I. Overview

Project Title

Turning Internet Programming into Child's Play

Abstract

Currently, only well-trained professional programmers can construct distributed programs. Existing programming tools for distributing computations across networked computers and dealing with the communication and coordination of the parts are not suitable for children and end-users.

We propose to create Distributed ToonTalk by extending ToonTalk, an existing ludic programming environment that has been successfully mastered even by young children, to enable the construction of distributed applications. Using Distributed ToonTalk as a creativity tool, children will be able to build multi-player networked games. Using it as an educational tool, they will also be able to explore advanced computer science concepts ranging from mobile processes to concurrent algorithms to security, in a concrete and playful manner. Using it as a communication tool, they will be able to set up interconnected many-to-many private secure communication channels capable of transmitting text, media, live programs, and other communication channels.

The core technical challenge is to give ToonTalk a communication mechanism that is robust, scalable, secure, and fault tolerant. In ToonTalk pieces of a computation running on a single computer communicate via a concrete metaphor of birds that take things to their nests. We propose to use the same metaphor for communication between ToonTalk programs distributed over the Internet. Birds and nests need to be able to be freely copied across the Internet. When given a message, a bird will deliver copies of the message to her nests, which may be spread out across the world.

We plan to accomplish this by using structured peer-to-peer overlay networks. We plan to use DKS (the Distributed K-ary System) peer-to-peer middleware developed at KTH/Royal Institute of Technology and the Swedish Institute of Computer Science (SICS). It supports scalable Internet-scale multicast, broadcast, name-based routing, and provides a simple Distributed Hash Table (DHT) abstraction.

III. Project Description

Background

ToonTalk is a unique ludic programming tool. It provides an animated game-like virtual world where children can build programs by training robots. These trained robots work on boxes, which contain numbers, text, pictures, sounds, sensors, remote controls, birds, nests, robots, and other boxes. Robot training is a form of programming by example where the child trains a robot to perform a specific task and then generalizes the trained robot by removing details from the training example. The underlying computation model is based upon concurrent constraint programming, a synthesis of concurrent logic programming and constraint logic programming. One of the core ideas underlying ToonTalk is that advanced computational abstractions can be provided as concrete analogues that preserve the semantics and expressive power of the original computational abstraction but are much easier for children to understand, master, and enjoy.

A child understands the concurrency model in terms of trucks that are loaded with a box and a team of robots that will work on the items in the box. A loaded truck drives off, builds a new house, places the robots on the floor and gives them the box to work on, thereby spawning a new process. Robots attempt to match the contents of their box with the boxes they were trained to work on. When there is a match, the robot repeats what it was trained to do. When there is no match, the robot passes the box along to its next teammate.

The complexity of coordinating the activities of a large number of autonomous processes is greatly simplified by the fact that boxes and their contents (i.e., data) are not shared. Each process has its own private copy of a box. Process communication is accomplished by carrier pigeon-like birds. When a bird is hatched from an egg, it provides a new private asynchronous communication channel. The bird represents the right to add messages to the communication channel, and her nest represents the right to remove the top element from a message queue. If the bird is copied, each copy will deliver things to the same nest. This many-to-one communication is critical in building shared state upon a framework in which data is not shared. A team of robots working on a box containing a nest and private data can implement state that is shared by all the processes holding copies of the bird that hatched from an egg in that nest. Shared state is thus only accessible via message passing. Synchronization between such processes results from the fact that a robot holding an empty nest waits until a bird brings something to the nest. Such a process is suspended until more data arrives. Recent doctoral thesis research has shown how children not quite 4 years old can understand the basics of birds, nests, and robots [15,16].

A bird's nest may also be copied. Then when the bird is given something, she replicates herself and the object and each copy of her nest receives a copy of the message. This provides a one-to-many or multi-casting communication facility.

As part of the EU-funded Playground research project (www.ioe.ac.uk/playground) ToonTalk was extended to support "long-distance birds". This early attempt at making a distributed ToonTalk has severe limitations, but it provided encouraging glimpses into how children might be empowered by a truly distributed version of ToonTalk. The idea is that a bird could be placed on the clipboard and pasted into an email message or file. Then someone on another computer could copy and paste the bird into a ToonTalk world. If this bird is given something, she'll fly "through the Internet" to her nest and leave it there. The ToonTalk process with the bird accomplishes this by creating and using a DirectPlay session with the ToonTalk process holding the nest.

Long-distance birds have many limitations:

1. The nest must remain on a computer with the same IP address it had when created. This prevents nest migration and breaks down when IP addresses are assigned dynamically.
2. All participating computers must have ToonTalk running simultaneously. There is no ability to store and forward messages.
3. Nests cannot be copied since the bird has no way of finding the copies.
4. Many school and university firewalls do not permit DirectPlay connections.

Despite these limitations there have been a few successful uses of long-distance birds:

1. An International Game Workshop was held within the Playground Project. Children aged 6 to 8 in London and Stockholm used long-distance birds to exchange games they had previously constructed in ToonTalk. They then used long-distance birds to communicate comments and modified games.
2. Proof-of-concept networked games were built by Playground researchers that in principle could have been played and modified by children. These games ranged from a battleship game to a multi-player Pong game. But researchers did not pursue this due to the fragility of the implementation.

3. A high school teacher in Switzerland had his students build chat programs using long-distance birds.
4. When four 11-year olds who had been involved in another ToonTalk research project (www.weblabs.eu.com) were asked by their teacher to introduce the rest of the class to ToonTalk, they quickly and with minimal help set up a network of about ten computers so that each child was able to send and receive messages with any other child.

We think that some of the limitations of long-distance birds can overcome by using a structured peer-to-peer middleware such as the DKS (Distributed k-ary System)[28,29]. DKS is a middleware that supports a directory service which is completely decentralized. The directory is partitioned and distributed on the nodes in the system. To join the DKS system, a node needs to have the physical address of one node that is already in the system. As nodes join and leave the system, DKS self-organizes the data in the directory such that data is still available.

DKS also provides a simple abstraction for sending messages to nodes based on logical identities rather than on physical addresses, such as IP-addresses. This has the advantage that nodes can leave the system and later rejoin with a different physical address while other nodes unaware of the new physical address still will be able to communicate with it. Furthermore, any node that is inside the DKS system, regardless of whether it is behind a firewall, will be able to receive messages from other DKS-nodes.

Project Details

Vision Statement

While the Internet has proved to be a fantastic resource for communicating, learning, creating, and playing for large numbers of children and adults, *programming* of the Internet is currently limited to a very highly-trained select few. We intend to build a tool that will enable children and amateur programmers to build a wide range of truly distributed applications and games in a playful exploratory manner.

Objectives

The primary objective is to enhance an existing playful programming environment so that children can build programs whose parts are spread out over the Internet. These program parts will be able to communicate and coordinate in a robust and scalable fashion.

Another objective is to develop learning materials including replayable demos, sample code, puzzles, activity sequences, and documentation that will enable children to master distributed programming techniques and concepts.

Expected outcome

1. Children will build simple networked multi-player games and communication programs.
2. Children will learn distributed programming concepts including mobile processes, various communication patterns, security, latency and bandwidth issues, and fault tolerance.
3. Children will use the distributed programming environment as a novel communication platform where they will set up dynamic networks of communication channels to exchange messages, media, and live program fragments in a safe and secure manner.
4. We would not be surprised if there will be additional unanticipated outcomes invented by the Distributed ToonTalk user community.

Description

The technical challenge of making ToonTalk truly distributed is to arrange things so that when a bird is given something, a copy of that thing ends up on every nest of that bird regardless of where in the world the nests are. Furthermore, when a nest is recreated or copied into ToonTalk, all messages that have been sent to that nest need to be retrieved. The implementation should be robust and scalable and not rely upon central servers.

While only some uses of distribution require a secure implementation we intend to provide it as an option. Besides encrypting the messages themselves, we need to ensure that the holder of a bird has only the ability to add messages and not to read messages sent via other birds. Similarly, the holder of a nest should be able to read all incoming messages but not to add messages that will be seen by other nests.

Our proposed solution to the distribution problem is to build upon use a Distributed Hash Table. Because of both technical advantages and the expertise of the project collaborators, we intend to use the Distributed K-ary System (DKS) developed at SICS and KTH in Stockholm by some of the project contributors. We intend to build the distribution extensions to ToonTalk in a modular fashion so that other Distributed Hash Table implementations (such as Pastry developed by Microsoft Research) could be substituted if they provide equivalent functionality.

The core idea is to associate a nest and all of its copies with a global unique ID (GUID). This GUID is then used as a key in a distributed global hash table to an associated message list. When a bird is given something it adds it to the message list. When a copy of a nest is loaded it checks for messages that have been posted since it last checked. It also registers itself as a “live” nest so that when birds post subsequent messages, copies of those messages can be delivered immediately to waiting nests, thereby greatly reducing latency and frequent polling.

When security is required, the copies of a nest can share a public and private key and the copies of a bird can share a different public and private key. The content of the message is encrypted by DKS as a matter of policy.

Since in general once a message is posted it is not possible to determine when it will no longer be needed, we anticipate implementing a policy where old messages are deleted after a certain amount of time has elapsed. Any messages that should be retained longer can be automatically reposted.

An additional requirement on communication in Distributed ToonTalk is transparent and efficient handling of pictures and sounds that have been imported into ToonTalk. We plan to extend and exploit the current ToonTalk solution to this which relies upon one-way cryptographic hashes of the picture or sound file itself. We expect in particular to be able to avoid a large amount of redundant file transmission by using globally unique names based upon the hash of the file’s contents.

Social and Cultural Relevance
Distributed ToonTalk will empower children (of all ages) to build distributed applications and networked games that hitherto have required a very high level of expertise. It will also enable them to form private and secure communities for exchanging messages, pictures, sounds, communication channels, and ToonTalk programs. It will also be an ideal learning tool for anyone learning about distributed computing and many other topics in advanced computer science.

There is currently wide-spread concern among parents and teachers that once children have email or instant messaging accounts, they will be vulnerable to abuse by strangers. In Distributed ToonTalk a child will take out a nest with an egg in it to hatch a new communication channel. The child keeps the nest private and distributes copies of the bird only to those he or she knows. The child can distribute a bird with a floppy, a memory key, a local area network, or email. Nothing can ever appear on their nest unless someone holding a copy of the original bird sent it. There is no way someone could guess the Distributed ToonTalk equivalent of email addresses or IM handles.

Furthermore, children can form clubs or discussion groups by sharing copies of a bird/nest pair, so that members of the club can easily send messages to all the others. They could even choose to limit the distribution of birds so only some members have the ability to post new messages. They can easily create subgroups by sending a message that includes a new bird/nest pair to the recipients of the original group asking them to join by simply saving the bird and nest. The installation of ToonTalk at a site such as a school can be customized so that every user has a nest that initially only the administrator can send messages to thereby bootstrapping a private virtual community.

This project will also expand the scope of applications of distributed hash tables. We are not aware of any attempts to use distributed hash tables as we have planned. This may lead to insights into the strengths and weaknesses of distributed hash tables and their various implementations.

Collaborative and interdisciplinary aspects
This project intertwines advanced computer science, education, and game development. The project is designed to be a collaboration between researchers affiliated with a school of education and those in a computer science department. The design of the ToonTalk programming environment borrows heavily from the computer and video game communities. Interactions with game development community will enable children to be produce better games.

V. Supporting Information

References

ToonTalk: www.toontalk.com
DKS: http://dks.sics.se
Playground Project: www.ioe.ac.uk/playground
WebLabs Project: www.weblabs.eu.com

[1] Chris DiGiano, Ken Kahn, Allen Cypher, and David Canfield Smith. Integrating Learning Supports into the Design of Visual Programming Systems. *Journal of Visual Languages and Computing*. 12, 501-524. 2001

[2] Ken Kahn. Programming as a Video Game. Proceedings of the Computer Game Developers Conference. April 1994.

[3] Ken Kahn, Metaphor Design -- Case Study of an Animated Programming Environment, Proceedings of the Computer Game Developers Conference. April 1995.

[4] Ken Kahn, ToonTalk□ -- An Animated Programming Environment for Children, *Proceedings of the National Educational Computing Conference*, Baltimore, Maryland, June 1995. Also extended version appears in the *Journal of Visual Languages and Computing*, June 1996.

[5] Ken Kahn, Drawings on Napkins, Video Game Animation, and other ways of Programming Computers, *Communications of the ACM*, August 1996.

[6] Ken Kahn, Seeing Systolic Computations in a Video Game World, *Proceedings of the IEEE Conference on Visual Languages*, Bolder, Colorado, September 1996.

[7] Ken Kahn, Helping Children to Learn Hard Things: Computer Programming with Familiar Objects and Actions, in A. Druin, editor, *The Design of Children's Technology*, Morgan Kaufmann, 1998.

[8] Ken Kahn, A Computer Game to Teach Programming, *in Proceedings of the National Educational Computing Conference*, June 1999.

[9] Ken Kahn, From Prolog and Zelda to ToonTalk*, Proceedings of the International Conference on Logic Programming*, edited by Danny De Schreye, MIT Press, 1999.

[10] Ken Kahn, Generalizing by Removing Detail, *Communications of the ACM*, 43(3), March 2000. An extended version is in Henry Lieberman, editor, *Your Wish Is My Command: Programming By Example,* Morgan Kaufmann., 2001.

[11] Ken Kahn, ToonTalk and Logo - Is ToonTalk a colleague, competitor, successor, sibling, or child of Logo? *Proceedings of the EuroLogo Conference*, August 2001.

[12] Ken Kahn, ToonTalk – Steps Towards Ideal Computer-Based Learning Environments, in Mario Tokoro and Luc Steels, editors, *A Learning Zone of One's Own: Sharing Representations and Flow in Collaborative Learning Environments,* Ios Pr Inc, June 2004.

[13] Ken Kahn, The Child-Engineering of Arithmetic in ToonTalk, *Proceedings of the Interaction Design and Children Conference*, College Park, Maryland, June 2004.

[14] Y. Mor, C. Hoyles, K. Kahn, R. Noss, and G. Simpson. Thinking in Process. Micromath 20/2, Association of Teachers of Mathematics, Summer 2004

[15] Leonel Morgado, Maria Gabriel Cruz, and Ken Kahn. ToonTalk in Kindergartens: Field Notes, in Mendez-Vilas, Antonio; Mesa González, José António; Solo de Zaldívar Maldonado, Inés (Eds.), "Information Society and Education - Proceedings of the International Conference on Information and
Communication Technologies in Education (ICTE2002)", *Journal of Digital Contents*, Vol. 1, Issue 1: Formatex, Badajoz, Spain, 2003, ISSN: 1696-313X, ISBN: 84-607-8369-3

[16] Leonel Morgado, Maria Gabriel Cruz, and Ken Kahn. Taking Programming into Kindergartens. EuroLogo, Porto, Portugal. August 2003

[17] Tholander, J., Kahn, K., & Jansson, C.-G. (2002). Real Programming of an Adventure Game by an 8-year-old. *Proceedings of the International Conference of the Learning Sciences,* 2002, Seattle, WA.

[18] Ken Kahn. A decade of progress in concurrent logic programming: A braid of research threads from ICOT, Xerox PARC, and Weizmann Institute. *Communications of the ACM*, 36(3), March 1993.

[19] K. Kahn and W. Kornfeld. Money as a concurrent logic program. In *Proceedings of the North American Conference on Logic Programming*. The MIT Press, 1989

[20] Mark Stefik, Gregg Foster, Daniel Bobrow, Ken Kahn, Stan Lanning, and Lucy Suchman. Beyond the chalkboard: Using computers to support collaboration and problem-solving meetings. *Communications of the ACM,* January 1987.

[21] Vijay A. Saraswat, Kenneth M. Kahn, and Jacob Levy. Distributed constraint programming--the DC framework and Janus. Technical report, Xerox PARC, August 1989.

[22] Vijay A. Saraswat, Kenneth Kahn, and Jacob Levy. Janus--A step towards distributed constraint programming. In *Proceedings of the North American Logic Programming Conference*. MIT Press, October 1990.

[23] Vijay A. Saraswat, David Weinbaum, Ken Kahn, and Ehud Shapiro. Detecting stable properties of networks in concurrent logic programming languages. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing (PODC 88)*, pages 210--222, August 1988.

[24] Eric Dean Tribble, Mark Miller, Ken Kahn, Daniel Bobrow, Curtis Abbott, and Ehud. Channels: A generalization of streams. In *Proceedings of the Fourth International Conference on Logic Programming*, May 1987.

[25] Kenneth M. Kahn and Vijay A. Saraswat. Complete visualizations of concurrent programs and their executions. In *Proceedings of the IEEE Visual Language Workshop*, October 1990.

[26] Ken Kahn. Director guide. Technical Report 482B, MIT AI Lab, December 1979.

[27] Ken Kahn and Carl Hewitt. Dynamic graphics using quasi-parallelism. In *Proceedings of the*

ACM/SIGGRAPH Conference, August 1978. Also AI Memo 480, MIT, June 1978.

[28] Luc Onana Alima, Sameh El-Ansary, Per Brand and Seif Haridi, *DKS(N, k, f) A family of Low-Communication, Scalable and Fault-tolerant Infrastructures for P2P applications*, The 3rd International workshop on Global and P2P Computing on Large Scale Distributed Systems, (CCGRID 2003) (Tokyo, Japan), May 2003.

[29] Luc Onana Alima, Ali Ghodsi, Seif Haridi. *A Framework for Structured Peer-to-Peer Overlay Networks*, In LNCS volume 3267 of the post-proceedings of the Global Computing 2004 (pp. 223-250), Springer-Verlag.