Distributed Cognition and Joint Activity in Computer-System Administration

Paul P. Maglio, Eser Kandogan, Eben Haber

IBM Almaden Research Center

Troubleshooting large computer systems is often highly collaborative. Because these systems consist of complex infrastructures with many interdependent components, expertise is spread across people and organizations. Those who administer such systems are faced with cognitive and social challenges, including the establishment of common ground and coordination of attention, as they troubleshoot in collaboration with peers, technical support, and software application developers. We take a distributed cognition approach to interpreting a specific instance of problem-solving in administering a web-based system, examining the movement of representational state across media in a single system administrator's environment. We also apply the idea of language use as joint activity to understand how discourse attributes affect what is accomplished collaboratively. Our analysis focuses on information flow among participants and other sources, and how these affect what information is attended to, transmitted, and used.

1 Introduction

Millions of users of online services such as banking and shopping rely on instant transactions, round-the-clock access, and foolproof record keeping. The computer system infrastructures needed to support such applications consist of diverse components, such as database management systems, web servers, and application servers, all of which must work together in complex ways to deliver fault tolerant, scalable, secure applications. Yet with such systems increasing in both size and complexity, *manageability* is quickly becoming a significant obstacle to system administration: Administrators who install, configure, maintain, and support such systems must handle larger and more complex tasks (Anderson 2002; Woods 1988).

Large-scale systems contain many interdependent components, often from different suppliers, that are not always designed to work together. Expertise and responsibility for different components is typically spread across people and organizations. Administrators are faced with daunting cognitive and social challenges. Complexity and scale are such that administering a complete system is usually beyond the abilities of a single person, making collaboration among team members and outside experts crucial to completing many tasks, especially time-critical tasks such as troubleshooting. As a result, administrators have developed many heuristics for problem-solving and many practices for collaborating with others to do their jobs effectively.

Collaborative troubleshooting involves coordinating activity and information from people and other sources. In this paper, we take a distributed cognition approach (Hutchins 1995) to understand problem-solving in system administration, focusing on issues of trust and its relationship to the management of attention. Distributed cognition treats certain arrangements of people and artifacts as cognitive systems, effectively computing functions by transmitting representations (e.g., language, computer commands) across media (e.g., air, computer screens). The idea is that the cognitive computation can be (partially) understood by tracking propagation of representations in this way.

We combine distributed cognition with the joint activity theory of language use (Clark 1996) to interpret the way discourse attributes affect problem-solving in system administration. On the joint activity view, people use language to create and complete projects together, such as the project of coming to a mutual understanding (e.g., agreeing on the cause of a problem and its probable solution) or the project of accomplishing some other task (e.g., following steps to enact problem resolution). An examination of language use as joint activity provides insight into why people interact the way they do (see also Fairburn et al 1999).

In what follows, we examine the process of solving a single problem that occurred during normal maintenance of a web-based system. This episode lasted two and a half hours and involved eight people using many different collaboration tools and resources. By tracking the movement of representational state across media in one administrator's environment, and by analyzing how joint projects are established, we examine how discourse attributes and information flows affect what information is attended to, transmitted, and used.

2 Study

Our data come from field studies conducted to develop knowledge of software system administrators' culture, organization, collaboration, work styles, problems, strategies, and tool use. Our overall goal is to improve products, practices, and processes of administration. In the study discussed here, we observed administrators for five consecutive days in a computer services group that hosts customer web applications. We used several techniques to gather data, including surveys, observations, video recording, formal and informal interviews, and material (hardcopy and online) collection.

In this chapter, we detail one problem-solving episode, analyzing the influence of access to information and aspects of discourse on administrators' collaboration practices and problem-solving effectiveness. The descriptions of agents, representations, and representational activities that follow are restricted to this episode only. We first describe (a) some technical details of the task; (b) the people, computers, and information sources involved; and (c) the kinds of representations that were used.

2.1 Task

A customer installation included a certain software product for managing the flow of data between the public internet and the customer's protected internal network. This software had two parts, a *player instance* running on a server in the public zone and a *maestro instance* running on a server in the protected zone. Communication between player and maestro was regulated by a network firewall (see Fig. 1). To handle additional traffic, the customer requested that a second player instance be added in the public zone (see Fig. 2). The task of adding a second player instance involved creating the new player, configuring the maestro to allow the new player to access certain resources, and configuring the network firewall to permit communication between maestro and player. Network communication between these systems occurs over specific *ports* (represented by integer port numbers) that must be set properly for both the sending and receiving instances (e.g. maestro and player instance). Furthermore, the firewall provides security by only permitting communication in each direction over a limited set of port numbers; these ports and their allowed directions are specified in firewall rules.

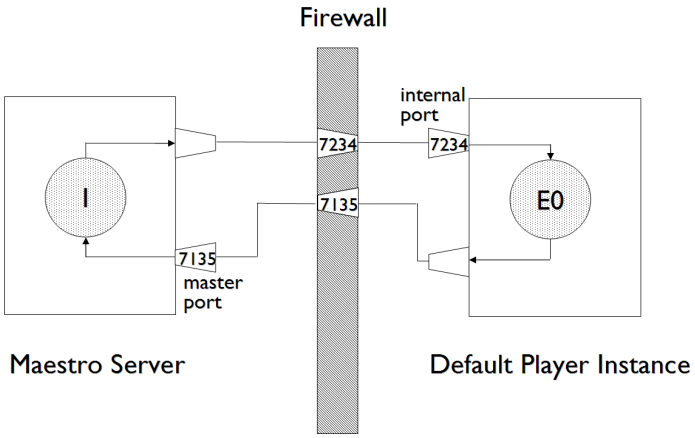

Fig. 1. In the initial configuration, the maestro server communicates with a single player instance through the firewall on ports 7234 (maestro to player) and 7135 (player to maestro).

2.2 People and Computers

Many individuals from many groups were involved in the problem-solving episode. Primary actors included:

- our main administrator (hereafter, ***Admin***),
- the project architect (***Archi***),
- technical support for the product (***Tech***), and
- ***Admin***'s colleague who had access the same systems (***Colle***).

Less important contributors included ***Admin***'s officemate, the customer relationship manager, the project executive, a product developer (and ***Archi***'s friend) , and ***Admin*** and ***Colle***'s manager.

Systems that received, processed, and transmitted information during the episode included:

- an *internal server* machine that ran the maestro application in the protected zone of the network,

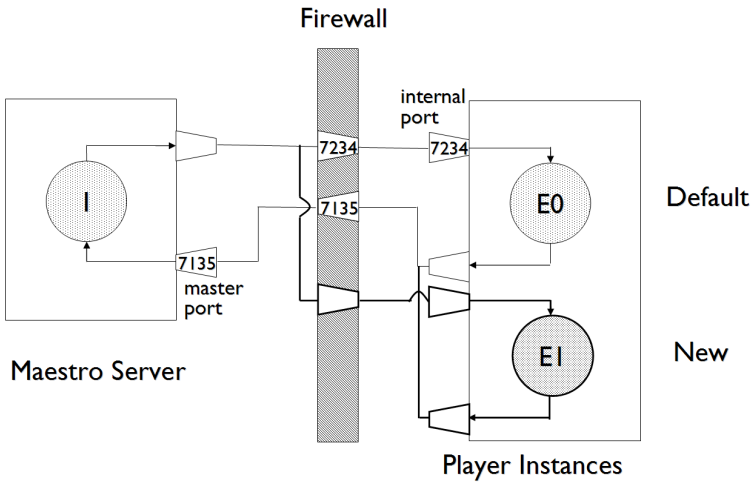

Fig. 2. In the desired configuration, a new player instance (E1) communicates with the maestro server through the firewall on ports 7137 (maestro to player) and 7135 (player to maestro).

- an *external server* machine that ran the player instances in the public zone,
- a *firewall* that regulated communication between internal and external servers,
- the *maestro* process that regulated player access to protected resources,
- the existing *default player* process**,** and
- the *new player* process that the customer wanted added on the external server.

2.3 Representations and Actions

As ***Admin*** and collaborators worked on the problem of adding a new player instance to the external server, they used various media and tools for interacting with each other and with the computer systems. Their communication involved verbal exchanges face-to-face or over the phone, and textual exchanges through email and instant messages.[1] When interacting with computer systems, administrators relied mainly on commands

[1] Email and instant-messages are often used simultaneously. Email is persistent and must be explicitly received. Instant messages "pop up" in a special window on the recipient's screen.

typed directly into the system's *command line*, a general human-computer interface that requires the user to know precisely the names and parameters of specific commands for the computer to execute. Command line users are typically very experienced. Commands can control processes and machines, and can display state and configuration information.

Information representations included *configuration files*, *log files*, online and paper instruction *manuals*, and *port listings*. There were separate configuration files for each computer process, which included settings to specify communication port numbers. Likewise, each process had its own log files that report errors and warnings that occur while the process is running.

We now turn to details of the problem-solving episode in which ***Admin*** worked with many others and consulted many information sources to add a new player instance to the external server.

3 Observations

We begin with an overview of the multi-hour problem-solving episode. We then focus on several specific interactions that illustrate how constraints on propagation of representational state and how discourse attributes affected what information was attended to, transmitted, and used. All observations were taken from ***Admin***'s perspective.

3.1 Overview

Initially, ***Admin*** received an email message describing the steps required to add a new player instance to the external server. These were generic instructions for completing the task. ***Admin*** had to substitute situation-specific information at each step for successful execution. To start, ***Admin*** needed to collect this specific information. Before our observation began, ***Admin*** had sent an email request to the network team to modify the firewall rules to permit communication over two ports: port 7137 from external server to internal server, and port 7236 from internal server to external server (see Fig. 3). Once the network team completed the job, ***Admin*** began to follow the rest of the instructions.

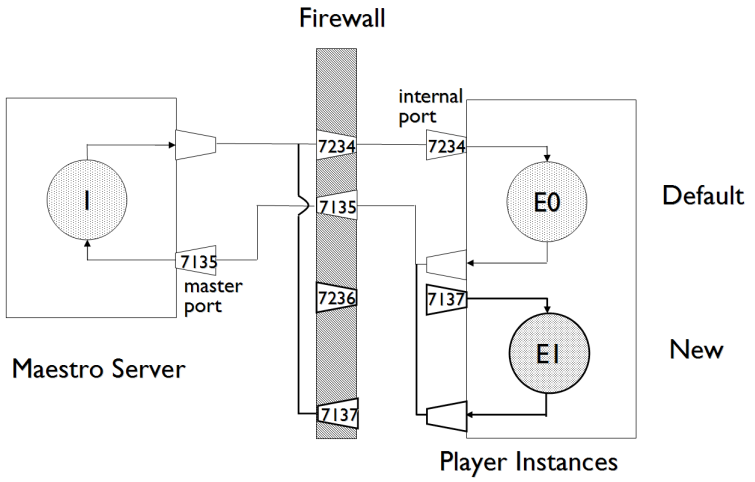

Fig. 3. In the problematic configuration, a second player instance was added (E1), but communication between maestro and the second player instance was not properly established. The second instance can communicate with the maestro server on port 7135 (player to maestro), but the maestro server cannot communicate with the second player instance through port 7137 because 7137 was set up to allow communication in the opposite direction (player to maestro rather than maestro to player) only.

First, ***Admin*** copied from the email the command to create a new player instance and pasted it onto the command line of the external server:

```
m_web create {instance} -m {internal-port}
```

He then proceeded to substitute "`E1`" for `instance` and "`7137`" for `internal-port` by directly editing the text on the command line, resulting in

```
m_web create E1 -m 7137
```

This paste-and-modify was typical of ***Admin***'s style—whether the command was copied and pasted from email or from other sources—he would substitute specific configuration information directly into the command line.

Admin executed the command without any errors, resulting in the configuration shown in Fig. 3. Unaware that anything was wrong, he then

copied from the email message a command meant to configure the maestro server to permit the new player instance access to certain resources. When he filled in the parameters and executed this command, the following error appeared on his screen:

```
Cannot reach server: Error 1231A
```

At this point ***Admin*** appeared confused, probably for several reasons: First, the error message was ambiguous. ***Admin*** was running a command on maestro server to allow player access to certain resources located on yet a third server. The error message, "Cannot reach server," did not specify *which server could not be reached*. Second, instructions and online documentation were not clear about the meaning of `internal-port`, defining it to be "intra-process communication port" *without any mention of the direction of the communication*. Third, the *error message was the result of a problem created during initial player configuration, yet appeared later during maestro configuration*. When ***Admin*** created the player instance (`E1`), no errors were indicated, so he assumed that the problem could not have occurred then.

To try to understand the error, ***Admin*** engaged in phone, email, and instant-message conversations with ***Archi,*** the application architect, and ***Tech***, the technical support person. As time passed and the problem remained unresolved, ***Admin***'s office-mate, colleague (***Colle***), and a developer friend of ***Archi***'s all joined the conversation. At several points, the customer relationship manager and the project executive requested updates from ***Admin*** on the state of problem resolution.

During interaction among ***Admin***, ***Archi,*** and ***Tech, Admin*** was in control of the systems, with ***Archi*** and ***Tech*** asking him for details of configuration and system state, and instructing him to run commands or make configuration changes. By contrast, ***Colle*** could access the systems directly, so his work on the problem was more independent, reporting back to ***Admin*** his findings and suggestions for a solution. In these conversations, various representations of system state, including error numbers, configuration file entries, and portions of log files, were exchanged over the telephone, instant messages, and email. As information was transferred, it was often transformed from abstract descriptions, such as `internal-port`, to specific names and numbers, such as `7137`, and vice versa.

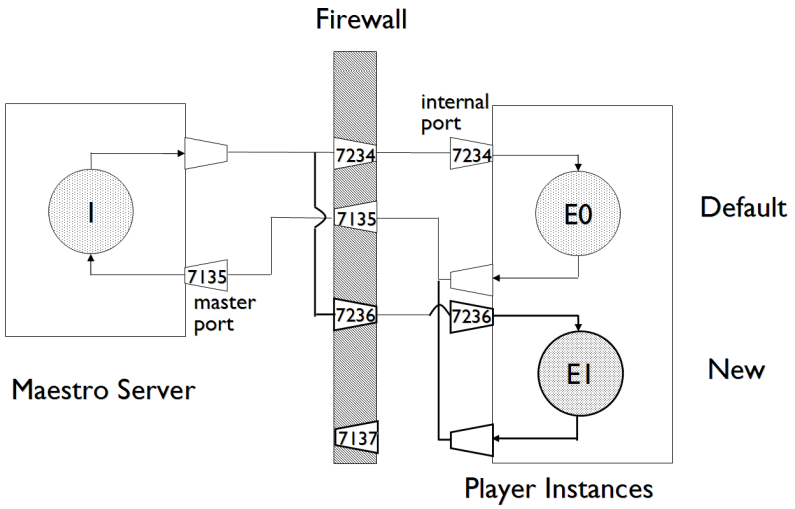

Fig. 4. In the final, working configuration, the maestro server communicates with the second player instance on port 7236 (maestro to player).

The problem was resolved after two and a half hours by ***Admin*** and ***Colle***. The core issue turned out to be a misunderstanding of the meaning of `internal-port`, as specified when creating the new player instance. The `internal-port` is used for communication from maestro to player—that is, the port on which player "listens" for data from maestro. ***Admin*** had originally believed that `internal-port` was used from communication in the opposite direction: from player to maestro. Because ***Admin*** had asked the network team to configure the firewall to allow communication in the wrong direction, the problem was an inconsistancy between the firewall rules and the player's port specification. The solution was either to change the firewall rules, or to change the player's ports. ***Admin*** eventually chose the latter solution (specifying the player's `internal-port` as `7236` rather than `7137`), as shown in Fig. 4.

Communication from all player instances to maestro was handled on a single standard port (`7135`), which was specified in the player configuration file as `master-port`. This specification was overlooked by ***Admin*** (in part because it was located in a part of the configuration file very distant from `internal-port`). ***Admin*** had incorrectly believed that a different port number was needed for each player communicating with maes-

tro, and that port was specified by `internal-port` (e.g., `E0`). ***Admin*** knew the original player used `7135`, so he thought that the new instance (e.g., `E1`) needed to use a different port, `7137`, instead. In fact, maestro has only a single `master-port` for receiving messages from all players.

At the center of the problem were specific transformations carried out by people and computers on various representations of system state. The various port labels `internal-port` and `master-port` do not inform about the purpose of these ports. Certainly, they do not indicate the direction of communication (e.g., `player-receiving-port` would have been a better label). Our observations of ***Admin's*** interactions with the manuals and instructions suggest that they were too vague to help him understand the system state. The time taken to resolve the issue was affected by these transformations, as participants attempted to reach a common understanding of the semantics and syntax of system components and their representations. Overall, eight different people or groups were involved in these exchanges (see Fig. 5).

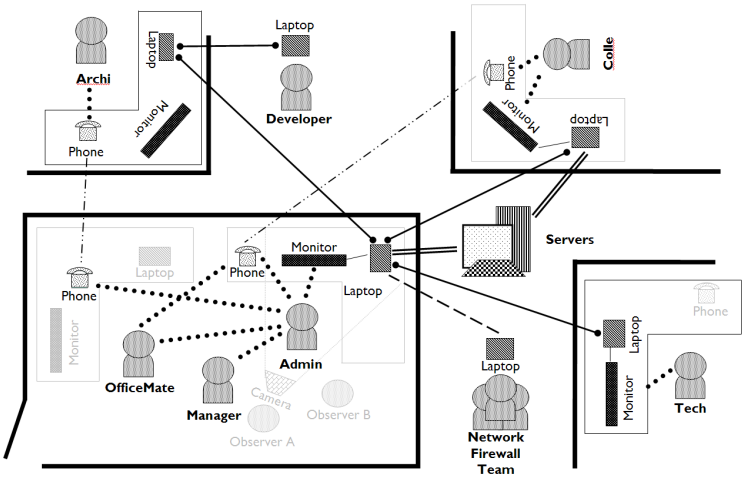

Fig. 5. Throughout the hours of troubleshooting, at least eight different individuals or groups participated. Single solid lines represent communication over the instant message channel; dashed lines represent communication over the email channel; dotted-and-dashed lines represent communication over the phone channel; and dotted lines represent communication via the air, whether visual or auditory.

We now turn to three specific interactions in more detail. These relate to problem diagnosis and problem resolution, illustrating how attributes of communication between participants influence how representational state propagates through the system and affects coordinated, joint activity.

3.2 Episode I: Do you have the manual?

After more than an hour of debugging by ***Admin*** and ***Archi*** failed to resolve the problem, ***Archi*** suggested a call to technical support. ***Admin*** was dubious, remarking that in his past experience technical support's solution to most problems was to, "reinstall, reinstall" the software. With no other option, however, ***Admin*** made the call and began to work with ***Tech,*** first by telephone, and later over instant messaging. Nearly two hours into the session, ***Tech*** and ***Admin*** exchanged the following messages [1:46:00]:[2]

> **Tech:** `Can you verify listen port 7234 or 7237 is listening?`[3]

At this point already ***Tech*** asked the right question that related to the source of the problem: *7234* was the listen port for the default instance, and *7137* was the listen port for the new instance (although ***Tech*** wrote "*7237*" he likely meant "*7137*"), as specified in the first command ***Admin*** executed (`m_web create E1 –m 7137`). ***Admin*** determined the listen ports using the command line, which displayed all listening ports on the external server including ports *7137* and *7234* and about twenty others. To himself, ***Admin*** muttered [1:46:35]:

> **Admin:** *7137 and 7234. This is the problem! Huh. Oh, no wait! Hmm, that should be fine.*

Admin might have realized that 7137 should not be a listen port, but he focused instead only on 7234 and filtered other information from the port list when reporting to ***Tech*** [1:47:10]:

> **Admin:** `It is listening 7234…is it ok that it listens on the same port as the default instance?`

[2] Timestamps in brackets indicate elapsed time from the beginning of the episode; in this case, 1 hour, 46 minutes, and 0 seconds in.

[3] Transcripts displayed in a `fixed width font` indicate communication via instant messages; *italics* indicate voice.

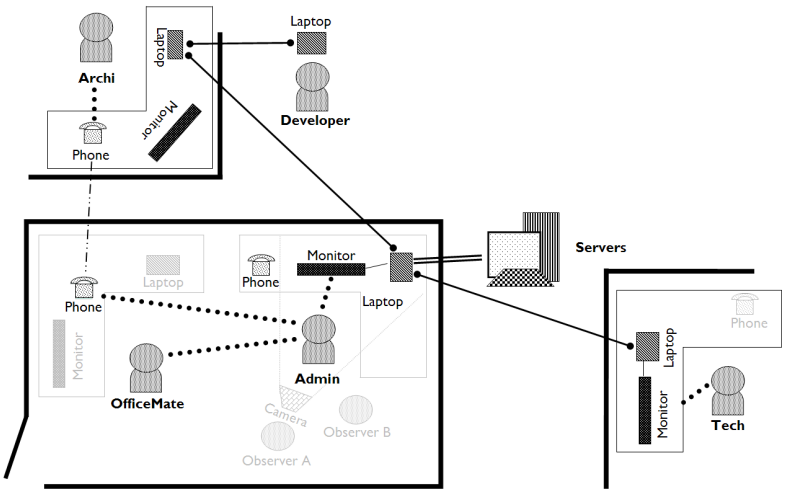

Fig. 6. In Episode I, *Admin* had sole access to the computer systems, and all information and changes passed through him.

Tech responded negatively, which might have led ***Admin*** directly to the solution [1:48:15]:

> **Tech:** Don't think so.
> **Tech:** Do you have the manual?
> **Tech:** I'm trying to find it... working from home today.

But on seeing these messages***, Admin*** spoke with ***Archi*** by phone [1:49:20]:

> ***Admin***: *You got to be kidding me! Oh God, this support guy is asking me for the manual.*

Archi told ***Admin*** that he knew someone else who could help, and eventually brought a developer into the discussion.

It seems clear that at this point ***Admin*** lost faith in ***Tech***. Yet just before asking about the manual, ***Tech*** had asked a question that would, in retrospect, have quickly led to resolving the problem (about which port was listening where). As Fig. 6 shows, after [1:50] ***Tech*** continued to send a few messages, but ***Admin*** only replied twice and then ceased communicating. In fact, after one of ***Tech's*** messages pointed out where to change the listening port for the new instance, and ***Admin*** responded verbally (to no one) [1:58:25],

***Admin**: This guy is totally useless.*

From this point on, ***Admin*** ignored ***Tech*** completely. The instant message windows that contained the exchange with ***Tech*** became covered over. ***Tech***'s last message arrived after a long period of no communication [2:13:00],

Tech: What is happening?

3.2.1 Analysis

All information in the discussion about system state passed through ***Admin*** (see Fig. 6). For ***Tech*** to help solve the problem, he had to get ***Admin*** to discover and report information about the state of the system. The flow of information between the new player instance and ***Tech*** was filtered by ***Admin***'s (mis)understanding of the system: what ***Admin*** reported differed from the results displayed on ***Admin***'s screen. For example, when ***Tech*** asked ***Admin*** to report the listening ports, ***Admin*** only reported back the port he thought was relevant. This kind of filtering is natural as the list was quite long and contained much irrelevant information. Yet just as ***Tech*** started to extract critical information from ***Admin***, ***Tech*** asked for the manual. From ***Tech***'s perspective, this can be seen as initiating a joint project with ***Admin*** to discover whether the new instance ought to be set up to listen on the same port as the old instance. From ***Admin***'s perspective, this appeared to be an inappropriate joint project, as he appeared to believe that ***Tech*** should simply know the answer without needing to refer to the documentation (especially given his dissatisfaction with prior advice from technical support). Thus, ***Admin*** did not take up the project.

The breakdown of the joint project can be analyzed as follows. As shown in Fig. 7, communication between ***Tech*** and ***Admin*** started over the phone, but moved to instant messaging after about 5 minutes. With instant messaging, messages tend to be short and lack the nuance and immediate interactivity of real-time voice conversation. ***Admin***'s response to ***Tech*** about the listening ports may have been terse because of the difficulty in copying all the information through the instant message system. When ***Tech*** asked for the manual, his intention was likely to double-check his knowledge. ***Tech*** already said that he did not think both instances could use the same port. Perhaps ***Tech*** wanted to point him to specific information in the manual regarding this. However, ***Admin*** assumed that ***Tech*** needed the manual because of a lack of knowledge. In this case, limitations of the communication media negatively influenced the effectiveness of the collaboration.

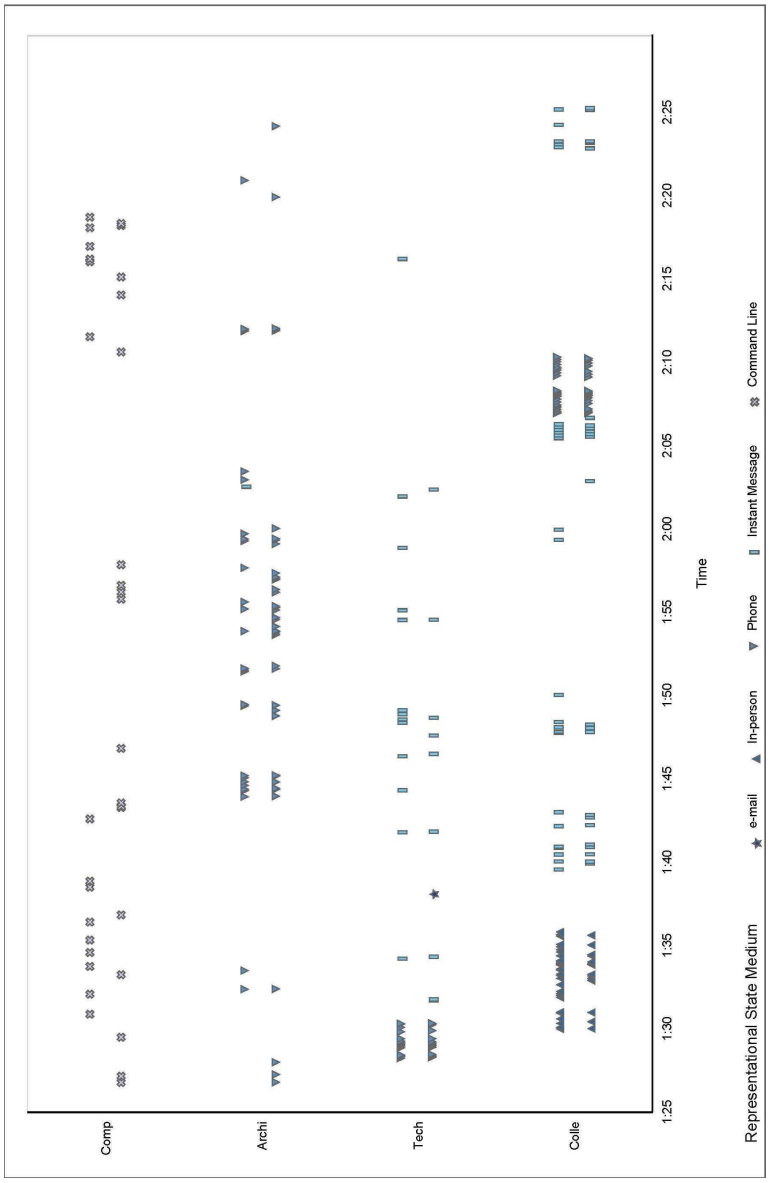

Fig. 7. A closeup detailing exchange of representational state between *Admin*, the computer, *Archi*, *Tech*, and *Colle* during the final hour of the troubleshooting session.

3.3 Episode II: What are you talking about?

After communication with ***Tech*** broke down, a series of exchanges between ***Admin*** and ***Colle*** eventually led to the resolution of the problem. ***Colle***, a close colleague of ***Admin***'s**,** was told by the customer relationship manager to help ***Admin***. As shown in Fig. 7, ***Colle*** checked with ***Admin*** in person (walking into the office and discussing the issue with him) about one hour into the session, and stayed in contact with ***Admin*** on and off via instant messages. ***Colle*** worked in the next office, where he used his laptop computer to access the same servers. Eventually, ***Colle*** discovered that maestro was trying to communicate with player over port 7137 [2:02:15]:

Colle: We were supposed to use 7236. Unconfigure that instance and …
Admin: Can't specify a return port… you only specify one port

Admin's response indicates that he did not know how to specify the port connecting maestro to player. ***Colle*** explained how he came to this conclusion (to use 7236 rather than 7137) by pasting into instant messages the commands he ran to test communication from internal to external server, attempting to persuade ***Admin*** that he was correct. The exchange became more heated [2:02:20]:

Colle: You specified the wrong port.
Admin: No, I didn't.
Colle: You did it wrong. Yes, you did. You need to put in 7236.
Admin: we just didn't tell to go both ways. The other port has nothing to do with this.
Colle: Well, all I know is what I see in the conf file
Admin: we thought that was the return port. That is not a return port.
Colle: there currently is no listener on <internal-server> on 7137. So use 7236. DO IT!

Admin, obviously frustrated by the exchange, called **Colle** on the phone [2:03:45]:

Admin: *What are you talking about? 7236?*
Colle: *Yeah?*
Admin: *We thought that it came in on 7137 and went back on 7236, but we were wrong, that 7236 is like an HTTPS listener port or something?*
Colle: *It will still come in on 7135 to talk to maestro server apparently...*
Admin: *right?*

Colle: *What's happening is it's actually trying to make a request back, um, through the 72... well actually trying to make it back through the 7137 to the instance...*

Colle: *..and it's not happening.*

Admin: *I know. I know that. But I can't tell it to...*

Colle: *.. just create it with the 7236. Trust me.*

Admin: *Why? That port's not, that's going the wrong, that's only one way too.*

Colle: *Trust me.*

Admin: *It's only one way. Do you understand what I am saying?*

Colle: *Cause it's the maestro talking back to the player server instance.*

Admin: *Yeah, but how does the player instance talk to maestro to make some kind of request?*

Colle: *7135 is the standard port it uses in all cases. So we had it wrong. Our assumption on how it works was incorrect.*

Admin: *All right, all right.*

Colle: *If it doesn't work you can beat me up after*

Admin: *I want to right now. (Laughter on both sides).*

3.3.1 Analysis

In this case, it was not necessary for all information in the discussion about system state to pass through ***Admin*** (see Fig. 8). Because ***Colle*** had access to the same systems as ***Admin*****,** he could examine system state directly (as ***Colle*** said, "`all I know is what I see in the conf file`"). From a distributed cognition perspective (Hutchins 1995), the movement of representational state among ***Admin***, ***Colle***, and the various computers offered both ***Admin*** and ***Colle*** different views onto system parameters and possible problems than those seen in the first episode with ***Admin*** and ***Tech.*** Here, the resources (systems) ***Colle*** could access gave him an independent view onto system configuration.

Communication centered on ***Colle***'s instructions for solving the problem by configuring the new player instance to use a different port. When ***Admin*** did not immediately take up the project to fix the port settings, ***Colle*** gave evidence supporting his solution. ***Colle*** shared the commands that showed him which ports were listening. Again, ***Admin*** did not take up the project proposed by ***Colle***. When the conversation shifted from instant messages to phone (see Fig. 7), ***Admin*** finally accepted ***Colle***'s project to change the port settings, but only after ***Colle*** stated that their understanding of how the system worked had been incorrect all along. Because ***Admin*** was upset, ***Colle*** made a special effort to appease him by jokingly agreeing to be physically harmed if his hypothesis turned out to be wrong. In both

admitting prior misunderstanding and joking, ***Colle***'s discourse was not about the business at hand, the establishment of common ground about the state of the system. Rather, ***Colle***'s statements served a different communicative function: establishing a different joint project that would enable ***Admin*** to follow ***Colle***'s directions. ***Colle*** found that rather than debugging ***Admin***'s knowledge of the state of the system (repeatedly explaining what the port settings should be), he had to debug ***Admin***'s model of the system itself (explicitly stating "our assumption was wrong" about the direction of the ports).

In Clark's (1996) terms, we can view discussion of *system state* and discussion of *understanding of system state* as operating at different layers in the discourse. When using the messaging medium, ***Admin*** and ***Colle*** worked mainly at the bottom layer, discussing system state and passing back and forth parameters of system state. When communicating by phone, ***Admin*** and ***Colle*** worked both at the bottom layer and above it, discussing both system state and their understanding of system operation. Moreover, when ***Colle*** said, "*If it doesn't work, you can beat me up after*," the discussion moved to yet another layer, this one ironic or joking, pro-

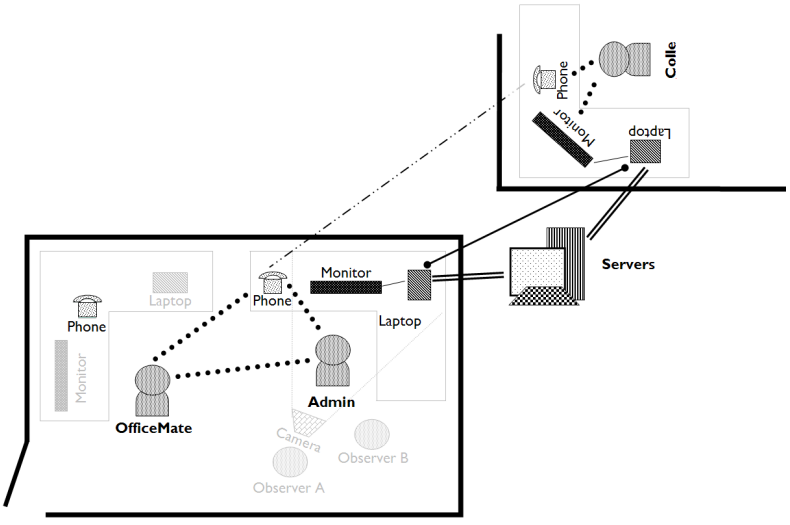

Fig. 8. In Episode II, *Colle* had independent access to the computer systems.

jecting a future in which ***Colle*** does not really believe ***Admin*** will physically harm him, and which served to lighten the mood. In terms of resources, it seems the message medium and the phone medium afford strikingly different discourse attributes. In this case, the phone enabled the participants to move rather easily among layers of the conversation, switching joint projects, and accomplishing their ultimate goals---whereas instant messaging did not enable such easy shifting among layers and projects, and seemed to stifle useful discussion.

3.4 Episode III: I've got too many people annoying me!

Throughout, ***Admin*** maintained multiple channels of communication (phone, email, instant messages, and face-to-face) with others. ***Admin***'s information environment was filled with many demands for his attention. One striking instance occurred near the end of the session. By this point, both ***Colle*** and ***Archi'***s developer friend had suggested the same root cause, and ***Admin*** had agreed to the try the solution. ***Admin*** and ***Colle*** spoke by phone [2:05:60]:

Colle: *Actually, you can create a new one.*
Admin: *Yeah, that's what I'm gonna do. (sighs)*
Colle: *I'm telling you man, this is what's happening. You can see by the connection it's trying to make. There is no 7137 listener on maestro right now, so what is it going to try to connect to?*
Admin: *Yeah, I understand what you're saying.*
Colle: *You know sure, we can see this in the logs, but I think we're already there where we've found out what the issue is.*
Admin: *All right, all right.*
Colle: *It's trying to make a return port.*
Admin: *All right!*
Colle: *I verified in the other player log that the…*
Admin: *Can you hang on please!*

Admin put ***Colle*** on hold and spoke out loud [2:10:15]:

Admin: *I can't, I can't think because I've got too many <expletive> people annoying me… There's too many people. I hate when there's too many people involved, and everyone's telling me to do something different and it's like you can only do one thing at a time, you know.*

After following ***Colle***'s instructions to solve the problem, ***Admin*** attempted to explain the process to ***Archi*** by phone [2:20:15]:

> ***Admin***: *All right I think we got it. What we did was, uh what did we do? The, uh, rather than specifying the 7137 port, that, cause...What happened was we had opened a port going to... We were under the impression for some reason that the port that player talks to maestro over is 7137 and then maestro returns on 7236, or 7135 and 7234, whatever. That was the impression we were under, so we opened the firewall ports with, um, and we opened it for 7137 to go from player to maestro and then 7236 to go maestro to player, so we only needed to open one port because, uh, and the port we needed to open was the one that maestro goes back to player on, so we already had that open, but it was the 7236 port so we just, I created the new instance specifying that as the port, so in the -m option I specified 7236 and I created all the junctions and everything looks cool at this point.*

3.4.1 Analysis

Colle coached **Admin** through the process of fixing the port settings (see Fig. 9). But in the end, **Admin**'s explanation was confused, suggesting he actually had little understanding of the details. The movement of representational state was from **Colle**'s screen to **Colle** to **Admin**. Yet there may have been too many information resources vying for **Admin**'s attention, leading **Admin** to put **Colle** on hold to execute the plan undisturbed (see the period between [2:10] and [2:20] in Fig. 7 where interaction is between

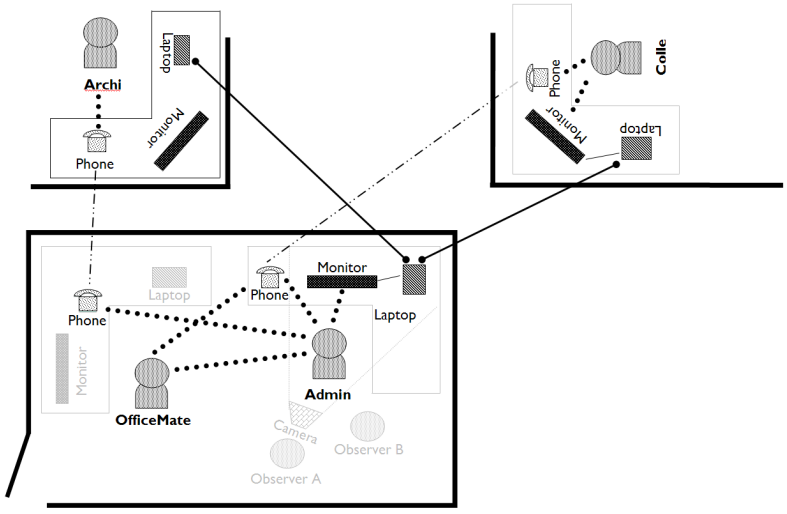

Fig. 9. In Episode III, *Admin* discussed configuration with both *Colle* and *Archi.*

Admin and the computer exclusively). To accomplish the job, **Admin** relied on memory of what **Colle** had said, commands **Colle** had sent via instant messages, and the manual to execute the command needed to create a player instance with the correct port number.

4 Results and Discussion

Our administrator (***Admin***) spent two and a half hours coordinating information from various sources to transform the initial configuration (Fig. 1) into the final configuration (Fig. 4). He coordinated information from many other people, from many configuration files and log files, from the output of many commands typed on the command line, and from many online documents including web pages and email (Fig. 7). We have sampled only a few of these interactions. Nevertheless, the story that emerges is one of how constraints on movement of representation and how attributes of communication media and content influence what information is attended to, transmitted, and used.

Consider first the interactions with technical support (***Tech***). As described, the support person was in fact on the right path to the solution when he asked our administrator to verify the listen ports. For his part, our administrator executed the commands to verify the ports. In examining the propagation of representational state (Hutchins, 1995), we find that the admin did not faithfully transmit all state information back. He focused on 7234, though he saw and mentioned 7137 as well. It appears that he filtered what he transmitted according to his incorrect understanding of the port data-flow direction.

According to the theory of language use as joint activity (Clark 1996), we can suppose that at the highest level, the administrator and technical support were engaged in a joint project to find and fix the problem with the new player instance. Subordinate to this was the project to establish common understanding of which ports were listening on the maestro and player servers. Note that only the administrator could determine which ports were listening because only he had access to the actual systems. Technical support attempted to draw out the relevant information by asking about the ports. However, when technical support initiated the project to obtain information from the manual, the administrator did not take up the project. Almost all useful communication between them ended at that point, as it seems the administrator did not see this as worthwhile.

The administrator's interactions with technical support and the architect (***Archi***) involved joint projects to determine, understand, and fix the problem; yet the administrator performed all diagnostic and repair operations. This contrasts with the administrator's interaction with his colleague (***Colle***), in which the colleague could access the system independently. As shown, the colleague was confident of his understanding of the problem and of the path to solution, but his repeated pleas for the administrator to simply perform the operations were ineffective. In this case, it seems as if the administrator understood the joint project with his colleague to be similar to those with technical support and the architect: the establishment of mutual understanding so as to develop a solution together. It seems the colleague, however, understood the joint project to be the solution of the problem itself. Sensing this mismatch, the colleague resorted to explanations in the form of commands to be run, his increasing agitation expressed in capital letters and exclamation points in instant messages. Once the conversation switched to the phone, further explanation attempts were made. Here is where the colleague seems to have realized a further mismatch: rather than a mismatch in knowledge of the various ports settings, he realized that the administrator did not have a correct mental model of the system with which to understand the details of the ports. To debug the administrator's model of how the system was put together, the colleague merely stated that their initial understanding had been wrong. Only at this point did the administrator begin to engage in the project the colleague had been proposing all along, changing the port settings.

The joint project of fixing a problem was accomplished without establishing common understanding about many technical aspects of the situation, and shifted between layers, focused sometimes on the system and sometimes on the understanding of the system. Movement of representational state about computer system parameters, whether correctly or incorrectly expressed, moved among participants but did not affect the actual computer system until representations of the entire configuration itself were conveyed. Solving the problem required participants to coordinate activity around *system model* rather than around *system parameters*. The telephone (as medium) enabled this change in coordination whereas text messaging did not. Discourse by telephone had a different character than discourse by text messaging: telephone resulted in give and take and shifting of projects, whereas messaging resulted mainly in opposing positions. That is, the rich interaction afforded by the telephone enabled participants to coordinate information not only about the business at hand (setting the parameters properly) but also about deciding what to do (debugging the system model).

5 Conclusions

Support and maintenance of large-scale computer systems is rarely done by one person working alone. Given the size and complexity of systems, many people with different expertise and skills are required to work together to keep systems running. Yet the establishment of common ground among participants in these tasks requires not only transmission of technical information but also establishment appropriate coordinated activity (joint projects) and management of attention. Our analysis suggests that information propagation is moderated by whom or what people pay attention to, which in turn is moderated by discourse attributes influencing project initiation and uptake.

6 Acknowledgments

We thank Anna Zacchi who contributed to the field study, Christine Halverson and Jeanette Blomberg who commented on an early draft, and the system administrators who participated in our studies.

References

Anderson E (2002) Researching system administration. Unpublished doctoral dissertation. University of California, Berkeley.

Clark HH (1996) Using language, Cambridge University Press, Cambridge, England.

Fairburn C, Wright P, Fields R (1999) Air traffic control as distributed joint activity: Using Clark's theory of language to understand collaborative working in ATC, In Proceedings of the European Conference on Cognitive Science.

Hutchins E (1995) Cognition in the Wild, MIT Press, Cambridge, MA.

Woods DD (1988) Coping with complexity: The psychology of human behavior in complex systems. In Goodsten HB, Olsen SE (eds) Tasks, errors, and mental models, Taylor & Francis, London.