What Happened to Pastry

Andrew Herbert
Microsoft Research, Cambridge
aherbert@microsoft.com

ABSTRACT

This paper describes how Microsoft Research goes about the process of technology transfer, using the experience of transferring the MS Pastry Distributed Hash Table and its applications as an example.

Categories and Subject Descriptors

C.2.4 [**Computer-Communications networks**]: Distributed Systems—*Distributed applications*; C.2.2 [**Computer-Communications networks**]: Network Protocols—*Applications, Routing protocols*.

General Terms

Algorithms.

Keywords

Technology Transfer.

1. INTRODUCTION

A companion paper in this issue by Roy Levin describes the mission of Microsoft Research (MSR) and means by which technology transfer is accomplished. This article rounds out Levin's introduction with an account of the transfer of distributed information systems research from the Microsoft Research laboratory in Cambridge, England. It also gives a perspective on how the process works in practice for a geographically remote laboratory.

1.1 Technology transfer at a distance

Levin [9] describes the role of the Microsoft Program Management (MSRPM) group and TechFest as means for researchers to make connections to product groups. For the non-US laboratories distance is an additional factor since most Microsoft product development takes place in Redmond, WA. In some ways this can be an advantage – researchers in overseas laboratories can perhaps more easily explore contentious topics and investigate fundamental problems. Since many major conferences and workshops are held in the USA, adding on extra days to visit Redmond is a comfortable way for overseas researchers to remain in contact with product group and US research colleagues. Moreover when researchers from different laboratories attend the same international conference they will often arrange a Microsoft "birds of a feather" meeting for similar purposes.

Each overseas laboratory has a local program manager who works in cooperation with the Redmond MSRPM team. They introduce new researchers to likely relevant product groups on the researcher's first trip to the "mother ship", provide project management support for technology transfers in process and increasingly have their own small development teams to incubate research results into more robust prototypes and conduct field trials of new systems.

For overseas researchers, TechFest is an annual "gathering of the clan" to showcase new projects and meet colleagues from across Microsoft Research as a whole. Typically TechFest is followed by workshops for researchers with similar interests to update each other and members of product teams on current work.

1.2 Technology transfer in practice

Technology transfer takes many forms at Microsoft Research. The most direct is when a product development group adopts research software directly into the code base of a product. For reasons which become apparent in the rest of this article, it is rarely that simple. More often a long term relationship grows between Microsoft researchers and colleagues in product groups with research results being used to drive an ongoing dialogue. At different times the researchers may consult on business strategy, work on product architecture and design even product implementation (i.e., coding). Often different product groups can be candidates for technology transfer: in the area of operating systems and networking research possibilities include the Windows operating system, the MSN and Windows Live Internet services, the Xbox Live Online computer gaming system, Microsoft's mobile computing businesses, Microsoft's internal enterprise network and customer Internet services such as Windows Update and product download sites.

Software "proof of concept" demonstrations are an important part of the transfer process – they make research results tangible to product developers. Having gained the attention of a product group, demonstrations are often further developed to provide a robust prototype which lays the foundations for architecting the product version and explores how the research results interact with other product features. Prototype software may pass from the researchers to the product developers, although it is more common for the code to be re-written to conform to product engineering standards. A transfer is deemed complete when the product group takes full ownership for the code.

Sometimes researchers are engaged for longer, especially where the results being transferred have a major impact on the design and implementation of the product and where perhaps the technology is outside of the experience of the product teams. For Microsoft Research Cambridge the inclusion of support for generic functions and other aspects of functional programming into the Microsoft Common Language Runtime and the Visual Studio product and its associated languages was such a case. The cross-cutting nature of generic functions meant this work touched all aspects of the product, from language design, library design, compilers, code generators and run-time verifiers and user documentation. The research team worked with the product team on the product code base over two releases of the product to ensure the technology was successfully transferred and that the product team had established complete confidence and competence in taking the technology forwards.

Good timing is important. Product groups work to a formal development process characterized by "milestones". In the early stages of the development cycle product groups are open to discussion about new technologies and indeed often seek the advice of researchers in their planning. Frequently this is done through meetings known as "Mind swaps" attended by researchers and product team leaders: the researchers present the state of the art in relevant areas and the product team leaders explain their ambitions for the product. From the resulting dialogue opportunities for technology transfer (or perhaps further research) are identified and technology transfer can begin.

As products progress through subsequent milestones schedules and feature sets become fixed and opportunities for technology transfer diminish, although if a product group hits a specific roadblock, or there is an external development they need to respond to urgently, they may ask Microsoft Research to help find a solution.

As a product heads toward release technology transfer opportunities are limited, but then, as the product goes into the final stages, resources are released to start the planning and design of the next version of the product and researchers can contribute once more. Indeed there have been examples where Microsoft Research has taken a role in the definition and incubation of new products or step changes to existing products – one notable recent example being a coordinated campaign across all the research laboratories to help the company build the Windows Live search engine and associated infrastructure.

"Executive product reviews" form part of the product development cycle – these are occasions where product groups plans and accomplishments are scrutinized by the most senior technical executives in the company. MSRPM members attend these meetings to learn what is happening in a product group, current status and issues. (Microsoft Research also takes part in a similar executive review process: four to six times a year notable projects are presented to the company's senior technical executives. Microsoft Research invites senior representatives from pertinent product groups to attend as observers and it is not unknown for the research review to stimulate a broad and challenging discussion between the researchers, the product team and the executives about the best use of the technology inside the company.)

1.3 Licensing

Not all technology transfer is internal to Microsoft. Sometimes the company will license technologies to other companies. This can happen for a variety of reasons: it may be that a technology has no benefit to current or planned Microsoft products; it may be there are applications of the technology in other markets than those in which Microsoft operates, or there is a prospect of growing the application and hardware ecosystem that surrounds Microsoft's own products. Examples of the latter include licensing the FAT file system format to manufacturers of storage devices and Windows Media codecs to manufacturers of media players. A particular version of licensing process is a program called "IP Ventures" where Microsoft works with the Venture Capital community to create a company to exploit a licensed technology, or to assist an existing company raise the investment required to license a Microsoft technology and take it to market.

1.4 Managing Intellectual Property

Licensing is principally about the transfer of "Intellectual Property" (IP) usually in the form of patents and possibly copyrighted software. For a commercial software vendor protecting intellectual property is an important part of doing business. Product groups expect Microsoft researchers to have been diligent in protecting intellectual property they have created and to be careful not to include third party IP in a technology transfer. Microsoft Research is supported by a dedicated IP team who help researchers secure patents for new inventions of strategic value to the company and ensuring the copyright of research prototypes is appropriately managed. This team provides a service to researchers rather than acting as a control system – researchers are expected to take the initiative in telling the team about new IP that needs protecting. The IP team also provides researchers with appropriate licenses to allow collaborative work with colleagues in universities and other organizations. In the simplest case results can be made available through an internal "code posting tool" which automates much of the process of putting software on the Microsoft Research web site and issuing a basic "research use only" licence. Most patenting is done in the USA and fortunately US patent law allows publication of research results up to 12 months ahead of filing a patent claim so that IP protection does not get in the way of freely publishing results and scientific collaboration.

1.5 Technology Transfer and Pastry

The remainder of this paper describes the technology transfer activities for a Distributed Hash Table (DHT) system called MS Pastry, developed at Microsoft Research Cambridge. It is an interesting example because it exercised many of the steps outlined above and had the novel experience of being a research result that was initially rejected as a candidate for transfer, then subject to external licensing before finally concluding as a joint internal technology transfer and IP Venture.

2. OVERLAY NETWORKS

"Overlay networks" have been a hot topic in the research community in the period 2000-2006. In part the interest was spawned by early peer-to-peer (P2P) file distribution protocols that appeared on the Internet, such as Napster and Gnutella. In the research community it was realized that these networks could also be suitable for distributed applications such as publish/subscribe systems, distributed databases, consistency management for replicated data and distributed failure detection.

An overlay network can be characterized as an addressing and messaging framework layered on standard Internet protocols to support decentralized models for computation and information processing. Overlay networks are interesting because they have the potential to scale incrementally with demand and to fully exploit distributed processing for robustness, in comparison to centralized server based solutions. The basic principle of an overlay network is that there are a set of resources distributed across a network of interconnected computers and at the basic level each resource has a unique identity to distinguish it from any other. The central purpose is to allow any computer in the system to quickly locate a resource given its unique identifier. Other services including communication, caching and replication to support higher level functionality are often layered above the basic naming capability.

To allow overlay networks to scale, the process of translating overlay resource identifiers is itself distributed – i.e., does not reply on either a central table of mappings or replication of complete knowledge at every node. There have been two approaches to achieving this – unstructured overlays and structured overlays.

In an unstructured overlay each computer has a set of links to other peers and name resolution consists of a flooding or random walk search through the aggregate graph structure until the desired resource is located. In contrast, a structured overlay has distributed routing table and a hop-by-hop algorithm for exact-match queries.

During the period covered by this article there has been much debate whether structured overlays were more expensive to maintain compared to unstructured overlays and whether their topology constraints make it harder to exploit heterogeneity and limit the kinds of queries that can be supported. Because of this, researchers at Microsoft Research in Cambridge explored both options: Kermarrec, Massoulié and others looking primarily at unstructured overlays in terms of "Gossip Protocols"; Rowstron, Castro, Costa and others looking at structured overlays in the form of distributed hash tables. A 2005 paper from the group [3] which showed that structured overlays could outperform unstructured overlays in terms of maintenance overhead, that structured overlays could be modified to exploit heterogeneity and that there were ways to perform floods and random walks that allowed structured overlays to support complex queries with better performance than unstructured overlays. In consequence the DHT work came to dominate.

In parallel with the investigations by researchers the Windows Networking product team also began developing overlay network technology as an extension to the Windows operating system. A first product was released in 2003 and has continued to evolve to the current day. The product system used by structured and unstructured overlays – structured for name resolution and unstructured for data communication. The motivation for the former was to provide fast lookup, which the motivation for the latter was to provide an adaptive system that could adjust to changing transmission patterns. In the current cycle of development a DHT like capability is also being added to the system. To explain this further, and show the technology transfer influences, it is first necessary to give some details of each system.

2.1 Gossip-based probabilistic multicast

Gossip-based multicast protocols rely on a peer-to-peer interaction model for multicasting a message either using IP multicast or a randomly generated multicast tree. Such protocols are scalable since the load is distributed among all participating computers and they use redundant messages to achieve reliability and fault tolerance. Though early gossip-based approaches were proven scalable, they generally relied on a non-scalable membership protocol: they assumed that the subset of computers was chosen uniformly among all participating nodes, requiring that each node to know every other node or for there be a central list of nodes. Kermarrec and colleagues at Cambridge developed an hierarchical membership protocol which provided each computer with a partial view of the system known as a "local subscription list" and an algorithm for determining the necessary fanout for random multicast trees that would give a high probability of any message reaching all computers in the overlay network [6]. In their system, when a node receives a new subscription request, it forwards the new node id to all members of its own local view. It also creates c additional copies of the new subscription (where c is a design parameter that determines the proportion of failures tolerated) and forwards them to randomly chosen nodes in its local view. (The local view of a node is initialized with the id of the node at which that node first subscribed to the group). When a node receives a forwarded subscription, provided the subscription is not already present in its list, it integrates the new subscriber in its partial view with a probability p with p derived from the size of its view. If it doesn't keep the new subscriber, the node forwards the subscription to a node randomly chosen from its local view. These forwarded subscriptions may be kept by the neighbours or forwarded, but are not destroyed until some node keeps them. Each node maintains two lists, a *PartialView* of nodes it sends gossip messages to and an *InView* of nodes that it receives gossip messages from, namely nodes that contain its node-id in their partial views. If a node i decides to keep the subscription of node j, it places the id of node j in its partial view. It also sends a message to node node j telling it to keep the node id of i in its *InView*.

2.2 MS Pastry

Work at Microsoft Research Cambridge on structured overlays is based on a distributed hash table called Pastry [13] arising from joint work between Ant Rowstron and Peter Druschel, at that time a visiting researcher from Rice University to Microsoft Research, Cambridge. Others joined the project and the Microsoft team developed their own version called "MS Pastry" [2] which has good performance under churn and an efficient implementation of proximity neighbour selection [4].

Pastry maps "keys" to overlay nodes. Overlay nodes are assigned node ids selected from a large identifier space of size 2^b (where b is a system parameter) and application objects are identified by keys selected from the same identifier space. MS Pastry selects node ids and keys uniformly at random from the set of 2^{128} 128-bit unsigned integers and it maps a key I to the node whose identifier is numerically closest to k modulo 2^{128}. This node is called the key's root. Given a message and a destination key in 2^b, Pastry routes the message to the key's root node. Each node maintains a routing table and a leaf set to route messages. Node ids and keys are interpreted as a sequence of digits in base 2^b. The routing table is a matrix with $128/b$ rows and 2^b columns. The entry in row r and column c of the routing table contains a random node id that shares the first r digits with the local node's node id, and has the $(r + 1)$th digit equal to c. If there is no such node id, the entry is left empty. The uniform random distribution of node ids ensures that only $\log_{2^b} N$ rows have non-empty entries on average. Additionally, the column in row I corresponding to the value of the $(r + 1)$th digit of the local node's node id remains empty. Nodes use a neighbour selection function to select between two candidates for the same routing table slot. Given two candidates y and z for slot (r, c) in node x's routing table, x selects between them using metrics such as Round Trip Time to further optimize performance.

An additional data structure at each node is called the leaf set which connects all the nodes in the overlay into a ring. Each leaf set contains the *l*/2 closest node ids clockwise from the local node id and the *l*/2 closest node ids counter clockwise where *l* is a system parameter influencing the scope for replication of data in applications using MS Pastry. The leaf set ensures reliable message delivery.

At each routing step, the local node normally forwards the message to a node whose node id shares a prefix with the key that is at least one digit longer than the prefix that the key shares with the local node's node id. If no such node is known, the message is forwarded to a node whose node id is numerically closer to the key and shares a prefix with the key at least as long. The leaf set is used to determine the destination node in the last hop.

Several applications have been built using MS Pastry as the foundation:

1. PAST – a robust archival storage system which provides scalability, availability, security and cooperative resource sharing [12]
2. SCRIBE – a generic, scalable and efficient group communication and event notification system supporting application level multicast and anycast. Scribe is efficient, self-organizing, flexible, highly scalable and supports highly dynamic groups [6]
3. SQUIRREL – a decentralized, cooperative web cache, enabling web browsers on desktop machines to share their local caches resulting in an efficient, scalable, self-organizing and fault-tolerant web cache exhibiting performance comparable to a centralized web cache in terms of hit ratio, bandwidth usage and latency [7]
4. SPLITSTREAM – a high-bandwidth content distribution system based on end-system multicast which distributes the forwarding load among all the participants, providing robustness to node failures and accommodating nodes with different bandwidth capacities [5].

2.3 Windows P2P Networking

Concurrently with the Microsoft Research activity, the Windows networking product group became interested in peer-to-peer technologies for a variety of scenarios including serverless instant messaging, real-time matchmaking and game play, shared workspace collaboration for ad hoc workgroups, file sharing, shared online audio and video media experiences, distributed processing of complex tasks and aggregation of computer resources. The first versions of Windows P2P Networking shipped in 2003 and have continued to evolve since that time.

Windows Peer-to-Peer Networking architecture consists of the following components [10]:

Graphing – this component is responsible for maintaining a set of connected nodes known as a graph and providing flooding and replication of data across the graph.

Grouping – this component provides a version of graphing with with secure facilities for group creation, invitation, and peer node connection to a group. The Grouping component uses the Group Security and Group Security Service Provider (SSP) subcomponents.

NSP – The Name Service Provider (NSP) is a generic Windows component that provides a mechanism to access an arbitrary name service provider. Peer-to-peer applications use the NSP interface to access PNRP.

PNRP – this component provides peer-to-peer name resolution.

Identity Manager – this component enables the creation and management of peer-to-peer identities.

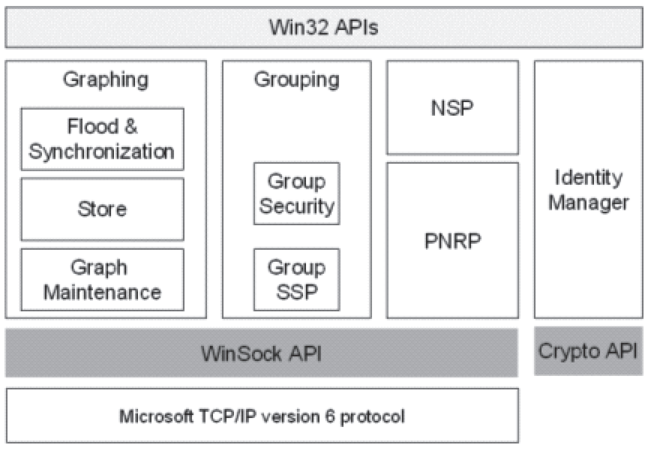

Figure 1: Windows P2P Networking Architecture

Of these components, PNRP and graphing are the two that were most strongly influenced by the work on gossip protocols and distributed hash tables.

2.4 PNRP

In PNRP a *peer name* is a label for a network endpoint, which can be a computer, a user, a group, a service, or any other object. Peer names can be registered as unsecured or secured. Unsecured names are just text strings that are subject to spoofing, as anyone can register a duplicate unsecured name. Unsecured names are best used in private or otherwise protected networks. Alternatively, peer names can be secured with an RSA key pair. Only the key owner is able to demonstrate ownership of a secured name.

PNRP ids are 256 bits long. The high-order 128 bits, known as the peer-to-peer (P2P) id, are a hash of a peer name assigned to the endpoint The low-order 128 bits are used for the Service Location, which is a generated number that identifies different instances of the same P2P id. The 256-bit combination of P2P id and Service Location allows multiple PNRP ids to be registered from a single computer.

Each peer node manages a cache of PNRP ids that includes both its own registered PNRP ids and other entries cached over time. The entire set of PNRP ids located on all the peer nodes in a system comprises a distributed routing table. It is possible to have entries for a given PNRP id located on multiple peers. Each entry in the PNRP cache contains the PNRP id, a certified peer address (CPA), and the IPv6 address of the publishing node. The CPA is a self-signed certificate that provides authentication protection for the PNRP id and contains application endpoint information such as addresses, protocol numbers, and port numbers.

The name resolution process for PNRP consists of resolving a PNRP id to a CPA. After the CPA is obtained, communication with desired endpoints can begin.

There are two different methods of performing PNRP name resolution based on the version of PNRP. PNRP version 1 was included in Windows XP and used a recursive system for name resolution, and is not discussed in this paper. PNRP version 2 was redesigned for Windows Vista to use an iterative approach with stronger security and reliability guarantees.

To perform name resolution in PNRP, the peer examines the entries in its own cache for an entry that matches the target PNRP id. If an entry for the PNRP id is found, a PNRP Request message is sent to the matching peer. If an entry for the PNRP id is not found, the peer sends a PNRP Request message to a peer in the cache that corresponds to an entry that has a PNRP id that closely matches the target PNRP id. (More precisely, a random choice is made between the two peers with the closest ids to the target).The node that receives the PNRP Request message examines its own cache and does the following:

- if the PNRP id is found, the requested peer replies directly to the requesting peer.
- if the PNRP id is not found and a PNRP id in the cache is closer to the target PNRP id, the requested peer sends a response to the requesting peer containing the IPv6 address of the peer that corresponds to the entry that has a PNRP id that most closely matches the target PNRP id. From the IP address in the response, the requesting node sends another query to the IPv6 address referred to by the first node.
- if the PNRP id is not found and there is no PNRP id in its cache that is closer to the target PNRP id, the requested peer sends the requesting peer a response that indicates this condition. The requesting peer then chooses the next-closest PNRP id.

The requesting peer continues this process with successive iterations, eventually locating the node that registered the CPA.

To keep the sizes of the PNRP caches small, peer nodes use a multi-level cache, in which each level contains 10 entries. Each level in the cache represents a one fifth smaller portion of the PNRP id number space (2^{256}). The lowest level in the cache contains a locally registered PNRP id and other PNRP ids that are numerically close to it. As a level of the cache is filled with a maximum of 10 entries, a new lower level is created.

In summary, PNRP provides a distributed routing table for which an arbitrary PNRP id can be resolved by forwarding PNRP Request messages to the next-closest peer until the peer with the corresponding CPA is found.

A peer graph is a set of nodes that are multiply connected to form a coupled network for the purposes of propagating data in the form of records or point-to-point data streams. Another way to think of a graph is as a collection of peer graph nodes connected such that any peer graph node may communicate with all other graph nodes via a series of logical neighbour connections.

A well-connected graph has the following properties:

- It is connected. There is a path between any two nodes
- It has a small diameter. There are a relatively small number of hops between the nodes on the farthest edges of the graph. The benefit to a small diameter is that updates are propagated rapidly to all graph nodes
- It is robust. The graph remains connected even if some nodes or some connections disappear.

2.5 Graphing

A graph is built based the connections of neighbours. A neighbour in a graph is a peer graph node that is one graph hop away (is directly connected via a TCP connection). A graph hop is a logical connection that operates above the Internet layer, and can therefore be on the same subnet, or one or multiple router hops away.

A flooding protocol is used to do the following:

- propagate the addition of new records to all the nodes of a graph
- propagate the updates of changed records to all nodes of a graph
- propagate the deletion of deleted records to all the nodes of a graph.

To perform these functions, each flooded record is identified by a globally unique identifier (GUID), and has an increasing version number or sequence number, and is further qualified by an age or a status. In addition, a synchronization process ensures that peers have the same set of records, if necessary by initiating further flooding of messages that have not been fully propagated.

A graph maintenance protocol defines how the group evolves to maintain robust connectivity and to maintain a small diameter as follows:

- a signature procedure computes the signature of a group. If the group is partitioned, each of the partitions will have a different signature. This can be used to detect that two or more partitions need to be repaired. Designated nodes in the graph known as contacts keep track of the signature records. Contacts are elected randomly.
- a reconnection procedure allows nodes to establish appropriate connections.
- a disconnection procedure allows nodes to leave a graph without creating a hole in the graph.
- when information is flooded across the graph, a graph node that has multiple connections will receive multiple copies of it. To decide which connections to keep and which to remove, a graph node evaluates flooded information and calculates a bidirectional utility index, a number used to indicate the usefulness of the information sent between given connected peers. The utility index has a low value when the information sent across the connection has been consistently previously received and is of no value.

On an ongoing basis, based on the current utility index and the information that is received during the flooding, peer nodes make adjustments in their connections to neighbouring nodes. Connections are created and removed so that the graph converges to a topology that is optimal for flooding for the current traffic pattern.

2.6 Comparison with MS Pastry

From the foregoing description it is clear there are strong overlaps but also strong divergences between the Windows P2P Networking protocols and MS Pastry. With the benefit of hindsight it can be seen the architectural differences reflect differing design goals between the two projects.

The Windows P2P Protocols were designed with a stronger focus on strict layer abstractions, whereas MS Pastry had a stronger focus on enabling higher level applications functionality: to some extent this reflects the fact that the Windows protocols were developed by a networking group and MS Pastry by a systems group, but also the need of product group to provide decomposable building blocks with APIs that can be supported long term. By contrast the various MS Pastry applications re-used software from earlier projects without ensuring a clean set of abstractions between "base" and "extension" functionality.

The Windows protocols separated node location (in PNRP) from peer-to-peer communication (graphing). Pastry and PNRP are both structured overlays but with different organizations. PNRP is optimised for fast look up of remote nodes and does not provide an integrated communications capability. The goal is to reduce the number of hops taken to locate a Node id rather than exploit locality. Locality is important in MS Pastry since intermediate nodes take on caching and forwarding roles in several of the applications.

PNRP uses an iterative search protocol whereas Pastry is recursive: the Windows design does not require intermediate nodes to do work on behalf of other nodes, nor can a rogue node exploit the search protocol to launch a denial of service attack by saturating the routing algorithm with bogus requests. The first point exposes a philosophical debate about the meaning of "peer-to-peer": should members of a peer-to-peer system be required to do work on behalf of others, even if it has no direct benefit to them or should those generating work be responsible for carrying it out? For the Windows P2P system the second approach was selected as it presents fewer opportunities for abuse, even though it may forgo some efficiency advantages.

The Windows graphing protocols, by contrast, form an unstructured overlay between peer nodes, and indeed the protocol design was influenced in the early stages by the results obtained in the Microsoft Research work on probabilistic multicast gossip protocols. As it appears today however, in the graphing protocol links between nodes are adjusted dynamically according to traffic patterns – number of hops was found to be less important than absolute latency to get messages through the structure.

3. THE MS PASTRY EXPERIENCE

By 2006 much of the research on the core functionality of MS Pastry had been completed. Researchers continued to use the Pastry code as a tool kit in other projects [11, 1] but in terms of the fundamental principles the research was completed.

There had been many discussions from 2001 onwards between the researchers and the networking product group with both sides looking to combine their ideas. Many concepts did flow from research to product as have been noted in the descriptions of the research technologies above, and some ideas were taken by other groups developing technology for the MSN Internet service and Windows media services.

. The networking product group regularly challenged the researchers on issues such as scalability and security - the latter being particularly important with "trustworthy computing" being a major driver for Windows development and marketing.

There were several attempts orchestrated by senior management to arrive at a single "Microsoft peer-to-peer" architecture during the development of the Windows P2P architecture which never completely succeeded, reflecting the challenge in maintaining compatibility with protocols and APIs that had already shipped – convergence required seamlessly extending the platform rather than replacing it with completely new technology. But, aware of the strong consensus in the research community on the merits of DHTs as a basic distributed computing abstraction, the Windows product team decided to revisit their architecture when planning for developments after the release of Vista. They began a refactoring of the PNRP protocol to expose the "key-based routing table" capability as an accessible API and included further ideas from MS Pastry and other DHTs. In particular key requirements identified included the ability to route short messages to a peer node "closest" to a key, to exploit locality in next hop selection and to allow a node to discover its leaf set.

With this final stage of internal technology transfer essentially completed in 2006 it was decided that the Intellectual Property (patents and software) from the MS Pastry work, and the Squirrel, Scribe and Splitstream applications could be made available for licensing under the IP Ventures programme. At about the same time a young company, Skinkers Ltd, based in London, UK was looking for peer-to-peer technology to enhance its product line. Skinkers develop products that push business communications to dedicated clients on desktop PCs and handheld devices. Messages arrive as alerts that appear over other applications such as desktop alerts, ticker tapes or widgets. The sender of the message can decide who it is relevant for, when it should be seen, how long for and in what format. The initial version of the product was server based using RSS feeds as the transport mechanism. Skinkers were concerned about the scalability of their product and had searched the literature for a solution and in doing so come across MS Pastry and Scribe; they were also interested in extending their system to handle video and audio traffic, including real-time capabilities and for this Splitstream appeared attractive.

Towards the end of 2006 Skinkers raised UK£2M (approximately US$4M) to licence MS Pastry and the applications from Microsoft and fund development of new products using the licensed technology. Microsoft transferred the copyright of the research software to Skinkers (retaining a license for ongoing research use) and licensed Skinkers to use the Microsoft patents embodied in the research code. In addition Microsoft has observer representation on the board of Skinkers board, and has undertaken to assist Skinkers with marketing their products to Microsoft partners and customers. There is a Technical Advisory Board with members from both companies.

An important part of the deal was the agreement for Microsoft to provide a software "shim" to adapt the new functionality being added to the Windows P2P protocols to support the functionality required by the MS Pastry applications code that Skinkers had licensed. This is an ongoing project between the Windows networking product group, a dedicated "technology transfer and incubation" team at Microsoft Research Cambridge and developers at Skinkers. It is planned that the enhanced Windows P2P functionality and the shim will allow Skinkers to migrate their applications that have initially used the MS Pastry code base to a supported product API later in 2007. Showcasing these products will form part of the marketing plan for the new Windows P2P networking functions. The Skinkers relationship gives Microsoft an early beta customer and provides external developer feedback for the Windows peer-to-peer system as it matures. From this perspective the licensing of MS Pastry to an outside company has had the effect of accelerating the transfer of MS Pastry concepts within Microsoft which has a delightful irony much enjoyed by both the research and product teams.

4. LESSONS LEARNED

Technology transfer is an important part of the research activity in an industry laboratory – indeed one of the key motivations for an industry researcher is to see one's work impact products that are used by millions of people every day. Technology transfer rarely takes the form of a single transfer of a body of software into the next release of a product. Product development may proceed in parallel with research, especially in new areas of technology. Technology transfer is often an iterative process with a steady flow of ideas in both directions. Product groups have a different outlook to researchers: product groups care about planning, durable APIs and modular architectures that can be evolved. Product groups are strongly driven by customer requirements and application scenarios rather than by the merits of a technology in its own right. Integration with other product features and compliance with over-arching product ambitions relating to issues such as security and inter-operability are also major constraints on product developers which researchers can afford to ignore. Product groups go through product release cycles and new technologies generally are only admitted to products early in the cycle for future versions, so often researchers have to be patient, waiting for the right moment to bring their results to the attention of product groups.

Because of these differences in approach, an industry research organization has to develop mechanisms to impedance match between researchers and product developers. At Microsoft Research the MSR Program Management team and the Technology Transfer and Incubation teams play an important role in this respect.

Licensing to outside companies is an increasing important form of technology transfer for many companies, including Microsoft, and is often an ongoing process as the intellectual property is transferred and the recipient supported in taking ownership and making effective use of it. Licensing is not just about deriving income from ideas, it can also be part of a corporate strategy to better understand a market and develop strong partners in important application areas.

5. ACKNOWLEDGEMENTS

This paper is very much a report on the work of others and the author wishes to thank to all those who contributed recollections, comments and advice including members of the MS Pastry project, (Ant Rowstron, Miguel Castro and Manual Costa) the Windows P2P Networking Protocols team (Sandeep Singhal), the Cambridge Technology Transfer and Incubation Team (Mitch Goldberg, John Miller), the IP Ventures team (David Harnett) and the team at Skinkers (Matteo Berlucchi, David Long and Luke Reid).

6. REFERENCES

[1] Caesar, M., Castro, M., Nightingale, E., O'Shea, G. and Rowstron, A., Virtual ring routing: network routing inspired by DHTs, *Proceedings SIGCOMM 2006*, (Pisa, Italy, Sept. 2006).

[2] Castro, M., Costa, M. and Rowstron, A. Performance and dependability of structured peer-to-peer overlays. *Proceedings DSN 04*, (Jun. 2004).

[3] Castro, M., Costa, M. and Rowstron, A. Debunking some myths about structured and unstructured overlays, *Proceedings of NSDI 05*, (Boston, MA, USA, May 2005).

[4] Castro, M., Druschel, P., Hu, Y. C. AND Rowstron, A. Proximity neighbor selection in tree-based structured peer-to-peer overlays. Tech. Rep. MSR-TR-2003-52, Microsoft Research, Aug. 2003.

[5] Castro, M., Druschel, P., Kermarrec, A-M., Nandi, A., Rowstron A. and Singh, A. SplitStream: high-bandwidth multicast in a cooperative environment, *Proceedings 19th ACM Symposium on Operating Systems Principles*, Lake Bolton, NY, Oct., 2003).

[6] Castro, M., Druschel, P., Kermarrec, A-M., and Rowstron, A. SCRIBE: A large-scale and decentralised application-level multicast infrastructure, *IEEE Journal on Selected Areas in Communication (JSAC)*, **20**, 8, (Oct, 2002).

[7] Ganesh, A., Kermarrec, A-M., and Massoulié, L. Peer-to-Peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, *52*, 2, (Feb. 2003).

[8] Iyer, S., Rowstron, A. and Druschel, P. SQUIRREL: A decentralized, peer-to-peer web cache, *Proceedings 21st ACM Symposium on Principles of Distributed Computing*, (Monterey, CA, Jul. 2002).

[9] Levin, R. A perspective on computing research management, *ACM Operating Systems Review*, **41**, 2, (April 2007).

[10] Microsoft Corp., Windows peer-to-peer networking, TechNet Article, available on-line at http://www.microsoft.com/technet/network/p2p/default.mspx (accessed 5 February 2007).

[11] Narayanan, D., Donnelly, A., Mortier, R. and Rowstron, A. Delay aware querying with Seaweed. *Proceedings of 32nd International Conference on Very Large Data Bases (VLDB 2006)*, (Sept.2006)

[12] Rowstron, A. and Druschel, P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, *Proceedings 18th ACM Symposium on Operating Systems Principles*, (Lake Louise, Alberta, Canada, Oct. 2001).

Rowstron, A. and Druschel, P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, *Proceedings of Middleware 2001*, (Germany, Nov. 2001).