Network Tools and Tasks

Kuczun, Kyle and M.D. Gross

Proc. ACM Conference on Designing Interactive Systems (DIS), Amsterdam. 215-222

1997

design machine group
University of Washington
Seattle WA USA 98195-5720
http://depts.washington.edu/dmachine

Network Design: Tasks & Tools

Kyle S. Kuczun, Mark D. Gross
Sundance Laboratory for Computing in Design and Planning
College of Architecture and Planning
University of Colorado
Boulder CO 80309-0314
+1 303 492 2807
kuczun@spot.colorado.edu, mdg@cs.colorado.edu

ABSTRACT
Designers often draw to produce artifacts for thinking and communicating about their designs. These artifacts (drawings) provide the designer with various levels of abstraction to conceptually frame the design problem. Because network designers traditionally make drawings throughout the design process, we propose that the computational environment should facilitate and capitalize on this activity. We describe a suite of computer based network design tools that employ freehand drawing as an interface.

KEYWORDS
local area networks, domain oriented design environments, freehand drawing environment, computer human interaction, levels of abstraction

INTRODUCTION
Recently an experienced network designer on the engineering faculty at our school (Evi Nemeth) was asked to evaluate some experimental computer based tools for network design. Her comment was telling: She said, the tools were interesting and perhaps valuable but she herself wouldn't be able to use them. When asked what tools she employed to design networks, she explained that she used a pen and a pad of yellow paper, which she carried everywhere. This anecdote served as the impetus for our work on a paper-like interface for network design tools.

Computer based tools can help designers in various ways: by providing relevant case studies, connections to databases, indexed storage of designs, or links to resources on the World Wide Web. However, until designers can see how computational support can actually help them make better designs, they will not expend the effort to use computational tools [2]. The premise of this paper is that computational tools for network design should be as easy to use as the traditional pad of paper or whiteboard, yet augment these media by providing back-end design support, in the form of case bases, simulations, critiquing, and access to the world wide web for collaboration and communication.

The rest of the paper describes a suite of five tools—Capture, Find, Advise, Lookup, and PostIt—that use a freehand drawing environment to support specific tasks of network designers. We begin in Section 2 with a design scenario that illustrates some basic tasks in network design. Section 3 discusses problems with typical commercial network CAD applications. Section 4 presents our five prototype tools that use drawing as a front end to support specific tasks of network designers. Section 5 concludes with a discussion of related work and future directions.

WHAT DO NETWORK DESIGNERS DO?
We conducted several informal surveys with network designers and administrators at the University of Colorado, Boulder campus to learn about the tasks and work practices of computer network designers. We interviewed designers at the school's Computing Network Services (CNS) as well as the campus board of network managers. We also spoke with system administrators in the departments of architecture and engineering. The following simple and typical network design scenario is based on what we learned in these interviews, and the experience of one of the authors (Kuczun) as a professional network administrator.

> *The system administrator at the College of Architecture has been asked to design an electronic design studio supporting 20 workstations, a file server, and assorted peripheral devices. The Architecture building already has an existing building network that connects to the campus backbone, with ethernet connecting faculty offices, a research lab, and instructional labs.*

Like many design tasks, this job does not begin from scratch. Rather it involves adding to and modifying an existing artifact.

> *The design process begins with the production of an augmented floor plan of the new studio. The designer draws the existing network as a basis to implement a new network design. The designer would look at protocol and performance issues, and physical constraints, as new network elements are sketched in. At this stage the designer is merely "ball parking" the design, producing an artifact that can be used for discussion or consideration.*

Next, the designer might compare the new network layout with similar configurations elsewhere on campus to check consistency or to explore other options. The design process continues by refining and redesigning. Typically, designers first sketch out generic machines, and later decide on specific platform types and protocols that would best suit the situation. Designers refer to a collection of magazines, catalogs, and the World Wide Web to obtain vendor information for product specifications. Then the designer might refine the design further, consulting with network administrators, other designers, and perhaps vendors. In each of these consultations, the discussion focuses on the design represented in the diagram.

This scenario illustrates several typical tasks in network design. First, the designer makes a representation of starting conditions, the already existing computer network. Second, the designer augments that representation to provide an initial, quite abstract, design for consideration and discussion. Third, the designer may compare the proposed design with known, already existing, configurations. Fourth, the designer may refine the abstract design, specifying the initial representation with details obtained from handbooks and catalogs. And finally, the designer may consult with others (key users, system administrators) who have an interest in the design.

Drawing is an important medium for each of these tasks. Our interviews with network designers and administrators revealed that much initial network design is done on paper and whiteboards. Designers make diagrams and sketches to record ideas and to convey them to others while they think about and solve design problems. During the design process the designer refines and specifies an initially highly abstract diagram into a detailed plan for implementing the network design.

Designers use various levels of abstraction in their drawings. An initial drawing may represent a highly abstract conceptual design. As the design process continues the designer draws a more detailed diagram. Figure 1 is an initial diagram drawn by a network designer. This first drawing is an abstract overview sketch that provides little detail about platform, protocols, or hardware. Figure 2 shows a more detailed elaboration of the same network design. Here the various nodes of the network are specified along with the type of router and wiring to be used.

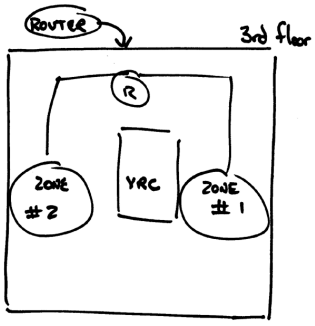

Figure 1 shows an initial, quite abstract, design.

WHAT'S WRONG WITH EXISTING NETWORK DESIGN TOOLS ?

Our survey of professional network administrators and available commercial network CAD packages seemed to indicate that the tools are ill-suited to the tasks of network designers. In our informal campus survey we found that few network designers actually used the commercial network CAD packages they owned for designing. Rather, they used them as utilities to help administrate or debug an existing network. The consensus was that these tools fell into two categories: Either they were best suited to administrating existing networks, or they focused more on presentation rather than supporting the actual process of design.

We examined four commercial tools for network design and management: ClickNet (PinPoint Software Corporation), CANE (ImageNet, Ltd.), NetSuite (NetSuite Development), and LANSurveyor (Neon Software). The first three are billed as network design tools; LANSurveyor as a management utility. ClickNet and NetSuite provide a tool-palette interface that enables a designer to construct a graphical representation of a network as a graph of connected icons, with descriptions of the components stored in a back-end database accessed by mousing on the icons. LANSurveyor monitors the state of an existing network, and enables a network manager to obtain performance and diagnostic information on specific components and subnetworks. CANE offers a layout tool similar to ClickNet and NetSuite, but also provides the ability to simulate various traffic loadings.

The palette and icon interfaces and back end databases that these tools provide are at first complex and bewildering (figure 3). They each require that the designer learn a fairly idiosyncratic scheme of linking visual representations with information about components in order to produce a network diagram. And the payoff is low: the tools do not offer performance modeling, debugging, or critiquing of designs. Thus, "design" in ClickNet and NetSuite consists of selecting icons of network components from a palette and arranging them nicely in a window. The user can add clip art and text annotations to these pictorial representations.

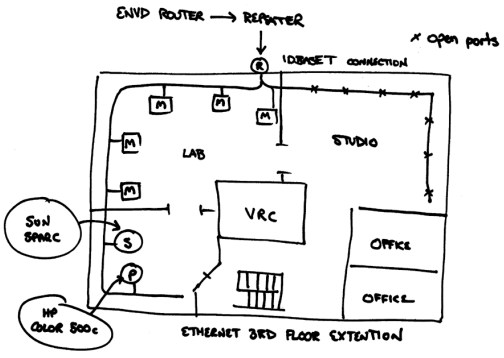

Figure 2 shows a later, more detailed, design.

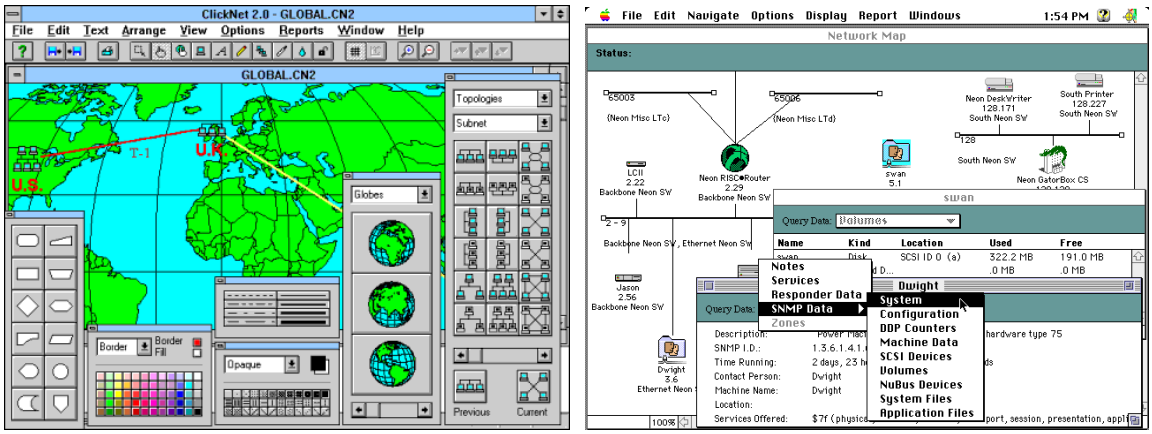

Fig. 3. Complicated interfaces in commercial network CAD packages . ClickNet (left), LANsurveyor (right).

The result is a tool geared towards winning the hearts of management through slick presentation rather than helping the designer build a working network. Conversely, tools like LANSurveyor provide valuable performance metering functions that can inform design, but they do not explicitly provide facilities for laying out, simulating, and evaluating designs.

The tools that do support layout of network designs do not support the varying levels of abstraction that we found in the whiteboard representations used by professional designers. For example, to instantiate a network component such as a file server, the user must select a specific platform and configure it. By contrast, a network designer would typically begin with an abstract notation for 'file server', and only incrementally specify the platform and needed configuration, based on other parts of the emerging design. The palette and menu interfaces that drive these programs may be partly responsible: With a hand drawn diagram a designer can begin abstractly and specify incrementally, which we argue is a benefit of the traditional way of working with diagrams.

In summary, most commercial network CAD packages fail to support the tasks of a designer because:

(1) they do not support abstraction. The designer must specify details too early in the design process.

(2) their 'generic' palette and menu interfaces take the designer away from familiar actions in the design process, such as drawing.

(3) they are too complex to support basic design. Many packages attempt to provide the ability to design *as well as* manage a network. The software contains a melange of extraneous features for debugging and administration that don't support the task of design.

To address the first two failures of current network CAD software (they don't support varying levels of abstraction; they employ generic palette-menu interfaces), we look to the traditional actions of network designers as a clue about the type of interface to use. We know through observations and experience that designers draw. Because designers draw in many stages of the design process and because drawing does support varying levels of abstraction, we chose to explore freehand drawing environment as that familiar interface for designers. The third failure, that network CAD packages are too complex, trying to serve both design and management of networks, is easily addressed by providing tools that specifically support only design.

FIVE PROTOTYPES: COMPUTATIONAL SUPPORT FOR NETWORK DESIGN

A useful design environment for network designers (and we hasten to add, for designers in any domain) must be integrated and embedded in the tasks at hand. We built a suite of five prototype tools (LAN-Tools) to support the tasks of network designers. We used a pen based drawing environment developed by one of the authors (Gross)—the Electronic Cocktail Napkin program [4, 5]—as a front end to our tools. They support the tasks outlined in the scenario in section 2, and repeated here:

Tool	**Task**
Capture	*Identify and represent* the context (the existing network). Draw initial designs or *augment* existing network diagrams to use as an artifact for consideration and discussion.
Find	*Compare* designs with existing network configurations and *find relevant cases* to check consistency and explore alternatives.
Advise	*Test, simulate, and critique* designs based on knowledge about networks

Lookup — *Refine and specify* network abstractions using details from magazines, catalogs, discussions, and the World Wide Web.

PostIt — *Communicate* network designs with other network designers, administrators, and vendors.

The first of these, Capture, enables a designer to obtain a basic layout of his or her current network. The second, Find searches the university campus backbone for existing networks similar to the current design. The third prototype, Advise, offers a simple simulation and critique of designs sketched on the Napkin. Finally, the last two programs, Lookup and Post-It, use the World Wide Web as a source of information that can be indexed to drawings or as a place for storing and displaying designs. We outline each of these tools below, following a brief explanation of the pen based drawing interface that we used to implement them.

Drawing as an interface - The Electronic Cocktail Napkin

The Electronic Cocktail Napkin tries to bridge the gap between freehand drawing and computational support. It supports the kind of informal drawing that designers do on paper or a whiteboard in the early stages of design. By supporting drawing, the Napkin integrates one of the primary media that network designers use to carry out their tasks. Napkin supports not only the initial act of drawing, but also the use of drawing to aid information retrieval, simulation, and critiquing of designs.

The Napkin program, implemented in Macintosh Common Lisp reads, identifies, and interprets marks the designer draws on a Wacom digitizing tablet. For our network design tools, we first trained the Napkin to recognize a set of symbols that represent various network elements. These included depictions of platform hardware—Macintosh, Intel-based machines, and UNIX computers as well as other elements of networking architecture—ethernet cables, routers, and gateways. Figure 4 shows the Napkin drawing board with three basic network symbols, a Macintosh, printer, and server.

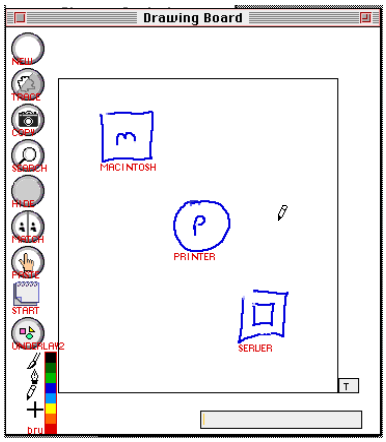

Figure 4. Network symbols recognized in the Napkin drawing environment: A printer is represented by a configuration of the letter P inside a circle.

Capture: representing initial conditions

In the scenario in section 2, the network design begins with a representation of initial conditions. Designers often begin by making a diagram—typically a floor plan annotated with the locations and connections of network components—that illustrates the current state of the network. Designers seldom invent new networks; rather, they add to and modify an existing network. Thus, most networks evolve from simple topologies to hybrid systems of varying complexity. For this reason a way of adapting a design environment to changing or existing conditions is needed. Our first prototype, Capture, scans an existing Appletalk network and produces a diagram of it, which the designer can modify and augment.

The domain of network design has a peculiar characteristic: A design tool can be used to capture and represent existing conditions. In most other design domains the designer must first physically survey existing conditions, a task that must take place outside the bounds of the design environment. For example, an architect must obtain physical "as built" measurements of an existing building before returning to the drawing board or CAD program to design an addition. In network design the designer can simply "ping" the different network devices to obtain a detailed representation of the network without needing to physically visit the site. Thus the ability to integrate information capture with the design environment is fairly unique to the domain of computer network design.

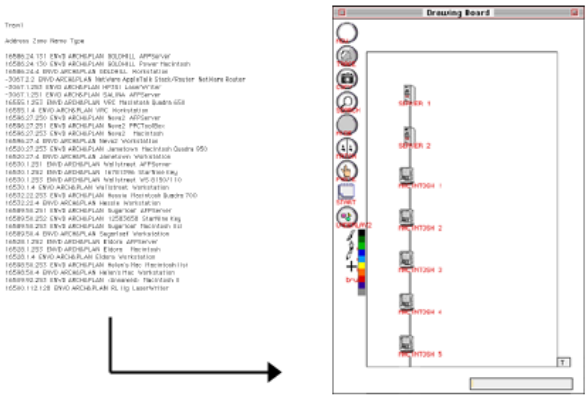

Figure 5. A current version of the designer's network transferred to the Napkin using data from the Trawl program.

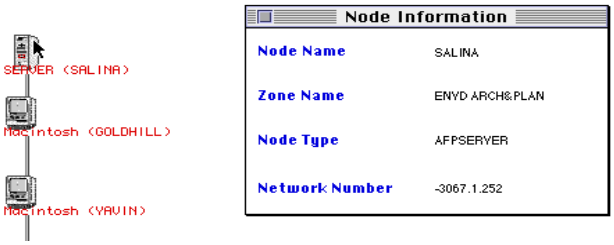

Figure 6. If the designer selects a specific node in the diagram more information appears.

We used a commercial utility program called Trawl [6] to analyze an existing Appletalk network (figure 5) and display it in the Napkin's drawing window. Trawl inspects packets transmitted on an Appletalk network and builds a data file of the network elements it sees. Our Capture program first invokes Trawl to scan the network and then reads in this file to the Napkin. The file is parsed by network element type and displayed in Napkin's drawing window as a string of icons that depict the current network configuration.

Trawl's scan of the network yields more data than can be concisely displayed (e.g., it includes the node types of the network devices). Clicking on an icon in the network diagram brings up additional information about the element (figure 6). Once the existing network has been loaded into the Napkin's design environment, the designer can add new components and connections. When the designer draws a trained network symbol, Napkin indicates that it recognizes the symbol by displaying its name. The designer can underlay a floor plan to provide additional context to the design (see figure 7).

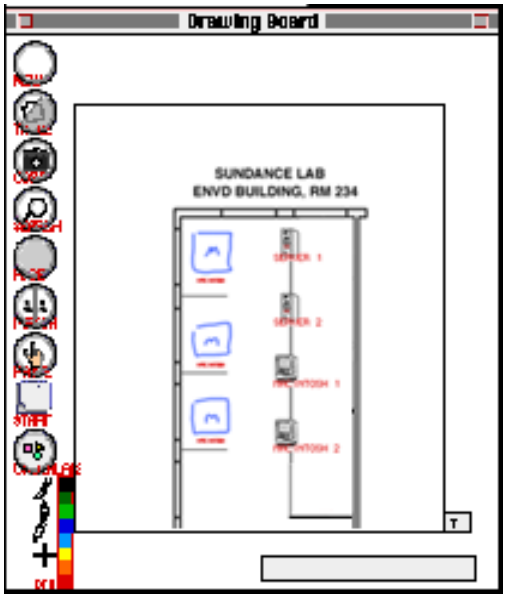

Figure 7. An artifact for discussion. Initial network design with data produced from scanning the existing network, plus additions from the designer.

Find: Retrieving similar network design cases.

In designing modifications and augmentation of a network, designers often refer to other networks they know. "Oh yeah, didn't they do something like that over in the new GIS lab in Geography?". The designer might want to compare a proposed network layout with other known configurations to check consistency or to suggest other options. The set of networks on campus function like a library of design cases [1], from which designers adopt and adapt ideas. As with capturing the current network configuration, this 'case library' is available by scanning the network using a simple utility program.

The Find prototype employs the same Trawl utility program to compare a network design drawn in the Napkin environment with different network zones on a campus backbone. Using Find, the designer can find similar networks elsewhere on campus.

Figure 8 shows the search window for making comparisons. The 'Current network report' field shows the number of each kind of element in the current network design, the numbers of Servers, Routers, Macintoshes, and Printers (S R M P). The example in figure 8 illustrates the report on a design containing no servers, one router, three Macintoshes, and a printer (0 1 3 1). The 'Search for report' field describes the characteristics of networks that the designer would like to find; in this example the designer is looking for networks with the same characteristics as the design: (0 1 3 1).

The designer can set the tolerance of the search using the 'Tolerance' field. The tolerance can be adjusted to allow a difference of ±1 and ±3 , or * to allow a match with any number. These tolerances are combined with the search characteristics (above) to allow inexact matches. In this instance the designers has set the search tolerances to (1 0 * 0), allowing the search for (0 1 3 1) to retrieve 0 ± 1 servers, exactly 1 router, any number of Macintoshes, and exactly one printer. The results of the search, a list of network "zones" that match these characteristics, is listed at the bottom of the report window.

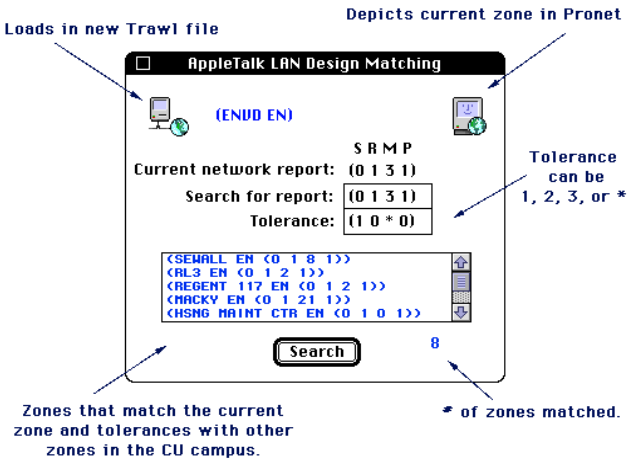

Figure 8. A report window shows a match between the current network diagram (not shown) and other networks.

Advise: Knowledge based critiquing and simulation

Our next prototype, Advise, connected the Napkin drawing environment to a simple network simulation environment called Pronet [10]. Pronet, built as a doctoral project several years ago in Repenning's Agentsheets programming environment [9], simulates various LAN components and provides feedback about the design. The user constructs a network design by assembling network elements from a

palette onto Pronet's worksheet. When the user runs the simulation, Pronet tries to identify errors that might occur in the layout, and offers suggestions. For example, Pronet might augment a network design by inserting a router or a gateway between components, based on its built-in 'knowledge' about network behavior.

The Advise prototype enables the designer to sketch a network diagram rather than construct a design by selecting components from Pronet's palettes and placing them in the worksheet. As the designer draws a network diagram on the Napkin, network elements are recognized and constructed in the Pronet worksheet. The Pronet simulation provides the designer immediate feedback about the design. Figure 9 shows a completed interaction between Napkin and Pronet: The designer's sketch (left) was first translated into Pronet elements, simulated, and finally Pronet provided a knowledge based critique, augmenting the design in the worksheet. The result (right) shows the designer that the initial LAN design would require an additional router, which Pronet has inserted (Macintosh, printer and server were connected via ethernet wiring; the other Macintosh was connected using localtalk.)

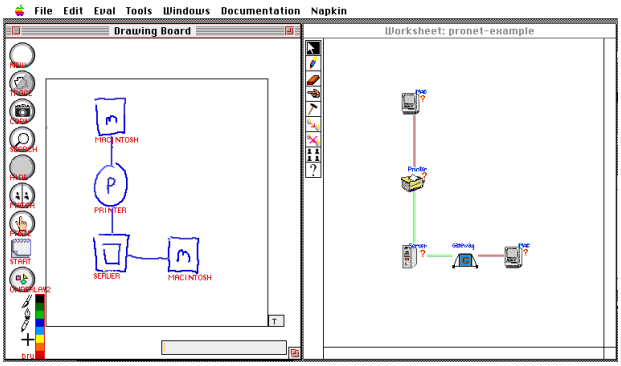

Figure 9 shows a completed sketch in the Napkin window on the left, simulated in Pronet on the right.

Lookup: accessing and indexing catalogs on the Web

As a network design progresses, representations become more formal and specific. For example, the indications of workstations and wires drawn abstractly in network designs must be specified. Network designers gather product information about network devices from a variety of sources, including catalogs, magazines, discussions with peers, and increasingly, the World Wide Web. Our last two prototypes, Lookup and PostIt, link the Napkin drawing environment to the Web for both gathering information and posting designs for comment.

In Lookup, the designer can link drawings in the Napkin's sketchbook with specific sites on the web, for example, to vendor sites. To index a drawing, the Napkin issues an AppleScript command to ask Netscape for the current URL, which is recorded as a hidden annotation to the sketchbook page. The designer can later turn to the sketchbook page, and the Napkin will send an AppleScript command to ask Netscape to go to the linked URL. Figure 10 shows a router symbol linked to Cisco's web site that provides technical information on their routers.

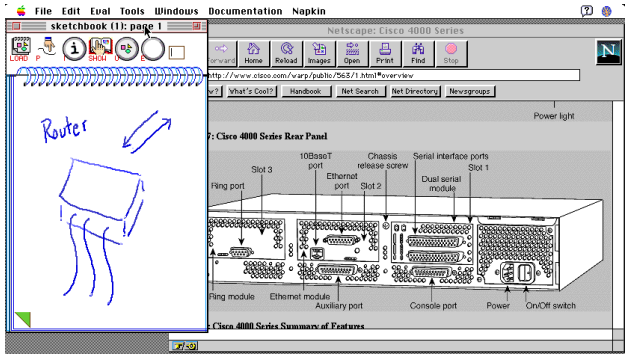

Figure 10. Bookmarking by drawing? A vendor web page has been linked to a sketch of a router.

PostIt: sharing design diagrams using the Web

PostIt supports sharing diagrams with colleagues and vendors for comments, feedback, and discussion. PostIt uses a common gateway interface (CGI) on the web server to receive, convert and post network diagrams drawn on the Napkin. When the designer issues a 'PostIt' command, the current Napkin diagram is first saved as a local PICT file, copied via ftp to an 'inbox' directory on the server. The server's CGI first converts the PICT to a GIF file, then adds a pointer to the new GIF file to a web page used for sharing information (see figure 11).

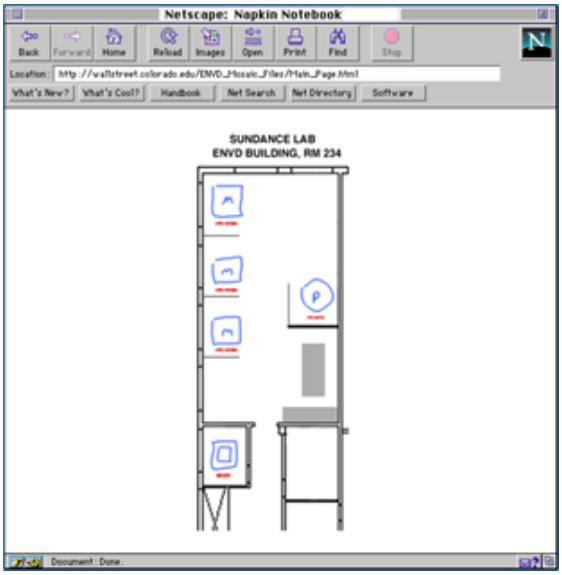

Figure 11. A network design is posted to a web page for discussion.

DISCUSSION

Related Work

Our colleagues in the Useful & Usable group at the Center for LifeLong Learning and Design (L3D) at the University of Colorado, Boulder have built several prototypes focusing on the domain of network design. WebNet (http://www.cs.colorado.edu/~l3d/u-u/project.html) extends network computational support to designers by adding group memory using GIMMe, a WWW collaborative design environment, and incorporates end-user programmability. WebNet serves as a domain-oriented design environment that focuses on specific tasks of the designer. However, WebNet currently supports only a conventional palette-menu interface.

Fischer, Nakakoji, and Ostwald [3] argue that systems that support sketching are more intuitive to a designer because they support the designer in understanding and solving problems using a familiar interface. For example, Nakajoji et al's eMMaC system [8], an environment for multimedia authoring, also supports freehand sketch queries in a library of images.

Future Work

Publishing design diagrams on the web should foster valuable discussion and collaboration. Unfortunately, just posting the diagram is a one way interaction; the 'audience' viewing the diagram has no immediate way to respond to the design. For someone to comment on the design they must communicate in person, by phone, or by email with the designer; but all these modes divorce the argument from the artifact. Better that people reviewing the document could add text, voice, or graphical annotations to the artifact itself. To make comments useful to the designer it helps if the viewer also knows the context and history of the design. More auxiliary information and context must be conveyed than our prototypes currently provide. Generally, how might we improve our tools to be more interactive and promote meaningful collaboration over the web?

One direction of future work would look at providing design rationale with posted design artifacts. We look to McCall's issue based information system for design rationale, PHIDIAS [7]. PHIDIAS provides an asynchronous collaborative design environment that allows users to record, access, and comment on design rationale. Another direction would be to connect two or more designers using the Electronic Cocktail Napkin engaging in a computationally supported design conversation. We can already do this technically, sharing a drawing space over an Appletalk network. Designers could manipulate a network diagram in real time on their Napkins or post drawings to the web for another designer using Napkin to annotate or redesign.

Conclusion

We have argued that the tools that network designers use should support the tasks they need to complete. Often, the tasks of network designers are carried out through the medium of drawing and diagramming. Drawings and diagrams are essential artifacts for thinking and communicating, and as a medium, drawing offers several advantages. One of the most essential is abstraction: Initial designs are abstract, final designs detailed. The ability to view and manipulate a design through varying levels of abstraction is essential. Computational design tools should support working at various levels of abstraction, and drawing is a familiar and traditional medium that can do this. In the excitement over computer based tools, we should not discard the benefits of traditional media and methods of design.

ACKNOWLEDGMENTS

The work described here was done at the Sundance Laboratory for Computing in Design and Planning, where Ellen Do and Adrienne Warmack provided programming assistance, advice, and valuable feedback on drafts of this paper. Members of Gerhard Fischer's 'Useful and Usable' group at the nearby Center for LifeLong Learning and Design (L3D) provided insightful conversations about tools for network design, and the CNS interviews were conducted with L3D's Jon Rieman and Ken Anderson. Material support was provided in part by a grant from the Colorado Advanced Software Institute (CASI) and our industrial collaborator USWest Technologies, and from NSF grant DMII 93-13186.

REFERENCES

1. E.A. Domeshek, J.L. Kolodner, C.M. Zimring, "The Design of A Tool Kit for Case-based Design Aids", *Artificial Intelligence in Design '94*, Edited by J. Gero, Kluwer Academic Publishers, Dordrecht, 1994.

2. G. Fischer, A. Lemke, A. Morch, R. McCall., "Making Argumentation Serve Design," Human Computer Interaction, Vol. 6, No. 2-4, 1991, pp. 393-419.

3. G. Fischer, K. Nakakoji, J. Ostwald, "Supporting the Evolution of Design Artifacts with Representations of Context and Intent", Proceedings of Designing Interactive Systems (DIS) '95, ACM, Ann Arbor, MI, 1995, pp. 7-15.

4. M.D. Gross., "The Electronic Cocktail Napkin - working with diagrams" Design Studies, Vol. 17, No. 1, 1996, pp. 53-70.

5. M.D. Gross, E.Y.-L. Do. "Demonstrating the Electronic Cocktail Napkin", ACM Human Factors in Computing - CHI '96 Conference Companion, ACM/Addison Wesley, pp. 5-6.

6. M. Lowe. "Trawl v1.03", 1993. Shareware. 53 Dolly Avenue, Springfield, NSW 2230, Australia.

7. R.J. McCall, P. Bennett, E. Johnson. "An Overview of the Phidias II HyperCAD System", ACADIA

(Association for Computer Aided Design in Architecture), Edited by A. Harfmann, M. Fraser, ACADIA, pp. 63-74.

8. K. Nakakoji, B. Reeves, A. Aoki, H. Suzuki, K. Mizushima. "eMMaC: Knowledge-Based Color Critiquing Support for Novice Multimedia Authors", Proceedings of ACM Multimedia '95.

9. A. Repenning, K. Schneider, "Deceived by Ease of Use: Using Paradigmatic Applications to Build Visual Design", Proceedings, ACM Conference on Designing Interactive Systems, Ann Arbor, MI, 1995, pp. 177-188.

10. J. Sullivan. *A Proactive Computational Approach for Learning While Working* [PhD dissertation]. University of Colorado, Boulder, 1994.