can exchange notes using i-WAT and other barter overlay network as rewards of collaboration. In other words, other application and overlays may use i-WAT network as a market and URP is used to link both transactions in the application network and i-WAT network.

With this article we have discussed what is requirement for URP. As the next step it should be utilized in real application like i-WAT and other networks to step towards multi thin overlays network architecture.

6.1 背景

インターネットの普及とともに、その利用法の幅は広がりつづけている。特に、IP アドレスで表現できない資源を識別・表現するために、オーバーレイネットワーク（以下オーバーレイ）等さまざまな方法が考案されている。一方で、それらの協調動作のための枠組がほとんど存在しないか、存在したとしても限定的な解である、という問題がある。

本章では、インターネットを用いて「統合分散環境」を実現するために、これらオーバーレイ同士を相互に接続する必要性について論じる。

6.1.1 「統合分散環境」に向けて

統合分散環境を、以下の 2 つの条件が成立する環境と定義する。

1. **分散**: 全世界を継ぎ目無く統合するネットワークが存在し、その上に無数の資源が存在する。
2. **統合**: 存在する無数の資源が、人間の目的達成を補助するために有機的に統合され、機能を提供する。

条件 1 は、IPv6 やネットワークアプライアンス等の普及により達成されると考えられる。一方で、条件 2 は、ネットワークを構成する資源の数や種類が多くなるにつれて困難になる。

条件 2 を達成するために、識別子による資源の表現能力に注目する。現在の Internet で多く用いられている識別子は、IP アドレス+ポート番号、そして URL である。IP アドレスはネットワークインターフェイスの識別子であり、これにポート番号を付ける事によってそのインターフェイスを持つ計算機上に存在するプロセスを指定する。同様に、URL は特定のプロセスが管理するファイルの識別子である。

これらの識別子は、「計算機上のプロセス」という粒度に縛られている。あらゆる資源を表現するために、その資源を管理するプロセスを指定する必要がある。一方、「実世界の人間」や「地理的な位置」等、

特定のプロセスに強く関連づけられない種類の資源が次々とネットワークに登場しつつある。

また、必要な資源を効率的に発見するためには、資源の特徴的性質を記述できる識別子空間が必要となる。その空間は一意に定義できるものではなく、発見の目的に応じて異なる空間が設計される。実際、単純な名前、資源の性質やサービスインターフェイス、セキュリティポリシーや匿名性等、目的に応じて異なるシステムが作られている。そして、システムの数だけ識別子空間は存在し、互換性は無い。

6.1.2 オーバーレイ

オーバーレイは、多様な識別子空間を実現するための一つの手段である。IP を基礎にしたインターネット上に、アプリケーションが必要とする識別子空間を実現し、その上で資源への到達性を実現する。

各種インスタントメッセージングシステム（Jabber[141] や ICQ[109] 等）や IRC[145] 等は、人やプレゼンスといった資源の表現のために発達している。一方、Tenbin[347] 等は、コンテンツという資源を扱う。Chord[271] 等の分散ハッシュテーブル（DHT）は、ハッシュ値という抽象的な資源の表現を扱う。他、Freenet[39] 等、IP ノードに対する匿名性の適用を主目的としたものなどもある。なお、本章ではオーバーレイ上のノードと区別するため、インターネット層におけるノードを IP ノードと表記する。

目的の数だけ資源の抽象化の手法は異なり、その結果多数のシステムが生まれている。本章では、これらのシステム全てを「オーバーレイ」という枠組で捉える。オーバーレイは、IP アドレスで表現できない資源を抽象化して取り扱い、資源への通信機能を提供する。

6.1.3 オーバーレイ相互接続の要求

オーバーレイ上に抽象化された資源は、ネットワークを経由して利用可能である。これを抽象ノードと呼ぶ。しかし、抽象ノードとして利用できるだけでは統合分散環境には不十分であり、これら資源を統合する（6.1.1 項条件 2）必要がある。ネットワークの価値を高めるには、目的達成のために最適な資源を自由かつ創造的に選び取り、要求を満たす資源の組み合わせを選択する必要がある。この資源選択の自由を提供する機能を「自由で創造的なランデブー」と呼ぶ。

現状では、異なるオーバーレイ上の資源を組み合わせて活用する事には困難がある。一方で、「自由で創造的なランデブー」はこの困難を乗り越えた先にある。また、資源の選択がオーバーレイの内部に閉じていると、「全ての要求を満たすオーバーレイ」を作らなければ目的は満たされない。複雑な目的の達成に向けて、単一の目的で作られたオーバーレイをモジュール的に組み合わせる事で、自由かつ創造的な資源の組み合わせを実現でき、かつ価値創造がオーバーレイの内部に閉じないので、自発的に多数の応用が生まれるものと考えている（6.3.5 項）。

オーバーレイをまたがる抽象ノードの統合的利用に向けた最初のステップとして、様々なオーバーレイ上の抽象ノードを指し示すための識別子空間、Uniform Rendezvous Pointer（URP）を定義する。

6.2 複数オーバーレイの統合に向けて

本節では、まず複数オーバーレイを統合する事により得られる具体的なシナリオについて紹介する。その上で、複数オーバーレイ利用のモデルを定義し、モデル中に存在する資源を一意に識別するために必要な機能を考える。

URP に対する要求は、複数のオーバーレイにまたがる文脈において、それぞれのオーバーレイにある資源を区別し、指定できる事とする。

6.2.1 複数オーバーレイのモデル

複数の異なるオーバーレイを利用するモデルを考える。

オーバーレイには、様々な手法で抽象化された資源が独立・並列に存在する。

資源は一意に指定できない場合がある。例えば Freenet では、検索を開始する場所によって得られる内容が異なる。この場合、資源を示す代わりに資源の探し方を指定する他ない。

オーバーレイは、IP ノード同士が結合して構成する。この構成方法にはいくつかのパターンが認められる。

1. 既定の初期 IP ノード（well-known bootstrap nodes）のみで情報共有し、他の IP ノードはクライアントとして構成
2. 既定の初期 IP ノードを中心に自動構成
3. 近隣の IP ノードとの自動構成
4. システム外の経路（Out of Band）によって隣

接 IP ノードを手動構成

5. DNS 等のディレクトリサービスへの依存により構成

手法 1 および 2 は、既定の初期 IP ノードをネットワーク構成の足掛かりとする。参加する全ての IP ノードは既定の初期 IP ノードを知っており、それら IP ノードの利用が前提となっている。

手法 3 は、リンク層ブロードキャスト等を利用した自動構成手法である。Apple Rendezvous ではこの手法を利用している。

手法 4 は、一部の Gnutella 由来のシステムで利用される手法である。IRC や HTTP によって初期 IP ノードを別途取得してネットワークを構成する。

構成方法によって、全世界で共有されるネットワークが一つできる（手法 1、2、5）のか、それとも特定の初期 IP ノードや物理リンクに関連づけられた独立したオーバーレイが多数発生する（手法 2、3）のか、という違いがある。本章では、前者をユニバーサルネットワーク、後者をコミュニティネットワークと呼ぶ。また、コミュニティネットワークにおけるそれぞれの断絶したオーバーレイをコミュニティと呼ぶ。

コミュニティやオーバーレイは接続者にそれぞれ異なる要求を行う可能性がある。具体的には、ただ乗り（free riding）を防ぐために何らかの帯域制御を行う、サービスに対価を要求する、認証を要求する、あるいは匿名性を保証するなどといったポリシーである。同一のオーバーレイに複数のポリシーが混在する事も有り得るし、ポリシー毎に異なるコミュニティを形成する場合もある。

当然、ネットワークサービスである以上、共通のプロトコルを利用する事は大前提となる。

以上、本章では、オーバーレイを以下の 4 要素に整理したモデルを考察の基礎とする。

1. 抽象ノードの抽象
2. ネットワーク構成/コミュニティ
3. ポリシー
4. プロトコル

6.2.2 資源指定方法の考察

本項では、前項で述べたモデルにおいて資源を指定するための方法について、どのような場面でどのような指定方法が必要になるか考察する。

6.2.2.1 アクセス手段の指定

URP は、多様なオーバーレイ上に抽象化された抽象ノードを指し示す。抽象ノード利用には指し示すだけでは不十分であり、抽象ノードに対してどのプロトコルを利用するか指定できる必要がある。

従来の URI 式の数文字のプロトコル識別子（`chord`、`freenet` 等）による指定方式の他に、より精密にプロトコルを示す指摘方式が必要な場合がある。例えば、Jini の様に、抽象ノードにアクセスするためのスタブクラスの定義により資源を抽象化する場合等がある。

6.2.2.2 ポリシー情報の提供

ネットワークへの参加の際、アイデンティティを示す必要や、資源の提供を求められる場合等が考えられる。

このような場合、IP ノードはポリシーに基づいて振舞うべきなのか知る事により、事前に資源を用意できる。また、ポリシーに従えないのであれば、無駄なアクセスのための時間・帯域の消費がなくなる。

6.2.2.3 コミュニティの指定

例えば、オンラインゲームのためのオーバーレイは、規模性やプレイヤーの快適性等を考えて、複数の「ワールド（＝コミュニティ）」を持つ場合がある。

このような状況下で抽象ノードを識別するには、抽象ノードが属するコミュニティを指定する必要がある。コミュニティを示すには、初期 IP ノードの IP アドレスを示す等の方法がある。

6.2.2.4 依存／外部参照の指定

オーバーレイは他の情報に密に結合している場合がある。例えば、特定の地域に存在する抽象ノード群を抽象化したオーバーレイを考えた場合、地域を指し示す記述から正しいコミュニティを選択する事になる。このような場合、初期 IP ノードを外部のディレクトリサービス、例えば DNS や DHT などを利用し発見する。そのための手掛りを URP に含められると良い。

6.2.2.5 資源成立条件の限定

あるオーバーレイにおいて、時間経過した後もその内部の資源が有効である保証は無いし、オーバーレイのどの IP ノードから資源探索しても同じ資源

に到達する保証も無い。

資源の有効性や同一性を確認するためには、資源が意味を持つための条件を示す方法がある。識別子に有効期限を与え、その期間のみ有効性を保証する方法や、MD5 チェックサムによって内容の同一性を確認する方法などが考えられる。

6.2.2.6 ユーザ識別子の記述

特定のオーバーレイに対して、「何者として」アクセスするのかを指示する識別子は、アクセス制御を簡略化できる。

6.2.2.7 Uniform, not Universal

URP は、全世界（Universal）の資源を統一的に表現するものではない。ある特定の環境においてのみ意味を持つ識別子、文脈に応じて示す資源が変わる識別子が存在する事を許す。

一方で、URP はどのような環境においても画一的（Uniform）に生成・解釈できる必要がある。

6.3 提案する URP

ここまでの議論を受けて、複数のオーバーレイの中で抽象ノードを識別し、かつオーバーレイの多様さを柔軟に吸収できるポインタを提案する。

6.3.1 URP により示されるもの

6.2.2.5 項での議論にあるように、オーバーレイ上の抽象ノードは一意に特定できる場合もあれば、そうでない場合もある。従って、URP は抽象ノードの位置を直接示す事を前提にできない。この制約下で抽象ノードを示すために、抽象ノードの探索方法を示すという方法が利用できるべきである。

従って URP は、「探索方法」を示す事により抽象ノードを指示する識別子である、と言える。

6.3.2 要素の検討

ここまでの議論に登場した、オーバーレイの要素を整理すると以下のようになる。

「何を利用して探索するか」:

- `accessor`: プロトコル識別子

「どこを探索するか」:

- `community`: 初期 IP ノード／外部参照

「何を探索するのか」:

- `resource_id`: 実際の資源を識別する記述

「どのように探索するか」:

- `identity`: アクセス者のアイデンティティ
- `condition`: 資源探索成立の条件
- `preference`: ポリシー要求等

資源を同定するため、最低限、`accessor`、`community`、`resource_id` が必要である。「どのように探索するか」は必要に応じて利用すれば良い。

6.3.3 表記案

URL の表記方法に則り、基本形は以下のように提案する。

```
{accessor}://{community}/{resource_id}
```

また、全ての要素を記述するために、以下の表記を提案する。

```
{accessor}://{identity}@{community}
  ?{condition}!{preference}/{resource_id}
```

（改行は含まない）

URP の各項目について以下に述べる。

「何を利用して探索するか」: `accessor` は、資源へのアクセス方法を規定する。アクセス方法の規定には (1) 明文化されたプロトコルによるアクセス (2) 抽象クラス定義によるアクセスの 2 通りが考えられる。

明文化されたプロトコルによるアクセスへの `accessor` は、従来の URL における `http` あるいは `ftp` 等の scheme と同様、短い識別子による記述を行う（例: `chord` `can` `iwat` 等）。

一方、「抽象クラス定義によるアクセス」は、未知ないしアクセス方法が一定しない資源への `accessor` を提供する。URP によって表現される資源の利用法はある程度抽象化できる可能性がある。例えば、`get` や `put` 等の基本的なデータ操作がある。このような抽象化されたデータ操作を、抽象クラスとして共有した上で、資源への `accessor` としての具象クラスをネットワーク経由で供給できる。抽象クラスの定義については 6.4.6 項で議論する。

「どこを探索するか」: `community` は、オーバーレイに接続するために必要な情報である。内容はオーバーレイの構成方法（6.2.1 項参照）によって異なる。既定の初期 IP ノードを持つ場合は空欄で良いし、DNS 等の外部参照を利用する場合はその参照に必要な式ないし名前を記述する。

「何を探索するか」: 各オーバーレイ内部で、資源を示す識別子である。オーバーレイに対応して、抽

象ノードの識別子、ファイル名、問い合わせ文字列等が書かれる。

「どのように探索するか」: identity は、{userid}:{credential}、あるいは{userid}のみになる。userid は各オーバーレイ固有のユーザ識別子である。credential は、識別子に対応づけられた秘密である。

例えば、ファイルの MD5 チェックサムを condition として URP に含める場合は、md5sum=16136a7.... のように記述する。

また、URP による資源探索プロセスに対価（地域通貨・資源等）が要求される場合がある。それら資源をどこまで提供するかここに記述できる。例えば、利用する通貨単位が mh（Man Hour）だった場合、condition に cost<0.4mh として記述する事で、URP による資源探索は 0.4 mh 以下で終了しなければならない（さもなければ中断）、という記述が可能になる。

preference には、資源を探索する際、資源利用者側に選択肢がある事柄について記述する。例えば匿名性が実現可能なネットワークでは、利用者が匿名であるか無いかを選択できる。探索方法を指定する URP においてこういった選択肢を記述する事は、簡易な資源利用を促すことができる。

また、自発的な資源の提供を意味する記述も preference に適当だろう。

6.3.4 提案する URP によるモデルの充足

6.2.1 項で述べた複数オーバーレイモデルに対して、提案する URP は必要な資源を指し示せるかを考察する。

まず、オーバーレイの構成手法（モデル要素 2）について考える。ユニバーサルネットワークではオーバーレイの構成のために必要な情報は既定のものとして与えられるため、URP の記法に関わらずネットワークを構成可能である。一方、初期 IP ノードが必要なコミュニティネットワークでは、community より初期 IP ノードを伝達できる。

ネットワークのポリシーの記述（モデル要素 3）は、preference あるいは condition によって行う。preference は、計算資源の提供や匿名性など、確定的な意思を記述し、condition はファイルのチェックサムや支払える対価など、条件を記述する。

プロトコルの記述（モデル要素 4）は、accessor によって行う。accessor は同時に、抽象ノードの抽象（モデル要素 1）を定める。chord にとって、抽象ノードは単なるデータであるが、通貨取り引きのためのプロトコルを利用する場合は、通貨あるいは取り引きを行う者を抽象する。

しかし、accessor によって抽象ノードの抽象とプロトコルの両方を定めて良いかについては、議論の余地がある（6.4.6 項）。

6.3.5 具体的応用例

ファイル共有を行うオーバーレイは多数存在する。それらには、ファイルを共有するためのコスト（HDD 領域・通信帯域・CPU 時間・電気料金等）や独自のコンテンツの提供に対するインセンティブが乏しいという問題がある。このようなオーバーレイを、ここではファイル共有オーバーレイと呼ぶ。

インセンティブの問題を解決するために設計された Mojonation では、内部にファイル交換と対価の支払いの仕組みの両方を持つオーバーレイを構築した。ファイル公開や交換のための資源の提供に対して対価の支払いを求める事によって、より質の高い資源の提供を促進するという狙いであった。しかし、この統合されたシステムは普及に困難があり、オーバーレイとしては成功しなかった。結局、対価のやりとりをあるオーバーレイの内部のみで行う事により価値創造をその中に限定してしまい、インセンティブとして成り立たなかったものと考えられる。

これに対して、対価のやりとりを行うオーバーレイをファイル交換オーバーレイと結合させて利用するというアプローチが考えられる。例えば、対価の一つとして「手形」のような地域通貨を扱うオーバーレイ [250] が存在する。このようなオーバーレイをここでは対価交換オーバーレイと呼ぶ。

この 2 つのオーバーレイを統合して利用できれば、ファイル交換と対価の支払いを対応づけを行う事は容易である。例えば、先行して普及すると考えられるファイル交換オーバーレイに対して、対価交換オーバーレイを扱えるプラグインソフトを導入する。対価交換オーバーレイにおける相手の信用に関する問い合わせや価格の擦り合わせ等の情報を、ファイル交換オーバーレイ上で URP により交換する。URP によってこのプラグインソフトが呼び出され対価交換が成立する、といったシナリオが考えられる。

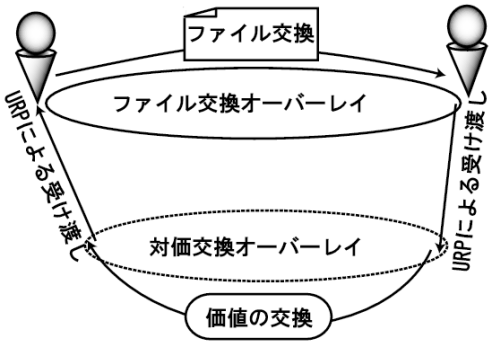

図 6.1. ファイル交換オーバーレイと対価交換オーバーレイの URP による統合（模式図）

図 6.1 は、このシナリオの模式図である。2 人の利用者が、ファイル交換オーバーレイと対価交換オーバーレイを同時に使用して、ファイルの交換と対価交換を URP によって対応づけている様子を示している。対価交換オーバーレイはファイル交換オーバーレイと独立に存在しているので、ここで交換された価値は URP によって広く再利用できる。すなわち、URP により価値創造の領域が広がり、交換される対価をインセンティブとして機能させる事が容易になる。

このように、単独のオーバーレイで全ての仕組みを実現するのではなく、自由で開かれた、創造的なランデブーのための空間を用意し、単純なオーバーレイをモジュールとして連携させる事により、応用が自己創出的に発展していく事が期待できる。

6.4 議論: オーバーレイ統合に向けて

URP により、オーバーレイ等の利用法、あるいはオーバーレイによる通信アーキテクチャにどのような影響があるか、以下に考察する。

6.4.1 利点: 記述の容易さ

URP により、オーバーレイへのアクセス方法の記述が容易になる。ユーザの目に見える範囲では、メールや口頭での通知等が容易に行える。実際のアクセスを行うプログラムがアクセス方法を判断すれば良いため、利用者に対する負荷が軽くなる。

6.4.2 URP の応用可能性

URP は複数のオーバーレイを連携させる事を目的として作られた。複数のオーバーレイによる連携には、6.3.5 項で述べたように、対価交換のような 2 者の直接即時的な連携が考えられる他、より大きな規模での連携を考える事もできる。

例えば、URP を収集する検索エンジンのクロウラー（収集プログラム）を考えてみよう。ある曖昧な検索式に応じて、具体的な「資源の探し方」のリストを提供する検索エンジンは、多様なオーバーレイの中に資源が抽象化されて存在する世界の利便性を確実に向上させる。

また、URP により、異なるオーバーレイ中に表現された抽象ノードの羅列を表現できる。具体的には、インスタントメッセージングにおける友人リストのようなものに、AIM、MSN Messenger、Jabber 等の利用者を URP によって表現し、格納でき、また単一のクライアントプログラム（あるいは他のプログラムを駆動するメタ・プログラム）に対して URP を渡す事で、適切なプロトコルでメッセージをやりとりできる。

このように、URP によって、オーバーレイに存在する「囲い込み」を乗り越え、オーバーレイ選択に自由と柔軟性を導入できる。筆者らは、この自由と柔軟性は、ネットワーク環境を「統合分散環境」に進めるために必須の性質の一つであると考えている。

6.4.3 議論:「秘密の共有」の崩壊

匿名掲示板などを「コミュニケーション」の切口で分析すると、そこには「秘密の共有」と「自己顕示」という二つの側面が出てくる。特に秘密の共有は、自己と相手の心理的な相互信頼を確立するための手段となっている。コミュニケーションを目的とするオーバーレイでも同様である。つまり、URP が指し示すような資源を発見する事に価値を見いだし、発見した者を「仲間」として迎え入れるような人的コミュニティの形があるかもしれない。

こういった人的コミュニティにとって、URP のような簡易なアクセス手段の記述は、共有される秘密へのアクセスを容易にし、相対的にその秘密の価値を下げるのだろうか。

また、URP により「検索」がオーバーレイに対しても可能になった場合の影響についても考察が必要だろう。Google 等の検索エンジンによって、個人的なページやそのリンク集を経由した情報の探索（俗にネットサーフと呼ばれていた方法）は減少した。検索エンジンは、URL によって無数の情報を発見し、検索者に提供できるようになった。その結果、WWW

上にひっそりと存在した人的コミュニティは一気に白日に晒され、その一部は公衆化する事を嫌い、あるいはそもそも非合法故に WWW の外（ファイル共有ソフト等）に潜った。

URP によって、多数のオーバーレイが現在の WWW と同様にアクセス可能になった場合、こういった人的コミュニティはどこに行くのか、どういったインパクトを与えるのか等も考える必要があるだろう。

6.4.4 普及への障害: 複数方式への対応

URP によって簡易に資源を指定できるようになっても、その資源が簡便に利用できるだけの機構が存在しなければ、URP によるメリットは乏しくなる。一方、将来いくら可能性があろうとも、現状でメリットが乏しいものを実装するインセンティブは低い。既存のフレームワークの URL 処理部への統合等、URP を利用する環境を発展的に成長させるための現実的な壁を壊していく手段を考える必要がある。

6.4.5 議論: URP 利用者の情報

`condition` や `preference` はあくまで利用者個々の事情であり、資源へのポインタの伝達手段である URP に含めるべきでは無いとする考え方もある。しかし、筆者は敢えて URP にこういった情報を入れるべきであると考える。なぜならばオーバーレイはあらゆる意味で不均一である可能性が高いからである。

例えば、同一のオーバーレイのプロトコルにおいて、匿名が「望まれる」場合と、実名が「望まれる」場合がある。実際に匿名で参加するか実名で参加するかは利用者が決めて良い事である一方、URP を伝える側がオーバーレイ側の希望を伝える経路があっても良い筈だ。これは匿名性に限らず、資源提供の緩いポリシーの記述にも役立つ。

一方、利用者側のクライアントプログラムには、`preference` および `condition` を評価し、場合によっては URP による資源探索をブロックしたり、これらのフィールドをユーザの要求に合うように書き変えたりする機構が必要になるだろう。これはブラウザにおける P3P[45] のようなものになるかもしれない。

6.4.6 議論: 抽象ノードの抽象と操作

オーバーレイにおける抽象ノードの抽象（クラス化）およびその抽象に対する操作（メソッド）については、さらなる議論が必要である。

オーバーレイにおいては、様々な資源が様々な軸によって抽象化され、提供される。これら抽象化された抽象ノードに対する操作にはどのようなものがあるか、オーバーレイによって異なる可能性がある。

一方、共通の資源の抽象クラスを導入する事で、URP における資源の扱いを画一化できる可能性がある。この場合、`accessor` としてプラグイン（あるいはスタブクラス）を用意する事で、ありとあらゆる資源にアクセスする機構をネットワーク側から取得できる事になる。但し、この方法は資源のあり方に制約をかける事になる。

また、URP に操作を記述すべきかという議論がある。URP によって資源と探索者をランデブーさせるが、ランデブーした後の操作は記述できなくても良いのか。URL の場合は、`GET` 要求なのか `POST` 要求なのか、あるいは `HEAD` 要求なのかは文脈で決定できたが、その前提を URP に対して仮定して良いのか。例えば、`accessor` として `chord.join://...` 等と書ける事によるメリットがあるかもしれない。

6.4.7 今後の課題

効果の検証が第一の課題である。そのためには、実際に URP を応用するシステムを作る事が望ましい。具体的には、6.3.5 項で述べた応用例のシステム試作を目指す。

また、並行して、URP の取り扱いに関する API 等、利用環境を定める作業が必要である。URL を扱うルーチンに取り入れるにはどうすべきか等も検討する予定である。

6.5 結論

本章では、目的別に生じたオーバーレイの自然な発展としての統合において、URP が必要となる理由について述べた。その上で、URP に必要かもしれないいくつかの機能を列挙し、求められる条件を満たすためにどの機能が必須であり、どの機能がオプショナルかを考察した。

当然、本章で述べた URP は、現状存在するオーバーレイに対する限定された考察に基づく以上、将来発生する全てのネットワークをカバーしているという保証は無い。一方で、Universal でないが Uniform であるポインタおよびその記述方法がオーバーレイ統合に寄与する事は本章で述べた。

本章で述べたような、実際に取り扱う様々な性質の資源を様々なオーバーレイの形で抽象化する、というアーキテクチャは未だに十分成立しているとは言えない。今後の研究課題として、オーバーレイをより積極的に活用したアーキテクチャを実際に作成し、「統合分散環境」としての評価を行っていく必要がある。