Report on the Seventh ACM SIGOPS European Workshop

Systems Support for Worldwide Applications

Connemara, Ireland, 9-11 September 1996

Edited by Andrew S. Tanenbaum, Program Chairman
Dept. of Mathematics and Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
Email: ast@cs.vu.nl
URL: http://www.cs.vu.nl/˜ast/

The Seventh workshop in this series was held in a spectacular, isolated, nearly-impossible-to-get-to site on the rugged West Coast of Ireland. Despite having a conference room with a view of the ocean and New York City far in the distance, more wholesome Irish food than anybody could possibly eat (especially an edible basket of home-made ice cream twice a day), a peat fire in the fireplace, and more attendees in the Youth Division (under the age of 3) than SIGOPS normally attracts, we had innumerable spirited discussions about systems issues on the present and future Internet as well as a lot of fun.

The choice of the topic was an intentional break with the past. The program committee felt that SIGOPS has been devoting too much effort to ever smaller and smaller kernels, and was in danger of missing the important new research on systems issues related to emerging worldwide systems. This workshop was designed to help redirect SIGOPS. Judging from the number of submitted papers (76, a 40% increase over last time), and their high quality, we are moving in the right direction.

There were seven classical sessions, each consisting of three 20 minute papers, with 5 minutes of discussion at the end of each paper. At the end of each session, the three speakers came up to front of the room to function as a panel and participate in a general discussion on the session topic for 15 minutes.

In addition to these sessions, we also had an evening panel, an evening debate, and a breakout session with small groups pondering what we had learned, followed by a report back to the whole group. Below is a short summary of the papers, selected questions, and the other events. For each paper, the title and authors are listed; the speaker's name is italicized. For more details and the full papers, please see our U.S. web site, *http://mosquitonet.stanford.edu/sigops96/* graciously hosted by Mary Baker or our European web site: *http://www-sor.inria.fr/sigops96* , graciously hosted by Marc Shapiro.

As program chair, I'd like to take this opportunity to thank the general chairman (Andrew Herbert), the local arrangements chairman (Vinny Cahill), the program committee and all the participants for their help, work, and enthusiasm in making this workshop a resounding success.

SESSION 1: PARADIGMS I, Chair: Andrew Tanenbaum; Rapporteur: Andrew Black

A Framework to Support Large Scale Active Applications, J. Bates, Cambridge

This work aims to support a variety of applications in the areas of multimedia, cooperative working, intelligent agents and active databases. As an illustration consider a group of collaborating nomadic users using audio, video, drawing and text. If, for example, one user draws on his (private) slate an event is sent to the others to draw the line. If a user moves, the system keeps them in the session by reconfiguring to their nearest workstation.

At a system level objects announce events they might issue. Other objects register their interest with the source of the event. Events are parametrised. Event arrival is similar to method invocation. Support for distribution is based on the ODP-ANSA programming system, with enhancements to support events and streams, and improved tools. Support for object mobility has also been integrated into the platform.

Using Events to Build large scale Distributed Applications, *R. Hayton*, J. Bacon, J. Bates, K. Moody, Cambridge

This presentation followed on from that of John Bates and described how the event mechanism is implement, as an extension to RPC and IDL technology, and can be used to build large scale distributed applications. The central idea is to model asynchronous callbacks as events instead of using ad hoc mechanisms. Events are objects within the system, and event notification requires prior pairwise agreement between the objects involved.

In a large scale distributed system, filtered multicasting is needed. The placement of the filters and multicast objects is a significant design issue. A heartbeat protocol is used to distinguish between nonoccurrence of an event and failure of its source or intermediate communications.

Techniques for Handling Scale & Distribution in Virtual Worlds, K. O'Connell, T. Dinneen, S. Collins, *B. Tangney, N. Harris, V. Cahill, Trinity College*

This presentation described the VOID toolkit for building virtual worlds. The world is populated with objects, which correspond to the entities in the virtual world, and they communicate via anonymous events. Like the Cambridge system, events in VOID are typed and carry zero or more parameters; unlike the Cambridge events, a client need not know who might raise an event in order to subscribe to it. Instead, subscription is a global system service.

To minimize overhead three techniques are used. The first technique is called Dead Reckoning. Imagine tracking a ship. Rather than sending its position constantly, all sites extrapolate its course; updates are only sent when it deviates from the predicted course. The second technique is called Notify Constraints. It allows a site to say that it only wants events for which some predicate on the parameters holds. The third technique is to divide the system into zones. You can express interest in certain zones only.

Q1: If there is a fire, and you haven't subscribed to fires, do you miss it?
A1: Yes. It is up to the application to subscribe to the relevant events.
Q2: Why does the system let individual objects decide if they are in a zone?
A2: Zones are application dependent. Applications, not the system,decide how to group them.

Panel discussion
Q1: Cambridge events are tied to a source so you don't need zones to limit the scope.
A1: We think subscription will be rare, but we want delivery to be cheap.

Q2: (To Tangney) Should the system know which zones are adjacent to one another?
A2: It can't because zones need not be geographical. They might be functional.

Q3: (To Tangney) Would there be any benefit to having the system provide zone support?
A3: It does. Only it is the application that decides which object is in which zone.

SESSION 2: MOBILE CODE, Chair: Andrew Black; Rapporteur: Andrew Tanenbaum

Towards a Worldwide Civilization of Objects, *M. Condict*, D. Milojicic, F. Reynolds, D. Bolinger OSF

Today's Internet is a collection of medieval cities surrounded by firewalls, with bandits lurking outside. When you let someone in, you cannot tell who it is. What is needed is an infrastructure (civilization) to allow travelers (mobile objects) to move around securely so cities (sites) can admit them without danger. This civilization should merge elements from the WWW, Java, and Corba. It should also provide caching, a hierarchical naming system, persistence, mobility, and security.

Q1: When do threads move?
A1: Whenever they want to.

Q2: What does persistence really mean?

A2: Changes to objects persist when caches are flushed to the file system.

Q3: How do you name mobile objects?
A3: The name server maps them.

Q4: How do you handle consistency?
A4: When you start a method, the object is synchronized

Active Networks, D. Wetherall, *D. Tennenhouse*, M.I.T.

Much printing nowadays is done not by sending data to the printer, but by sending a PostScript file, which is a program, to the printer. This scheme is highly flexible. What we need is to do the same for networks. Processes should be able to send capsules (small programs) to routers and have them stay there to process subsequent packets. The programs sent there should be in an interpretive language, to restrict what they can do. An initial implementation has shown that the IP options can easily be implemented using this technology.

Q1: Can this technique replace firewalls?
A1: Yes, but the capsules must be signed by a certification authority.

Q2: How can the capsules make security decisions?
A2: They cannot.

Q3: How can you prevent a capsule from messing up the routing tables?
A3: A capsule can only touch packets and data related to its flow.

Transportable Agents Support Worldwide Applications, *D. Kotz*, R. Gray, D. Rus, Dartmouth

Getting information about the ever-increasing Internet is becoming harder and harder. The way to find out what is out there is to send out agents, which, in our system are Tcl programs. Agents can be launched from a laptop, roam the Internet while the laptop is disconnected, and return to the laptop when it reconnects. These programs can hop from machine to machine, collecting information and reporting back their results. Security is handled by PGP. Global resource allocation is handled by buying resources with cybermoney.

Q1: How do you protect agents from malicious hosts?
A1: It's hard.

Q2: What happens when an agent has spent all its cybermoney?
A2: It is killed off or sent home.

Q3: Why did you use Tcl?
A3: It is interpreted, available on many platforms, easy to learn, and has GUI support.

Panel discussion

Q1: What is the difference between a mobile agent and a mobile object?
A1: A mobile agent is doing something for a specific user.

Q2: (To Kotz) How do you debug agents?
A2: We have a debugger for tracking agents, setting breakpoints, etc.

Q3: (To Tennenhouse) How do you know what primitives capsules can use?
A3: Router vendors will publish specs.

Q4: Do people want creepy-crawly things entering their machines from the net?
A4: Not now, but when security improves, they will like what they can do.

A5: What good is mobile code?
A5: It accelerates the pace of innovation.

SESSION 3: APPLICATIONS, Chair: Mary Baker; Rapporteur: Marc Shapiro

NILE: Wide-Area Computing for High-Energy Physics, K. Marzullo (UCSD), M. Ogg (U. Texas), A. Ricciardi (U. Texas), *A. Amoroso* (U. Bologna)

The CLEO collider in Cornell produces ‘‘event’’ data, which is filtered and compressed down to one 2-KB event per second, and stored by chunks of one hour of collision data. This data is replicated at approximately 60 data servers. Analysis programs run later. Each analysis will examine on the order of 100,000 chunks and run for several hours. It is easy (because events are essentially independent from one another) to break down an analysis into small, parallelizable components. The bottleneck then will be the connections between the data servers and the analysis processes; a global scheduler attempts to maximize the flow in order to maximize execution speed of each job. An initial version of the Nile system is running now.

Q1: You have few events and lots of processing. It easy to parallelize. What’s the problem?
*A1: Each analysis program needs 10**5 multi-hour runs and new ones are invented all the time.*

Q2: Is this system just for a single application? Will the scheduling work for other applications?
A2: It has other applications.

Supporting Broad Internet Access to TACOMA. *D. Johansen* (Tromso), R. van Renesse (Cornell), F. Schneider (Cornell)

Deploying an application used by many clients poses the problem of distributing and installing the software at all the client sites. TACOMA avoids this problem by using WWW browsers and e-mail. A TACOMA application is installed only in a small number of servers, which export an HTML form to any Web browser. The browser invokes TACOMA by sending the filled-in form to a Web server that interfaces to TACOMA via a CGI script. The result is sent back to the browser as a Java applet. Alternatively, the form may be e-mailed to the server; the e-mail system provides spooling and disconnected operation.

Q1: Doesn’t the Tacoma software have to be installed by hand?
A1: Not at the client. At servers, it is installed by network providers.

Q2: Why not compute at the client, or near the client?
A2: Use the network as a computer (only remote).
A2+ (Fred Schneider): You need a few Tacoma gateways e.g. one at M.I.T. can export all of M.I.T.’s data.

Q3: Why not use HTTP directly? Or Java?
A3: We are exploring these alternatives.

Wide-Area Information Access to Multimedia Historical Sources, *T. Mills,* K. Moody, Cambridge

Information retrieval (IR) is an application that poses very interesting systems problems. The Cobra system supports queries over text, video, audio, and image data and provides relevance feedback. The current focus is the unique database constituted by the records of the English town of Earls Colne, covering the years 1400 through 1750, and containing 13 million words of text, plus maps and photographs. Particular attention was provided to people and places. Cobra represents documents as sets of attributes. Cobra exploits the typed markup information to provide rich search semantics: date range specifiers, identifying alternative names for the same place, and disambiguating common names or tracking surname changes after marriage. Interesting systems issues are to develop distributed scalable solutions, supporting the needs of the specific user community, and monitoring and supporting client context.

Q1: How do you index audio? Based on phonemes?
A1: Use word-spotting: convert audio to {text+match probability}, not content.

Q2: Can this system be used for information discovery?
A2: I'm not interested but why not?

Q3: What is the relevance measure?
A3: This is a big debate in the IR community. The right answer would be user satisfaction.

Q4: How do you discover that ‘‘dog’’ = ‘‘pet?’’
A4: The system needs a thesaurus.

Panel discussion
R1: Much of the work makes previously accessible results available by the Web.

Q1: What is the most important lesson to be learned from these applications?
A1: (Mills) Re-use existing code.
A1: (Johansen) To scale, centralize function in a small numer of servers.
A1: (Amoroso) Parallelizing is easy; the problem is scaling the computation.

Q2: (To Amoroso) Why did you reinvent rather than use existing technology?
A2: Our application is just an integration of existing technologies, nothing new.

Q3: Were you successful in transferring the technology to the user community?
A3: (Mills) We are actively pursuing exploitation.
A3: (Johansen) Yes, too much so.

SESSION 4: PARADIGMS II, Chair: Jean Bacon; Rapporteur: Frans Kaashoek

Etel: A Newspaper-Based Distributed Information System, M. Banatre, *Valerie Issarny*, F. Leleu, IRISA

The goal of Etel system is to deliver information with a high quality of content and service. To ensure quality the electronic version of the newspaper is directly generated from the paper version, which has been professionally edited. Since the newspaper service has to run well on low-bandwidth links (e.g., ISDN), Etel employs aggressive prefetching. At the client-side a predictive prefetching scheme that exploits the structure of newspapers is employed; at the server-side data is grouped according to user access patterns. Early results from a simulation study show that the system is able to deliver data in bounded response time and in most cases the data is at the user before he/she starts reading it.

Q1: Is the goal of the project to try it with real users?
A1: Yes.

Q2: Are you considering user-controlled prefetching, like Point-Cast?
A2: Yes.

An Architecture for a Wide-Area Distributed System, *P. Homburg*, M. van Steen, A. Tanenbaum, VU

Globe is a middleware architecture for turning a collection of machines distributed across the world into a transparent scalable distributed system. It provides a scalable global name space and shared objects to support easy construction of wide-area applications such as news, FTP, etc. The naming service maps a human-readable name into an unique OID (object identifier), which is mapped by the location service into a contact address for a shared object. Caching, hashing, and other techniques are used to make this feasible for a trillion objects. Shared objects are composed of local objects, which may reside on different nodes. The local objects cooperate together to provide the programmer with the illusion of a single shared object. Each local object has four subobjects: for control, communication, replication, and semantics. Using these subobjects programmers can construct shared objects with different fault-tolerance, replication, and consistency semantics.

Q1: Does the hash function destroy locality for the OID lookup?
A1: No, constructing a careful hash function can maintain locality.
A Stake in Cyberspace, *T. Kindberg*, QMW College

Mushroom is a distributed system that supports interactive collaboration. Mushroom grew out of the observation that the WWW is inadequate for interactive collaboration. For example, it is hard to design an application on top of HTTP to allow people to collaboratively debug a remote user's problem. Such applications require shared events (if the remote user types something, it shows up both at the remote user's computer and the support person's computer), concurrency control (the remote user and the support person have a consistent view), security (restricting which data the users can access), and an integrated programming framework for managing both the shared data and the user's view of the system.

Mrooms support these abstractions. Mrooms are system and GUI abstractions in which multiple users can collaborate and have the same view of the state of the system. A user can enter an Mroom and obtain a copy of the system state that the other users in the Mroom observe. An Mroom's state consists of directories and objects. The objects represent either users or information. The Mushroom architecture is a collection of peers and servers. A Java prototype is being implemented.

Q1: What is the concurrency/consistency model?
A1: Users import, process, then export objects.

Q2: Some file systems support private name spaces. Did you learn from them?
A2: They do not deal with the integrity and GUI issues.

Panel discussion
Q1: (To Kindberg) Many CSCW force the wrong user model on people. How do you avoid this?
A1: Mrooms have a minimal model. Users can do what they want to.

Q2: (To Homburg) Why build a single distributed system for all applications?
A2: Globe is not a single OS. It is a middleware layer that helps programmers build new applications.

SESSION 5: CACHING, Chair: Frans Kaashoek; Rapporteur: Jean Bacon

AFS and the Web: Competitors or Collaborators?, *M. Satyanarayanan*, M. Spasojevic, CMU

AFS has been in existence for 12 years. A comparison of it with the Web is instructive, especially with respect to the browsing metaphor. URLs work as ‘‘fuzzy pointers’’ whereas accurate pathnames are needed by AFS. The Web is read-only with a cache hit rate less than 60%; AFS caching is effective at more than 90%. The talk gave three examples of using AFS and the Web together.

Q1: Was development inhibited by IBM having sole rights?
A1: It was a disaster! We couldn't give away software to establish a usage base. Exclusive rights must time out.

Q2: Could the protocols have been published to allow independent development?
A2: It would not have made a great deal of difference. The web interface is easier to use

Q3: What does AFS have to teach the web about caching?
A3: Location transparency, replication and caching, and consistency are useful.

A Multicast-Based Distributed File System for the Internet, *B. Gronvall*, Ian Marsh, S. Pink, SICS

Jetfile is based on multicast rather than RPC and clients may take over server functionality. Neither the network nor the hosts are trusted; digital signatures are used on data and metadata with keys of limited lifetime. The aim is to support a large number of clients where hosts are attached to both high and low speed networks, sometimes over long distances. Clients on high speed networks prefetch large numbers of files whereas caching is used to decrease latency on low speed networks. IP multicast is augmented to give SRM (scalable, reliable multicast).

Q1: Doesn’t scalability depend on the information in the routers rather than the IP-6 address range (2**112)?
A1: It depends on the number of active files. There is room for large tables.

Q2: What is the granularity for load and existence control?
A2: We hope that a file can be cached for a small scope of servers. Timeouts will be used.

Hinted Caching in the Web, *J. Mogul*, DEC WRL

The philosophy is to think globally and act locally in allocating and removing cache entries. HTTP 1.1 has features to assist cache control and servers can provide hints, based on page type, to aid prefetching and removal. LRU is bad for WWW client cache management because size is important. A measure based on (benefit/cost-of-reload) * probability-of-rereference works well, can be generated by the servers and given to the clients. Predictive prefetching completes soon enough and congestion problems can be avoided.

Q1: Do caches behave the same when local and shared? What information is useful to the server?
A1: The server distinguishes per client and shared caches. It’s better to send facts than instructions.

Q2: Can prefetch traffic affect performance?
A2: Prefetching can be killed instantaneously.

Q3: Did you think of including callbacks e.g. when pages are changed?
A3: We would have liked to but it wouldn’t work in a world with firewalls.

Panel discussion

Q1: Are there any measurement to show that traffic is reduced by Jetfile caching?
A1: There are no measurements yet. Parallelism could be used to ease congestion.

Q2: (To Satyanarayanan) Could the web benefit from using AFS?
A2: Yes -it was used by NCSA in 1991. AFS has a consistency mechanism and aggressive caching.

Q3: (To Mogul) Can accurate hit counts be sent to the Web servers, e.g. for billing?
A3: This work is needed and is in progress.

Q4: (To Mogul) What about pages to be generated on-the-fly.
A4: In HTTP 1.1 the URL is used to decide cacheability.

Q5: (To Satyanarayanan) Is it useful to compare HTTP with AFS?
A5: No, the GUI is the critical factor. AFS didn’t have one so we can’t judge.

SESSION 6: AVAILABILITY, Chair: Karin Petersen; Rapporteur: Mark Wood

Cryptographic Support for Fault-Tolerant Distributed Computing, Y. Minsky, R. van Renesse, *F. Schneider*, S. Stoller, Cornell

This work combines cryptographic techniques with replication and voting to make mobile-agent computations fault-tolerant. The paper focuses on ensuring that an agent successfully visits a sequence of hosts in a network. Replication and voting after each stage provide reliability; performance can be improved by combining the voting for a collection of stages into a single vote. Experiments to quantify this performance were described. Replication and voting alone is not sufficient to ensure reliability, because malicious hosts could collude and mislead a voter. This problem can be avoided by the use of cryptographic techniques. One technique presented involves chains of authentication; the other uses a form of secret sharing.

Q1: Could you improve completion time by optimistically not waiting for all votes?
A1: We haven’t tried this.

Q2: How much does encryption cost?
A2: It is a polynomial function on the number of stages.

Q3: What could you apply these techniques to?
A3: A web crawler that traversed links on its own (not browser driven)

World Wide Failures, *W. Vogels*, Cornell

Failure management should be viewed as a fundamental distributed systems service. The implementation should be system- and application-independent It also should provide increased accuracy and scale well. A key idea is to separate suspicion of failure from definitive detection of failure. For improved accuracy, the operating system is instrumented, exploiting the close coupling present between the operating system and process. Accuracy is enhanced further by providing support for process state checkpoints, which are used to determine liveness, and process upcalls, which may be used to interrogate the process. Since failure investigation is best done as locally as possible, node failure monitors provide failure investigation services for a subnet.

Q1: Failure detection is in the eye of the beholder. How can you automate it?
A1: (A long, inconclusive discussion followed)

How to Scale Transactional Storage Systems, *L. Shrira*, B. Liskov, M. Castro, A. Adya, M.I.T.

Borrowing from the idea of cooperative caching in file systems, the authors have developed a technique called split caching, in which servers reliably cache modified objects, and clients cache disk pages. Because network activity is assumed to be much cheaper than disk activity, a server can more quickly respond to a request for an object by redirecting the query to a client caching the disk page containing the object then it could if it had to read the object from disk. To install updates to objects on a page, servers retrieve the data from a client's cache if possible, thereby eliminating the need for an installation read. The authors' cache coherence protocol, fragment reconstruction, eliminates the effect of false sharing while supporting lazy updates.

Q1: How would you compare this work to distributed shared memory?
A1: DSM systems do not deal with disks.

Q2: Could you modify the protocol to send updates instead of invalidations?
A2: It is too expensive to send updates at commit time; sending them asynchronously requires a coherence protocol

SESSION 7: OPERATING SYSTEMS, Chair: Yolande Berbers; Rapporteur: Robbert van Renesse

System Support for Mobility, *A. Black*, J. Inouye, OGI

Mobile computing will become the norm, but existing applications assume a static environment. Applications cannot adapt because current operating systems refuse to inform the application about environment changes. We need to develop a methodology that allows both applications and operating systems to adapt. The application should make its assumptions explicit, and the system should report any violations. The Synthetix operating system is being developed to investigate these issues.

R1: To discover environmental changes, the OS must do work the user may not want to pay for.

Q1: Why can't the application just specify what it needs?
A1: Translations that the application can't do may be needed (e.g., frames/sec to device parameters)

R2: Sometimes a complete rewrite of the application is necessary in response to system changes.

The Persistent Relevance of the Local OS to Global Applications, *J. Lepreau*, B. Ford, M. Hibler, U. Utah

The Web currently does many things wrong, but these problems will be fixed. Once this is done, operating system problems will surface. New abstractions that are being developed will to some extent deal with this, but end system security is in a crisis. The security mechanisms are a mess, and the policies are bankrupt (see, for example, Java security). We need a fully isolated execution environment and are developing a virtualizable architecture named Flux. The main idea here is nested processes. The protection model is basically capability-based. Even though the architecture is simple, it turns out to be fairly hard to build.

R1: This is still like the Java model, and we know that it's hard to get it right.

Q1: Why is the system hard to build?
A1: NSA actually thinks that this is a simple system.

R2: You make the ‘‘shepherd’’ of mobile code responsible for the actions of the agent. Not so with capabilities.

Server Operating Systems, M.F. Kaashoek, D. Engler, *Greg Ganger*, D. Wallach, M.I.T.

The goal of server operating systems is high performance and modular, incremental constructions, while maintaining interapplication protection. Their system allows applications to do their own resource management, provides scatter/gather I/O operations, has a combined disk cache/retransmission pool, and zero-touch disk to network movement. The TCP implementation lives in user space, and uses an aggressively optimized event driven style. Optimized protocol stacks are provided by compiled ILP techniques from high-level specifications. The performance of their experimental HTTP server is an order of magnitude better than other servers.

Q1: How do you schedule multiple servers on the same machine?
A1: Hope is that bursty access to the servers will be uncorrelated.

Q2: How you can optimize the protocol without changing the clients?
A2: By reducing the number of packets that the server sends.

Q3: Is it realistic to require server implementers to write system code?
A3: There are only a few servers, really.

Q4: How is a server operating system different from a regular one?
A4: Different security concerns, no graphics interfaces, and lower level abstractions.

Q5: You have to be careful when you go into production, e.g. what about backups?
A5: The exokernel allows multiple OSes to run concurrently and safely on the same machine.

BREAKOUT SESSION, Co-chairs: Karin Petersen, Mary Baker; Rapporteur: Yolande Berbers

The attendees were split into five groups. Each group got three questions, one of which was common to all the groups. After one hour the groups came back and one spokesperson per group presented the answers. One group gave their presentation in the form of a story which is impossible to render well in a summary. The answers to the questions by the other groups, and the reactions of the other participants, can be summarized as follows.

The question common to all groups was: ‘‘Is there going to be any widely-used worldwide application other than the Web in the next 5 years?’’ Answer: Yes, but the Web will stay as the interface and platform. The following applications were found likely to be important in 5 years:

- Conferencing (for business and academic communities)
- Banking (more for businesses, but also personal financial transactions)
- Multi-user games (worldwide Doom ?)
- Structured access to information
- Electronic commerce
  + Between businesses mail and fax orders will be replaced
  + Small business will also go worldwide
  + Customisation: clients will be able to specify their needs and shop worldwide

- Shipping services
- Proprietary applications on proprietary networks, e.g. banking, travel agents
- Stealing and electronic crime

Q1: How should worldwide naming and object location really be done ?
A1: Improved DNS with 2-level names and federated name servers in a single naming tree.

Q2: What is an ‘‘outage’’ or ‘‘black-out’’ of a worldwide application or service?
A2: Either no access to the net or a much lower quality of service. Caused by

- *Bugs in particular code (can be fixed in a reasonable time)*
- *Bugs in a particular design (can take months to fix)*
- *Bugs in particular specifications and standards (can take years to fix).*

Prevention must be based on better software engineering, on diversity of code and of design (so that not everybody suffers from the same bugs). Black-out is most likely to be caused by bugs, but can also be caused (in decreasing order of likeliness) by a criminal act, by terrorists, by military attacks.

Q3: Who will make money on worldwide applications, services, infrastructure, etc ?
A3: ISPs, ad agencies, vendors, consultants, software providers, lawyers, but not SIGOPS members.

Q4: How will security for worldwide applications be enforced?
A4: Technological solutions, but also legislation and law enforcement.

Q5: Will worldwide applications force a common natural language upon people?
A5: No. We will need automatic translation plus multilingual browsers.

PANEL: 20-20 Vision: The Internet 24 Years Ahead

Moderator: Andrew Black; Rapporteur: Mary Baker
Panelists: Sape Mullender, Mike Condict, Brian Bershad, John Bates, Tim Kindberg, Jeff Mogul

The question the panelists addressed was: What will networks and distributed systems look like in the year 2020? The main differences in panelists’ opinions centered around such issues as whether telecommuting and cooperative meeting software will ever really work. Some panelists see technology in the future driving people further apart, whereas some see it creating new ‘‘villages’’ amongst us. Some panelists see global distributed systems as solving more of our unpleasant tasks, such as banking, travel arrangements, etc. Others (and Jeff in particular) see a very dark side of global distributed systems in 2020.

Sape Mullender

Sape compared the current situation with 24 years ago, looking at where things had changed to predict where things will change. In 24 years: wireless Internet access for everyone. 100Mbps (immobile) access for everyone. Personal computers in all shapes and sizes, all interconnected. All interpersonal telecommunications and most business information exchange done over networks. Electronic payments replace cash. Appointment bookings, reservations, etc. done electronically. Some, but not all shopping. Reality remains real.

Mike Condict

Extrapolated from 15 years ago. Starting from 300 baud to current rates, we see bandwidth doubling every two years. In 2020, there will be enough bandwidth for ‘‘total immersion.’’ Virtual reality will be the preferred interface but will be used as an interface to real people, not virtual ones. Pornography will continue to drive technology. Maybe we’ll have 3D papers.

Brian Bershad

Agent technology will be used to improve actual reality. Problem with reality: there are unpleasant tasks including bills, paper reviews, unwelcome correspondence, travel arrangements, etc. It's this stuff that will be handled by virtual agents, freeing us from this unpleasantness. We'll only be disturbed when some catastrophy requires it.

John Bates

By 2020, Bill Gates will have bought the Warsaw Pact. Each side will know if the other launches its Java applets. We'll all be wearing virtual reality suits. Strong authentication needed or you'll find you're having virtual reality sex with a different person than you intended. TVs, phones, etc. still around. Nobody will telecommute, although maybe cooperative meeting software will cancel the need for workshops. Ubiquitous computing devices available: for the newspaper on the train, for downloading suspension preferences into your motorbike, etc.

Tim Kindberg

Looking year by year through the next 24 years, how will things change? 1996: SIGOPS. 2000: Network access integrated with the desktop, still wrangling over encryption chips, video conferencing widespread. 2005: Ubiquitous computing environments, electronic cash, adaptive user interfaces, still wrangling over encryption chips. 2010: 50% of people in service industries ‘‘telework,’’ European unemployment is at 30%, greater than 70% of people can program their home entertainment devices, global user identity registration necessary, QoS guarantees for a price. 2015: Gap between info haves and have-nots widening, new Luddites attack computer installations, hackers still exploit software interactions (widespread and vast crashes), movement back to an ‘‘organic’’ society. 2020: we're physically present at SIGOPS again.

Jeff Mogul

We can't predict technology. We can predict that in 24 years the world will be a rotten place. (Rich richer, poor poorer. Corporations outrank governments. Telecommuting shifts power upwards (hard to unionize when there's no workplace anymore). Internet globalization creates conflicts: Privacy versus access, censors won't give up easily, English will be the language of the powerful. People will create private communities: restricted membership, often aligned with small-scale geography. Drug-like addictions will be common, and flat-rate Internet service might not be scalable. We'll appreciate remote interactions: safer, physical disabilities don't much matter, families are dispersing. Health care will be distributed: hospitals are dangerous sources of infection, no longer a need for centralized info; we'll use remote doctors.

Questions

Q1: Is the future of the net a medieval civilization? Will no one be safe on the net?
A1: People still go to New York.
A1+: Some people don't go back, though.

R1: ‘‘Back to the village’’ effect on the family. We'll all stay home with the kids and telecommute.

R2: Paperless office never happened. Computers don't *do* things for us. We just end up with more responsibilities.

Q2: Which sorts of skills will people lose?
A2: Handwriting? Expressing themselves? Touch typing? Arithmetic? Probably the ability to be patient, and this could be a terrible problem.

DEBATE: MOBILE CODE: VIRUSES ON THE RUN OR TRUE DISTRIBUTED COMPUTING?

Moderator: Frans Kaashoek; Rapporteur: Karin Petersen

Pro team (for mobile code): Robbert van Renesse, Doug Terry, David Tennenhouse

Slogan: Mobile code is Noble code; Mobile code is global code

Con team: (against mobile code): Andy Tanenbaum, Satya, Mike Jones
Slogan: Just say NO!

Robert van Renesse, from the Pro team started off the presentations. He brought up a list of mobile code examples, dating as far back as 1968 with RFC5. The only gap in appearances of mobile code occurred just after the Morris virus in 1988. His basic argument was the code has to be brought to where the action is. He supported the argument using three graphs with exactly the same two curves, one for RPC and one for mobile code, and latency as the Y-axis. The only variable was the X-axis. The graphs said that mobile code has lower latency as soon as you execute more than 8 operations, if the number of bytes accessed is larger than 4K, and with the network technology of 1996 and later. In Van Renesse's view the network should be the computer.

Andy Tanenbaum spoke next, and his bottom-line was that he doesn't want random uninvited processes wandering around his computer, the same way he doesn't want random uninvited people wandering around his house. Tanenbaum made the case that the only way to make your computer secure against mobile code was to disable everything, which would make it useless for yourself. In particular, he believes that ACLs don't help because graduate students are going to break them. He also warned that arguments for fool-proof virus-encapsulators should be treated like arguments 30 years ago that nuclear power plants were completely safe. For Tanenbaum the solution is ''Just say NO! Don't let the mobile-code genie out of the bottle.''

Doug Terry then continued the Pro argument saying: (1) Mobile code is Noble code (at this point the intent seemed mostly literary); (2) Mobile code is global code: the point being, that mobile code greatly simplifies the maintenance of client/server based distributed systems, instead of having to maintain code compatibility through versioning, mobile code could just be sent to the servers; and (3) Mobile code is ''No, Bill!'' code: it allows code to become available from other sources. Finally, Terry made the point that not all mobile code is uninvited. There are restricted instances of secure code, like video sources that send both data and the decoder to the video sink, and Bayou's mergeprocedures used for conflict resolution.

Continuing with the Con arguments, Satya's introductory remark was that ''Mobile code is Insidious.'' It runs without you asking for it. Basically, it is like getting mugged without you knowing it. He immediately proceeded to predict the obvious rebuttal, for instance, the existence of many programs, like OS daemons, that run without one's knowledge. The difference being, that most of these daemons are not provided by random people. He also believes that until now there was a clear distinction between the potential maliciousness of read and execute operations, and that mobile code is breaking that distinction. For Satya you should ''Just say NO, while you still have the chance.''

Closing the arguments for the Pro team, David Tennenhouse took a personal stand against the Con Team. He argued that the Con Team was just afraid of a little bit of code, for which the only new thing is its mobility. To address the safety issues one should specify safe implementations of interfaces, that operating systems can run code in sandboxes, and that it was a NOOP if running Java. Tennenhouse listed a number of reasons why mobile code is Noble: it can adapt, it can be used for monitoring and fault detection, it can merge information in the network, it can accelerate the pace of innovation, and it is fun! He closed with two statements ''It's time to put computing into distributed systems'' and ''All good things must come to a continuation.''

Finally, Mike Jones of Microsoft came up, put a slide with the Microsoft Windows logo on the projector, said that Microsoft had decided that mobile code was not a good idea, thanked the audience, and went back to his chair, amid much hilarity. He then came back for some more verbose arguments. He indicated that although the authenticity of shrink wrapped software does not equal that of digitally signed software, it does provide a certain degree of protection, since CD manufacturing is harder than sending code over a network. He also said that graduate students would be able to sign anything they could get a key for anyway. He also reasoned that giving your credit card number away over the phone does not have the same risks as giving it away over the internet: first, because phone tapping is hard; second, because internet hackers can operate remotely; and third, because hackers can steal ''more than they can carry.'' In his closing, true to the Con team, he asked to ''Just say NO.''

At this point, the audience got into the act. Brian Bershad said that the Con Team had successfully argued why code (in general) was good or bad, but had not made any particular arguments regarding mobile code.

Mike Cronklin took a stand with the argument that mobile code is always uninvited, for example, he said you bring over the Java applets when you click on them. Doug Terry mentioned again that at times mobile code is invited, like in the MPEG decoder example.

Eric Jul stated that mobile code should be handled similar to people that you invite to your house, that is, you have to be careful about who to invite; that you need the chance to say no; and that you would like to know who the people really are, so Verisign type signatures of software are necessary.

Some conversation also focused on the impact of how much of the mobile code was sold versus it being given away, and the security implications of that.

Andrew Black mentioned that applications created by students are often more robust, and better designed, than commercial software from major vendors. For example, Hanover, NH (Dartmouth) can usually be relied upon to produce better Macintosh software than Redmond, WA.

Marc Shapiro pointed out that Andy Tanenbaum was being contradictory, since one of Tanenbaum's students in an earlier presentation had mentioned that in their system code gets dynamically linked so that objects can move.

Satya insisted that with Java applets you are not explicitly asking to receive code, that clicking on a link was significantly different than making an FTP request for example.

Jeff Mogul pointed out that none of us were typical users. Andy Tanenbaum may rightfully not trust his computer, but that ordinary, more trusting, users would need real protection.

Fred Schneider turned the argument to mobile code being just another opportunity for operating system scientists to rise to the challenge. He said that we had done so before with faster processors and GUIs. He sees the real problem being C - the language: it is here to stay, which means that there will be interesting problems to be solved.

Finally a vote was taken, and to no one's surprise the majority of the audience voted for the Pro side of the debate.