Console Servers (T9)

Getting Up To Speed

With Console Services

version 1.6

David K. Z. Harris
zonker@bigbandnet.com

This presentation will be temporarily available at:
http://www.conserver.com/consoles/LISA2002-T9.pdf

This presentation is a supplement to my console services web pages located at **http://www.conserver.com/consoles/.**

These pages have a substantial amount of information noted below each slide. I do this to help minimize the amount of note-taking that you need to do in class, and this should give you more time to listen to the instructors.

If you feel that you learn better by taking notes, please feel free to do so.

This presentation is meant to be a a quick-start guide to Terminal Server and Console Server hardware, and to logging Console Server applications. We will look more deeply into the Open Source “Conserver” application , which is freely available from http://www.conserver.com, as an example for some of the discussions.

We will also explore some real-life deployments, to explore some of the points made during the talk.

About BigBand Networks

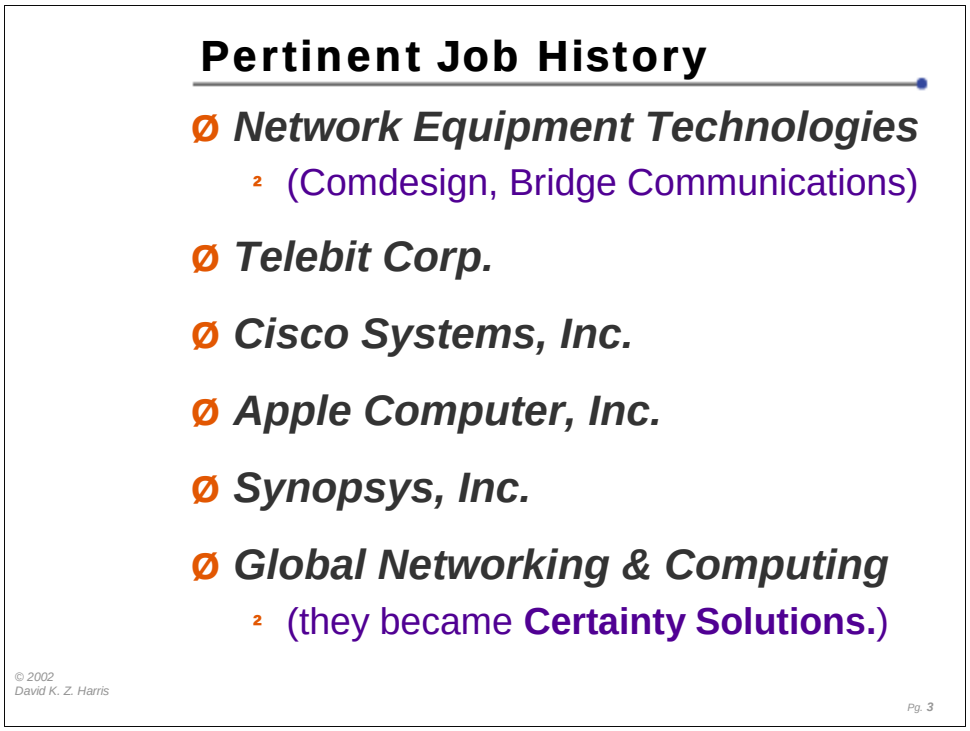
BigBand Networks makes Digital Video grooming hardware for Cable and Satellite companies, to help them make better use of the bandwidth they now have.

Essentially, the hardware and software produced by BigBand Networks can aggregate information on a broadband network, turning a bunch of allocated frequencies into one “Big Band”, allowing the users to spread their data across many channels.

The serial consoles on their “BMR” chassis run at 115 Kbps, and are VERY verbose. The output is ideal for realtime system monitoring, and statistics gathering, but would be a horror to log for any significant amount of time.

While the chassis normally uses SNMP for management, the serial console also regularly streams information and updates. Administrators can use a Command Line Interface to manage the devices, and automated configuration can be done using ASCII text uploads, or send/expect scripts, etc.

If you are interested, you can find more information, product information, and white papers at http://www.bigbandnet.com/



Before moving into networking, I was a hardware hacker, working in repair and R&D roles. I have been tinkering with serial devices for more than a decade.

My experience, plus reading plenty of manuals, has taught me that there are a few safe bets that you can make when working with unknown serial devices. I've also learned a few tricks that make the job of connecting serial devices to terminal servers easier. I'll share these tricks throughout this presentation. You can also find some good, basic clues on my "Minor Scroll of Console Knowledge" (http://www.conserver.com/consoles/msock.html)

I've also been testing terminal server and console server hardware for a number of years, and I've posted some information about serial console remote access to the web that others have found useful. These pages are posted at my console info web site (http://www.conserver.com/consoles/)

This tutorial is a 'high-speed, low altitude pass' over the topics involved in deploying Remote Access to the serial consoles around a large environment. Due to the time available, we will sacrifice some of the depth, in order to get more breadth covered during this class. The intention is to give the attendees a wide variety of information, and look at the costs, benefits, and justifications for such a deployment. We will look at the hardware vendors, as well as explore the options for adding logging servers, and integrating the console access into real-time system and network management tools. We will also explore some large deployments, as they will highlight the topics within the class.

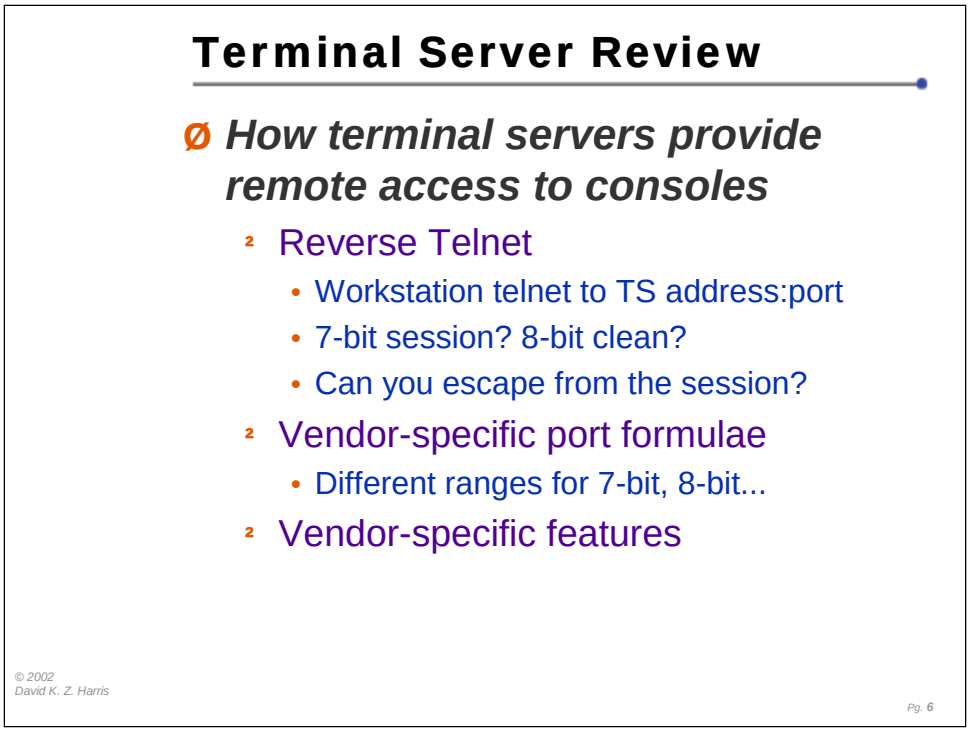
The "Advanced Remote Console Access" tutorial will go into more depth, especially related to the hardware involved. It will also explore other options for console server applications, both commercial and open source.

Neither class will be covering physical-level debugging very deeply, due to the length of time involved. However, I'll be happy to discuss those issues after class, or at the Conserver BoF on Wednesday night.

There are many interesting sites and applications for Conserver. During this tutorial, we'll look at a few current deployments of Conserver. The examples have been picked to showcase certain features, and to reinforce particular points I want to make in this tutorial.

The sites well look at are large deployments. Implementing even a small deployment isn't cheap, but there are significant benefits to be gained.

The primary cost to start a remote console access deployment will be the console server hardware. Adding a logging server, and integrating your console logs with your system and network management tools add significant value to the hardware deployment.



Originally, modems or ‘dumb terminals’ were connected to terminal servers, and users would telnet from the terminal server to other points around the network.

Today, most terminal servers allow you to open a socket-based connection to the IP address of the terminal server, but at a high TCP port number, to connect to a particular serial port.

Some vendors allow only 7-bit sessions, while others provide the option for full 8-bit sessions, and even “non-escapable” sessions (where the attached device needs to drop the DCD or hardware handshake lead to disconnect you session).

The list below tells you the formulae to determine the TCP port number for two of the more popular terminal servers (where ‘n’ is the line (serial port) number you wish to connect).

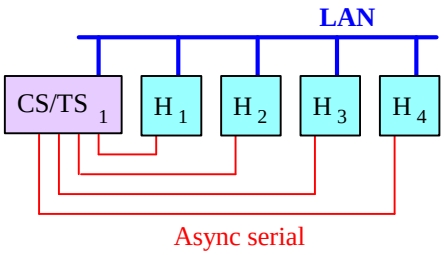
Cisco: 2000 + n

Xyplex and iTouch: 2000 + (100 * n)

IOLAN: 10000 + n

Basic Serial Hookups

- Console Server connected to the same LAN with the hosts
- Serial connections from the consoles of each host to the Console Server

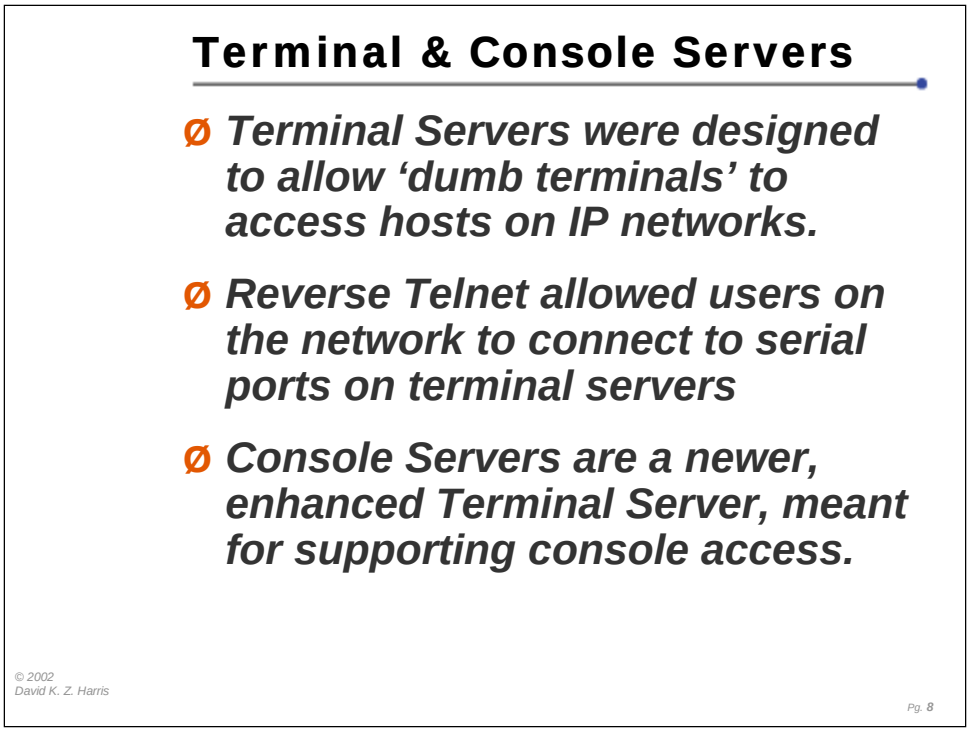


In the most basic configuration, you connect all of the consoles that you care about to a serial port on a Terminal Server or a Console Server.

Each serial port can be configured for different serial speeds (that is, for any of the common serial speeds), as well as setting the number of bit, parity, and stop bits.

In a simple, isolated deployment, the console server would be connected to the same ethernet network as the hosts. (We'll discuss other connection options later in the tutorial.)

The console server could be a host with a multi-port async module, or a dedicated Console Server device. If the console server is a host, with it's own keyboard and mouse and screen, it may not need to be connected to the network at all.

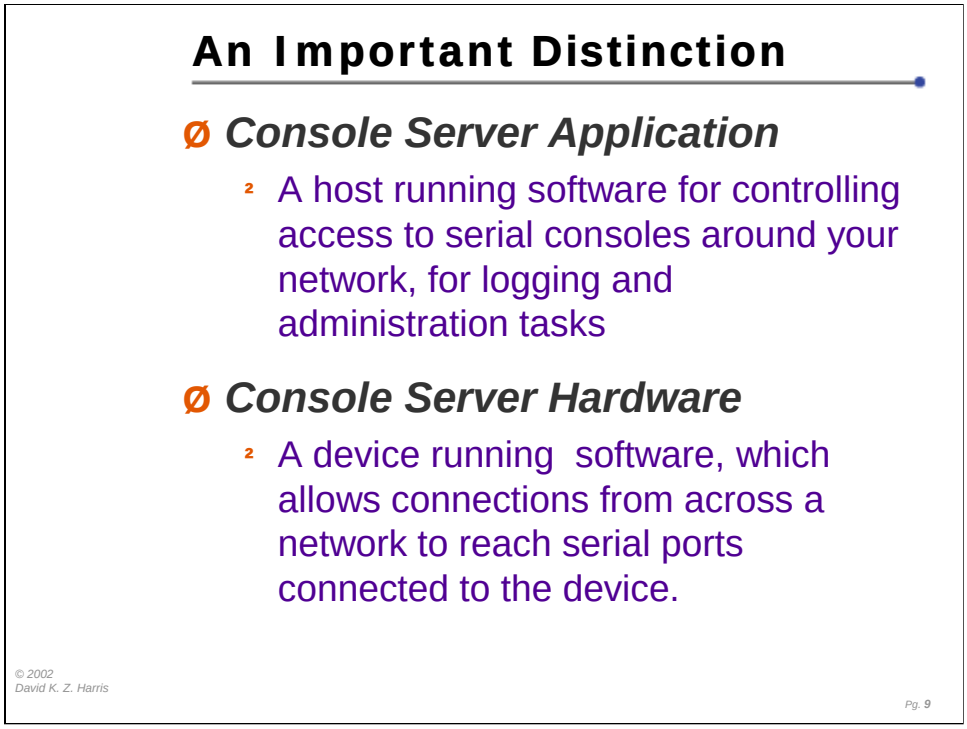


Terminal servers are still readily available, and you can pick them up fairly cheaply on eBay, or other dot.com auctions.

However, Terminal Servers didn't need to care about Serial BREAK in the old days, and sometimes it was designed to send BREAK when sessions cleared up (to reset a modem, for example). So an older unit may send Serial BREAK to attached devices under some conditions. If you have SUN computer consoles attached to your terminal server, a BREAK can halt your SUN.

Serial BREAK is actually a Good Thing, in moderation, and useful for administration of your hosts. You can get more information from my Serial BREAK-off testing pages, http://www.conserver.com/consoles/breakoff.html

Newer Console Servers are generally better about not sending Serial BREAK at the wrong time, but if you have SUN gear, you should make sure that the equipment you are buying is marked "Sun-safe", or otherwise states that it won't send Serial BREAK unless you want it to.



With the advent of “Console Server” hardware, some distinctions needed to be made. Without these distinctions, discussing Console Servers becomes a bit vague, and can become confusing for the participants.

Consider a host, running a Console Server Application, but also having a multi-port serial board, attached to some serial consoles on other hosts. It serves both of the purposes defined above, controlling access, and logging data, while it also moderates the connection from the Console Server Application to the hardware serial ports. I consider this a Console Server Hybrid.

Cisco has been taking remote access to consoles fairly seriously (Worldcom was very fond of the 3600 family). Cyclades has been in this market for more than a decade, and is a strong player. Digi and Lantronix are still doing well these days.

Computone had been on the ropes in the Point of Sale market, but then they decided to make terminal servers for that arena. This saved the company, but they needed to evolve their product. The economy didn't wait, and Symbiat bought them. You can't find terminal server hardware on the new web pages. They are now out of the hardware business. The shame is that the new products Computone was developing were coming out quickly, but not quickly enough. They also had the BEST Users's documentation I have seen to date.

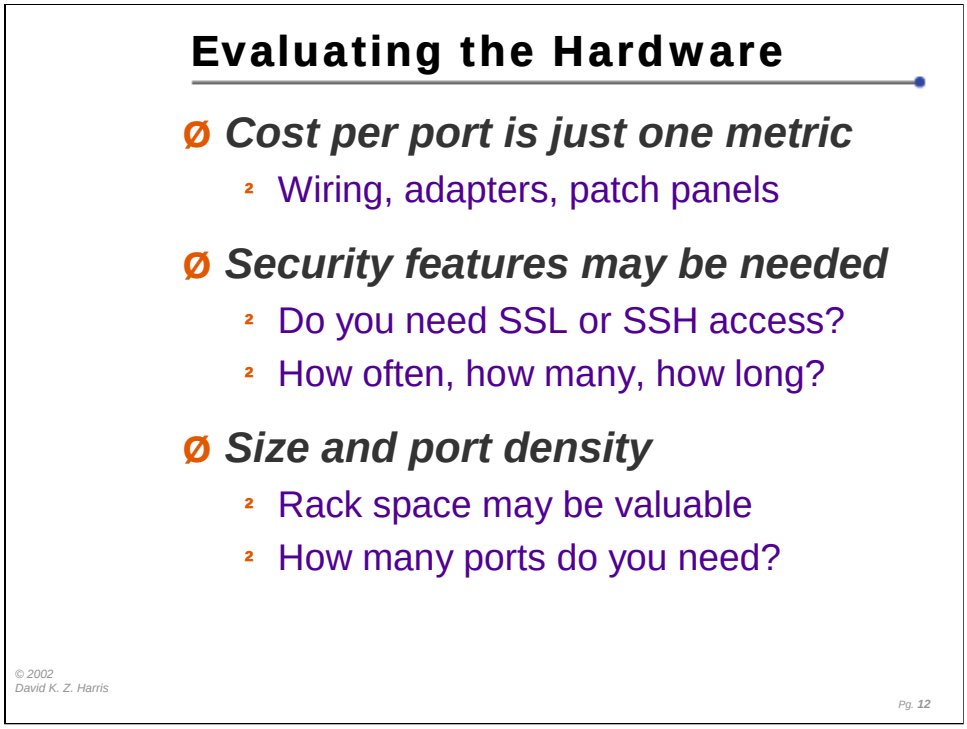
Xyplex all but died, and was acquired by nBase, who didn't evolve the product, although MCI was very attached to this product family. iTouch Communications took over the line, and continues to sell the older line, but they have evolved new products to be Sun-safe, and expanded the line! They will still support the older gear as well. MRV, the mother company of nBase and iTouch is changing the logo and color, but features will remain the same.

The (Chase Research) IOLAN line was acquired by Perle, who also picked up the Specialix line as well. You can still get the old IOLAN products, but the newer Perle CS line is a very nice family.

Synopsys had a working installation, based on Xylogics Annex hardware. Even though it sent Serial BREAK on power-off, they kept it. Even after the product line was bought by Bay Networks, they kept it, because they could still buy compatible hardware. Only when Nortel Networks bought Bay Networks, and killed off the product did Synopsys move to the costlier Cisco hardware, and then they retrofitted the Cisco gear across the entire enterprise, replacing all of the older units.

Many folks have been buying older units cheaply, on eBay. But the units didn't come with their software media (floppy disks, or PCMCIA flash cards), so they wouldn't boot. If you don't get the software, and manuals, the unit probably costs too much!

iTouch Communications will still support older Xyplex terminal servers, for the cost of a support contract. Ask yourself how much your hardware is worth if it doesn't work…and how much is it worth to keep it working.

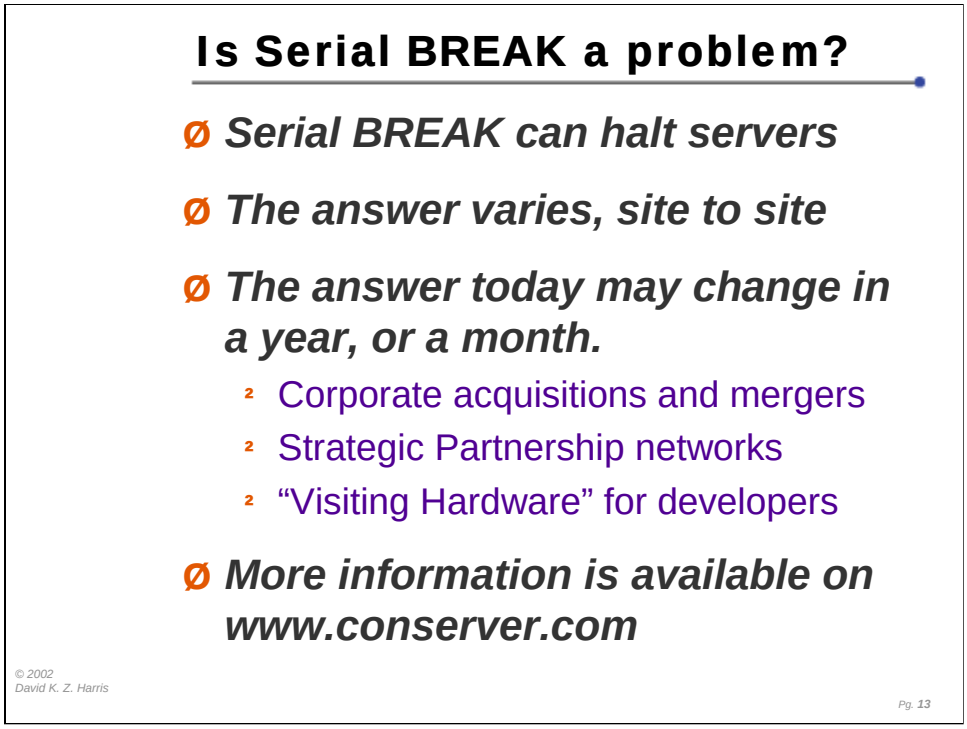


While many Console Servers now have SSH access capability, SSH V.2 capability has been slow in coming.

Most vendors blanche when you tell them you want to make 32 simutaneous SSH connections to their Console Server, and they'll ask "For how long?", indicating that their SSH may have memory leaks, or SSH sessions may severly tax the performance of the device. (I haven't done a large-scale test, with 32 busy async devices connected to a console server, using SSH to connect to all of the ports, but I also haven't found a vendor who wants me to put their gear up to that test, either.)

SSL access is also starting to be offered by vendors, but then you need to manage certificates.

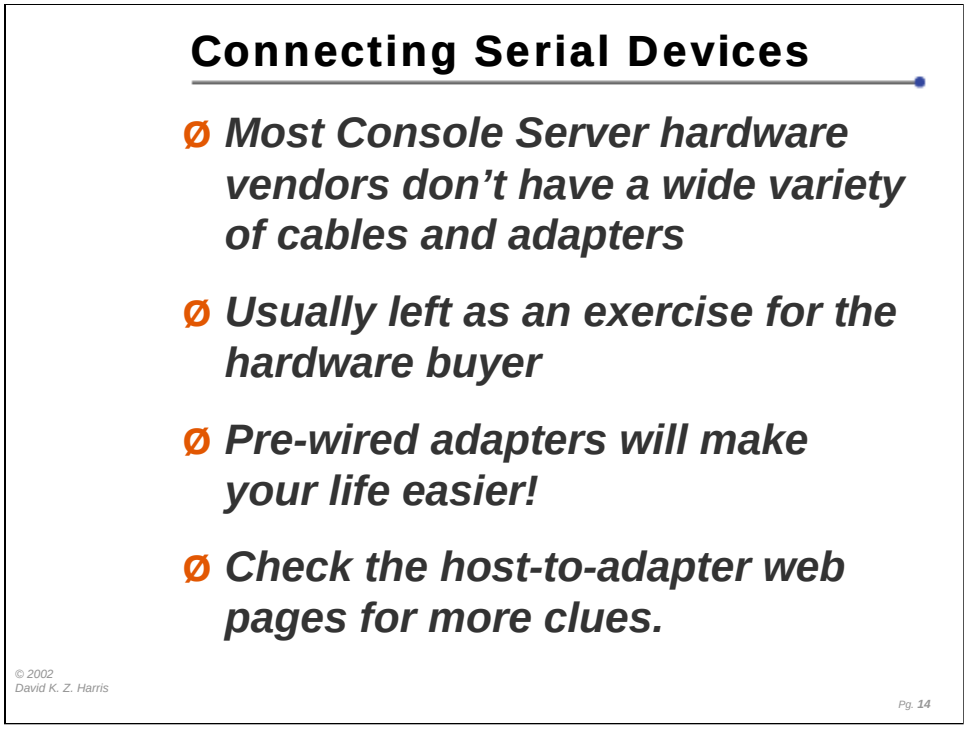
None of my vendors wanted to discuss where their SSH and SSL core code have come from. But, few Console Server devices have been listed on the SANS vulnerability alerts…



Older SUN hardware is vulnerable. So are older SGI IRIX machines. Newer SUN machines can be patched and modified. Modems may react badly to BREAK, depending on their configuration. Some telecom test gear is rebooted, and setting reverted by serial BREAK.

Newer console servers don't cause this problem. I've been testing a variety of terminal server and console server hardware, to see which devices send Serial BREAK without the operators instructions. You can find the testing results on http://www.conserver.com/consoles/breakoff.html.

If you only have one or two devices that are susceptible to BREAK, you may be able to connect them with Nu-Data Cisco Serial interfaces (cost is ~$100 per port). If you have more than a dozen ports to protect, you should consider purchasing a Console Server that won't send BREAK unless you tell it to.



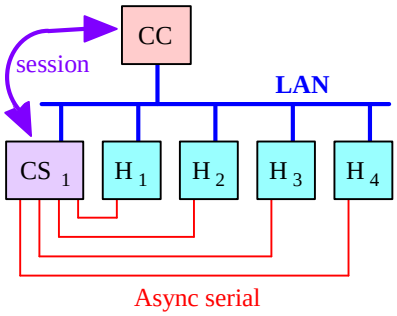
Please pardon what may seem like 'blowing my own horn', but I've compiled some web pages which are intended to make it easy to connect various devices to Console Server hardware. I've collected RS-232 signal information on more than 300 devices, and worked with Americable to have them produce and sell pre-wired and labeled adapters and cables for a variety of Console Server devices.

Go to http://www.conserver.com/consoles/, and select one of the four Console Connection Guides. Use the links to view the host-to-adapter guides, which tell you what adapters to use to connect any particular devices to the Console Server hardware you have.

I understand that Cyclades has also talked to Americable about making assorted adapters for their hardware as well. (I also understand that Cyclades provides up to six different adapters insome of their new units.)

Basic Architectures (#1)

- *Basic Reverse TCP sessions*
  - CS = console server
  - CC = console client
  - Hn = hosts



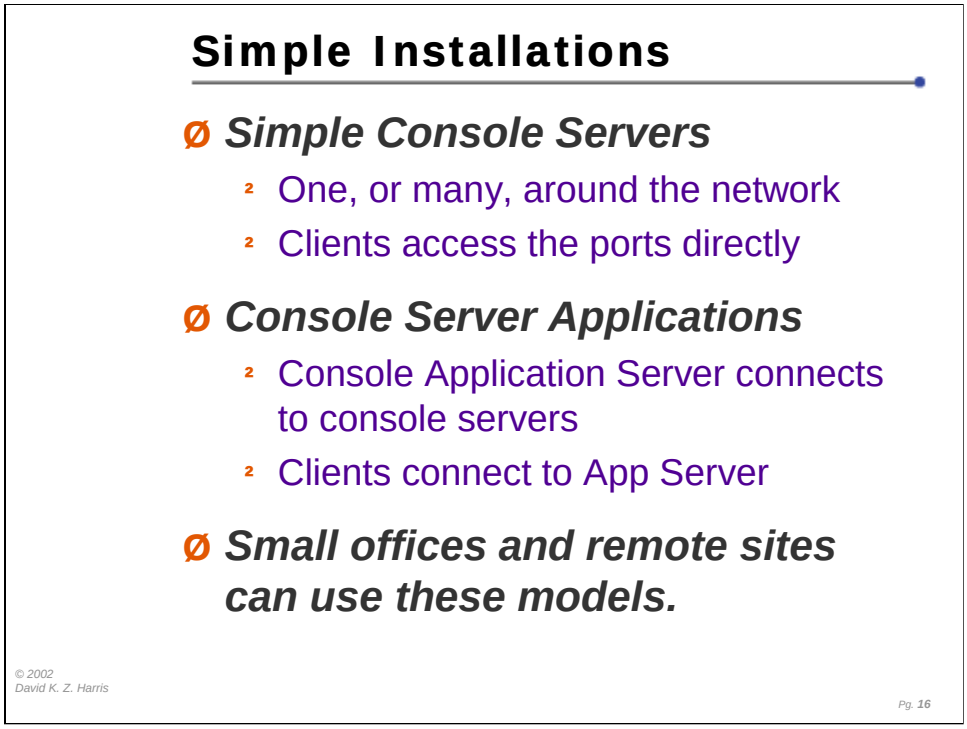
In this configuration, we presume that we don't worry about internal security.

In this case, the console server hardware (CS) has multiple serial ports connected to the various consoles on hosts that you care about.

In our example, you would use a telnet client on (CC) to connect across the network to the desired serial port on (CS). Anything you type goes to the serial console you are talking to, while anything coming back from that console is sent back to the telnet client.

If nobody is connected to a particular serial port, the data coming from the attached console is normally lost.

If CS is a logging Console Server Application, then data coming in from the consoles are logged, but the CC to CS sessions are similar.



The Console Servers will be the largest expense in your deployments, whether you use console servers, or hosts with multi-port access cards for serial access. Adding a logging Console Application Server can add value to that initial investment.

Smaller offices and networks can get away with a single Console Application Server.

Think About Security

- *Do you have a security policy that covers remote access to consoles?*
- *How concerned are you about 'internal' threats?*
- *What are you trying to protect?*
- *And what is that worth to protect?*

We will discuss some security issues in this talk, but security is a touchy subject, and most cases are unique in some aspects.

Due to this, we will discuss general points during the class, and the materials will give you some questions to think about and discuss.

There will be a BOF on Wednesday evening, and we'd welcome any additional questions there, if you are comfortable asking them in that forum.

The biggest worry when implementing remote access to your consoles is whether you are concerned with the console traffic being monitored within your network. (Most companies use a "jelly bean" security model…hard on the outside, soft on the inside…meaning that they are not worried about folks on the inside sniffing on the wires.

In a switched Ethernet environment, it's harder for folks to see the packets to and from the terminal servers but it's not impossible.

If you have a console server host, you should consider if it is worth making that host single-purpose machine, and limiting the login accounts.

Security Concerns

- *Which network will you connect your console server(s) to?*
- *Do you have a Security Policy to comply with?*
- *Extending your management network may add cost to your terminal server deployment.*
- *Is your secure network also 'physically secure'?*

Do you have an existing security policy? Does it cover access to serial consoles?

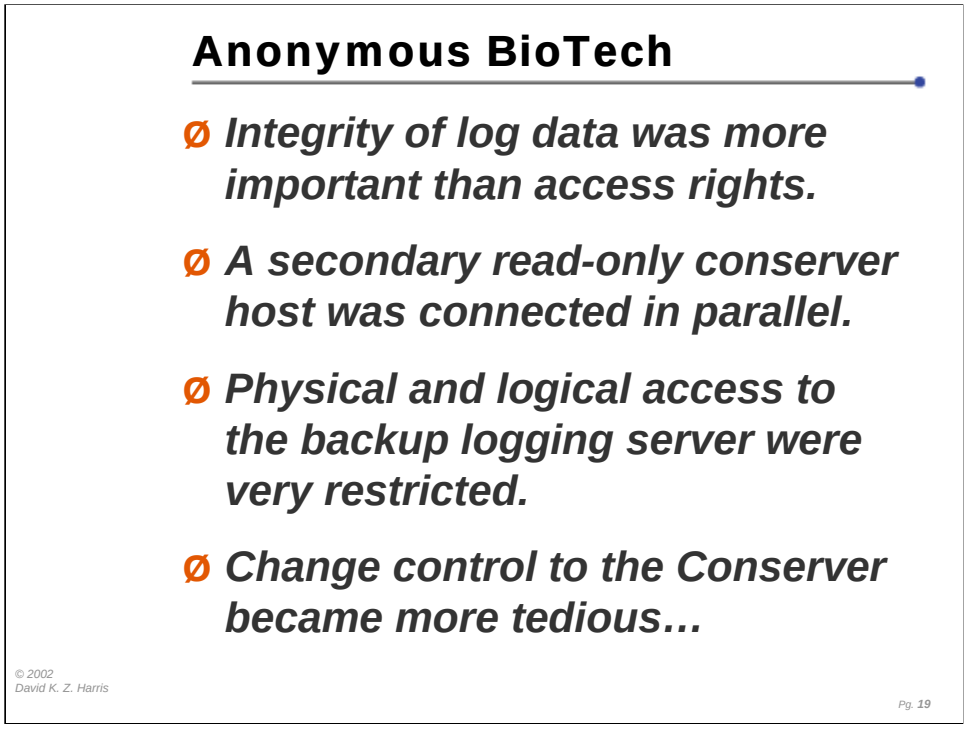
Do you have an existing management network? Where is it physically deployed? How hard would it be to extend it to other locations? (And is that expansion covered under your security policy?)

Physical access to the "secure" networks is part of your security scope. If you make it hard for someone to sniff your packets over the network, but don't prevent them from connecting to the network itself, you still have an exposure.

You can lock some ports down on certain Ethernet switches. That is, once locked down, the switch only recognizes the MAC addresses of existing devices. If someone connects without authorization, they will not get link, or see packets.

There is still the exposure that someone could bring a laptop into the data center and plug into the console, but then the logs in your console server would stop accruing. It would take more sophistication to splice into the serial line and monitor both sides of the serial conversation.

Again, you need to consider the value of the information (when you have the secrets, and 'they' don't), and the nature of the perceived threats. Only then can you look into solutions to the threats, and see if they are cost-effective.

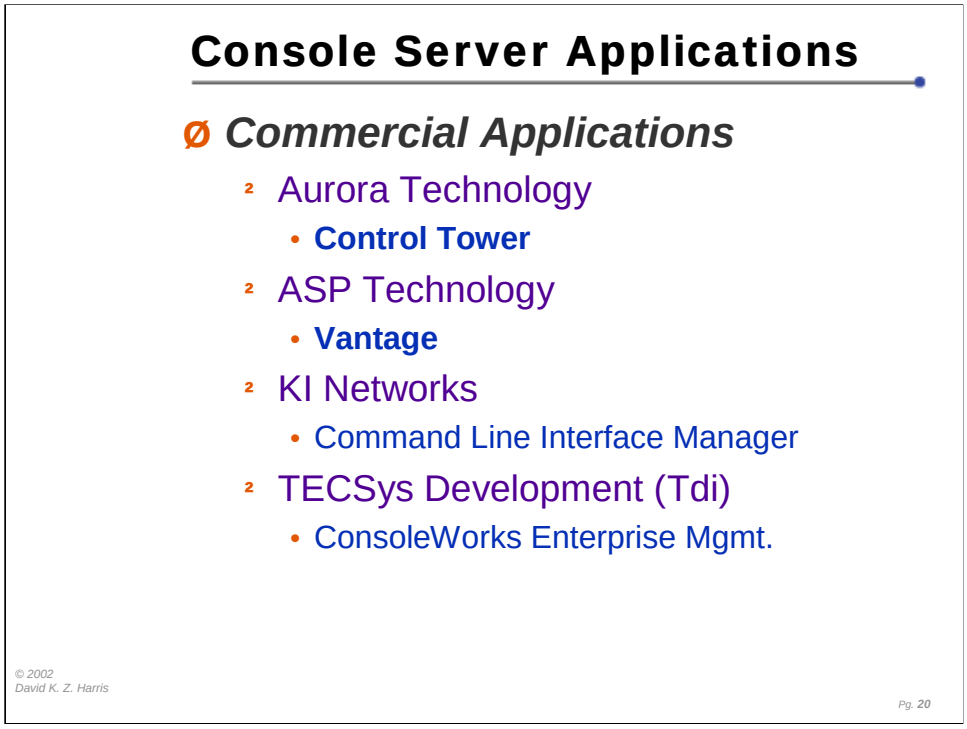


I cannot mention the name of the company, but this security twist was interesting to me, and I wanted to pass along the idea.

The patch panels for the primary conserver were kept in a physically locked cabinet, anchored down in the data center. The reason stated for this was the importance of data integrity on these logs. The connections from serial connections in the locked host cabinets came into this same cabinet.

Inside the cabinet, the ground, transmit, and receive data lines from the monitored devices came to an RJ-45 patch panel, as you might expect. But the ground and receive data (from host to Conserver) were also extended to a second RJ-45 patch panel, with different-colored jacks. There were then two sets of terminal servers in this same cabinet.

One set of terminal servers (the ‘read/write’ set), were connected to the network, accessible to the Conserver that admins used to manage the devices. The second set of terminal servers were connected to this isolated host, which watched all of the ports, but the admins couldn’t use this server, nor access the logs. In fact, few folks knew about the second machine.



In larger companies, there are usually supporters of Open Source applications, as well as advocates for commercial applications. The push for the commercial applications usually stems from wanting to buy a support contract, versus developing the expertise in-house. The push for Open Source is usually stronger in sites where they don't want to rely on external vendors for support, usually for critical infrastructure, or for applications where customization and/or fine-tuning will be needed.

Aurora Technology's Control Tower app has its roots in the RTTY code tree. It is well developed, and Aurora works to integrate the code with hardware for a stable deployment environment.

The ASP technology team has been working with Terminal Servers for a while, and they developed a hardware fix for some terminal server devices to fix the Serial BREAK problem.

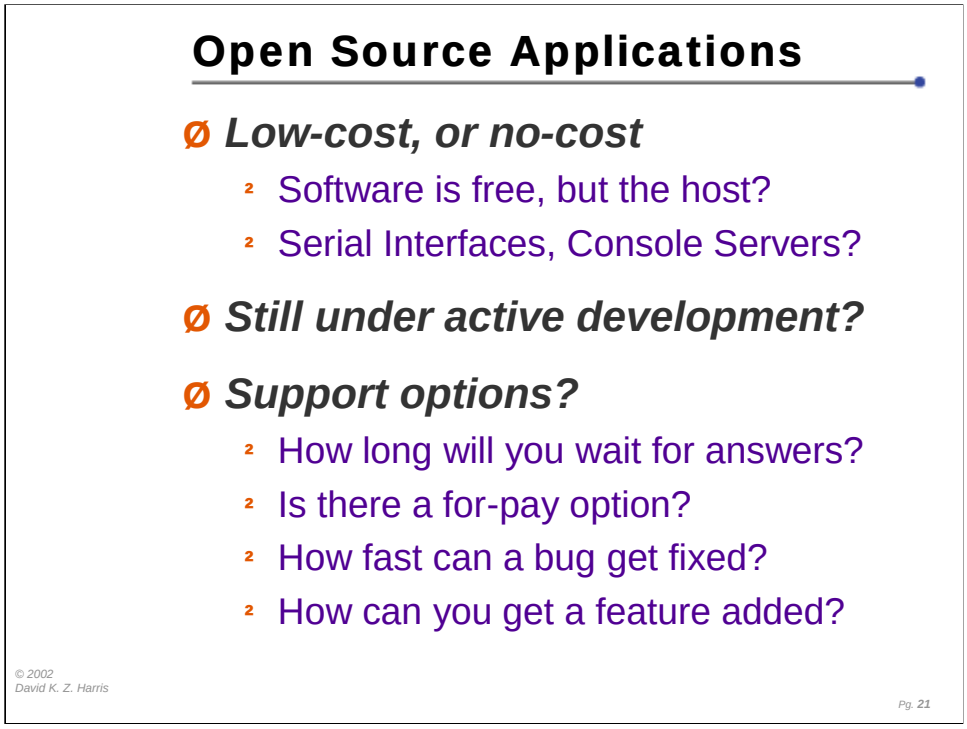
Both Aurora and ASP have a long record in the area of Console Server Applications.

Control Tower (http://www.auroratech.com/)

Vantage (http://www.asptech.com/)

CLIM (http://www.kinetworks.com/overview/CLIM-Consoles.html)

TECSys (http://www.tditx.com/)



Open Source is a great option, if you have some good coders in-house, who want to learn a new application (if you require customization), or if you are happy with the basic package(s).

Since many open source projects are distributed at little or no cost, these projects can sometimes go stagnant, or stop developing. Unless you want to take on code maintenance, make sure that your open source applications have a strong user community, or an active developer core. (Conserver meets both of these criteria.) An active user community can also be a part of the support effort, when used with a mailing list, or if the user questions end up in a good FAQ or knowledge base.

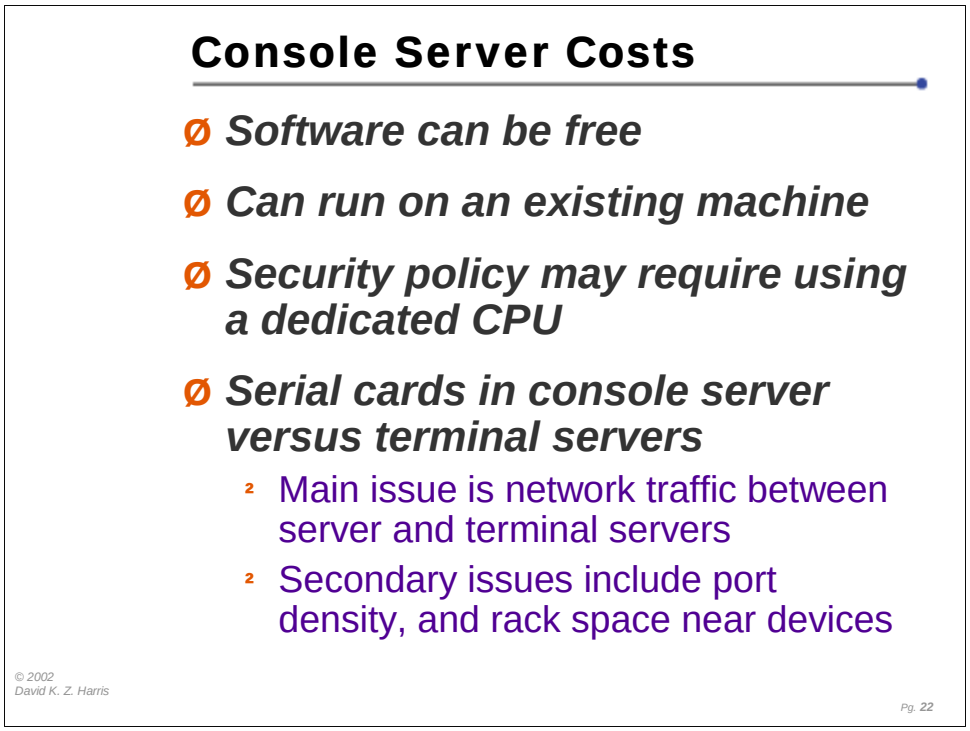
No matter whether you pick a commercial application, or use Open Source code, I encourage you to try at least one of these options, and test a small deployment on your network. See for yourself how useful remote access to your serial consoles can be at your site.

Keving Braunsdorf's Code: ftp://ftp.physics.purdue.edu/pub/pundits/conserver-8.4.tgz (not always working…)

Thomas Fin'e pages: http://hea-www.harvard.edu/~fine/Tech/console-server.html

Doug Hughes inexpensive Console Server recipe: http://www.eng.auburn.edu/users/doug/console.html

Paul Vixie's RTTY code: ftp://ftp.vix.com/pub/vixie/rtty-3.2.tar.gz



There are many free solutions available, depending on the features that you want. Most notably, free software often lacks a strong support offering, which means you need to be able to support your site with minimal help.

Our favorite free applications is the conserver application, from http://www.conserver.com/.

There are at least two commercial applications available to you, if you feel the need to have a pay-for support option.

You can often use free software on an existing server, as the needs of the software themselves do not require much memory or CPU. This can help offset the costs of a new terminal server deployment.

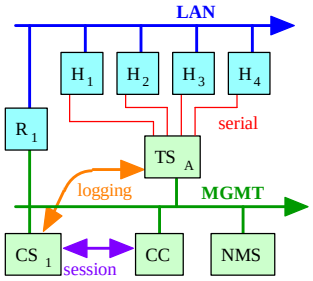
However, shared machines often mean security vulnerabilities in the other applications, which could allow unauthorized folks to see the logs (or tamper with them). A shared system often means more users that have legitimate access to the machine, which can be another vulnerability.

If security and integrity of the logs is a concern, you should consider purchasing a dedicated machine, installing a hardened OS, and limiting the number of folks who have login access to the CPU.

Adding multi-port serial cards to a machine increases power consumption, heat output, and (depending on the card) can also increase CPU load.

Advanced Architectures (#2)

- *Addressing Security Concerns*
  - Add a management Network
  - Put console server and clients there



If you are concerned about someone sniffing the client-to-server connections, or the logging streams, then you probably already have a control/management network in place, where your monitoring and control activities take place.

In this model, the console server Application host (CS1), and the client(s) all live on the management network, so that the client sessions, and the logging activity, only happen across the management network.

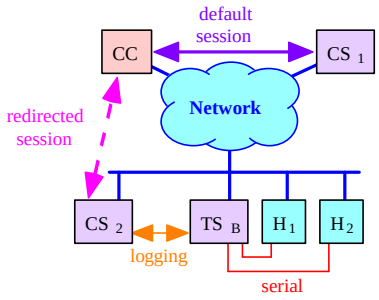
The Interop+Networld Show Net routinely used a separate management network (per Carl Zwanzig).

A good practice is to ensure that your management network is connected using switches, rather than hubs…this reduces the risk of someone sniffing the client-to-server console session traffic. (On a hub, anyone connected has the chance to hear whatever traffic comes through the hub, including the console traffic.)

Few console servers use SSL or SSH for the client-to-server connections, so these sessions travel in clear text. For this reason, if you are located outside of the management (or security) perimeter, you should consider making an SSH connection to a client host that is on the management net, and then making the client connection from there.

Advanced Architectures (#3)

- *Distributed console servers*
  - One master configuration file
  - Client is redirected from 'local' server.
  - Logging continues if WAN fails!



In the case of conserver, you can indicate in the configuration file which of a number of distributed conserver hosts is maintaining the logs for any particular device, as well as indicating which local serial port or remote terminal server is directly connected to each device. You can then push this master configuration file to all of the distributed conserver hosts.

Each client should try to connect with the conserver host that is closest to the client for all connections. (You can do this via hostnames, or even with a local host file entry.)

If a client (CC) connects to the local conserver host (CS1), but is trying to reach a device that is connected to another conserver host (CS2), the client is redirected to the other conserver host. The client session is then connected with the session of the desired device.

Whenever you have a distributed conserver implementation, you must be sure to append the "@server" tag to each console entry, otherwise each console server will try and manage the same console. We'll illustrate this feature in the slides to come.

www.conserver.com

- *Latest code*
- *Pointers to FAQ and mailing lists*

The Conserver.com website will always point you to the latest distributions, and you will find a link to the latest CHANGES file, which is a brief revision history for the Conserver application.

You can also find pointers to the users (support) and announcement (new releases) mailing lists, in case you are interested in subscribing.

The site has ht:DIG installed, to make it easy to search the site for useful information on the site.

You can retrieve the latest code via FTP or HTTP.

Conserver Background

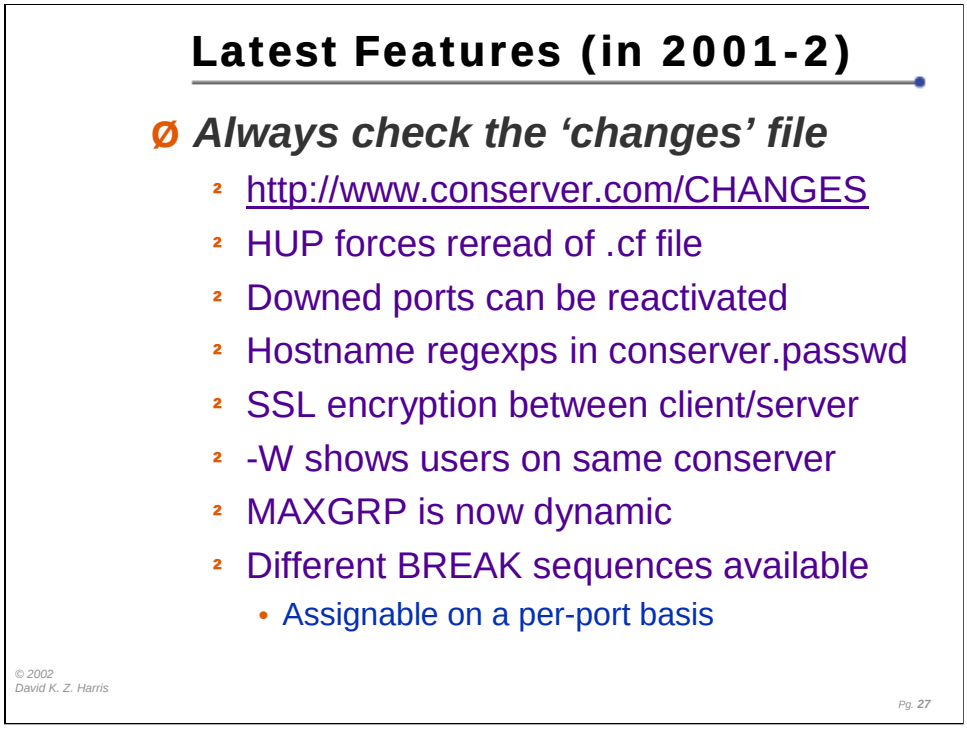
- *Presented at LISA in 1990*
  - Tom Fine – Ohio State University
- *Identical branch appeared later*
  - Kevin Braunsdorf – Purdue University
- *Further developments (TCP/IP)*
  - Robert Olson - Argonne National Labs
- *Code is written in C*
  - See the "Systems Tested" link on the conserver.com website

Thomas Fine was the principal coder for the original Conserver code. It was presented at the LISA conference in 1990, and it was a logging host, with multi-port serial adapters installed on the host. Kevin Braunsdorf took a branch from the code (and was still developing it in 2000). About 1993, folks at ANL added the ability to use a 'Reverse TCP' connection to terminal server ports across a TCP/IP network.

Surprisingly little documentation or references appeared between 1990 (introduction at LISA) and 1995, when Arnold de Leon (now at Microsoft) had deployed the code at Synopsys, with Bryan Stansell doing the updates and revisions desired by the company, and making this updated code available to others. (The deployment at Synopsys began early in 1993, with Arnold de Leon working with the code. In 1994, Bryan Stansell joined the team, and took over the code maintanance. In 1995, Arnold presented a write-up that included references to the version of code that they were using.)

In December 2000, Bryan Stansell and I presented a tutorial at LISA, looking at Conserver after a decade, and introducing mailing lists and soliciting users to come forward with suggestions, patches, and bug fixes.

Bryan Stansell continues to be the lead developer today, with help from Todd Stansell and Dave Stuit, and with a number of active Conserver users submitting their enhancements. 2001 was a very good year for Conserver, due in large part to the participation of the user community. (The CHANGES file in the conserver distribution documents who submitted which changes.)

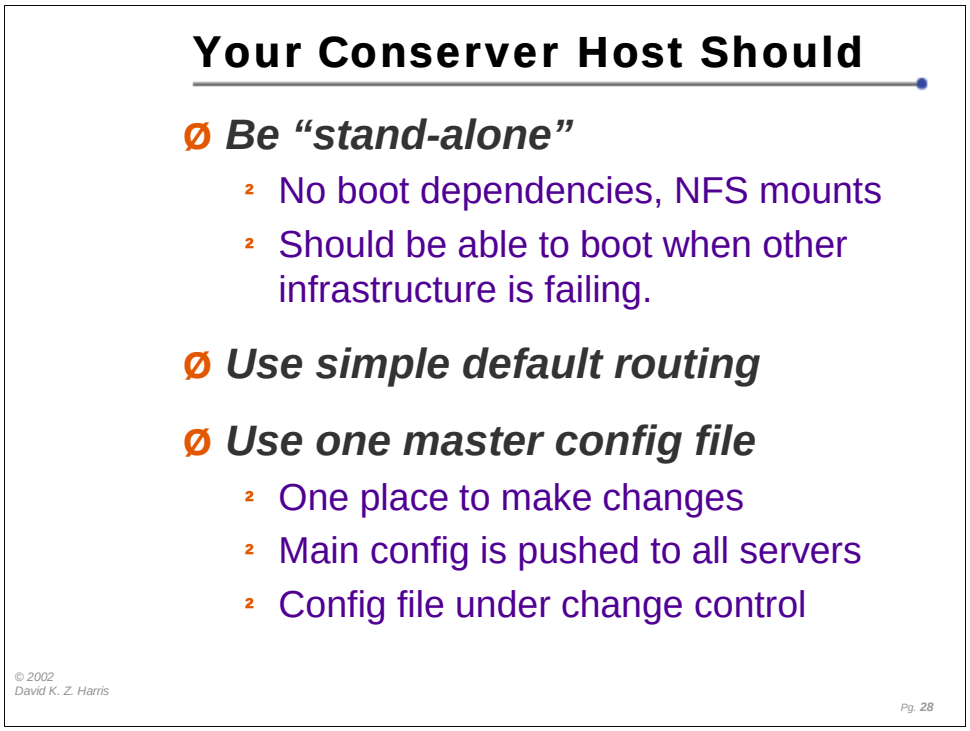


The CHANGES file is included as part of every distribution, but it is also available for browsing from the Conserver.com main web page. This lists the change history since version 6.05.

After the LISA-2000 talks and BoF, there was a significant increase in enhancement requests, and in new patch submissions. This flurry resulted in a rash of bug fixes and new revisions, and Bryan opted to adopt a new version numbering scheme , removing the "GNAC" prefix after version 6.17.

I believe the most-requested feature was a way to re-read the conserver.cf file without quitting conserver (and breaking any active client connections). This can now be done with the SIGHUP feature (`kill -HUP [main-conserver-psid])`.

A recent change is the addition of SSL between the clients and conserver application servers. This is still being fine tuned, having been added in the 7.2.4 version.

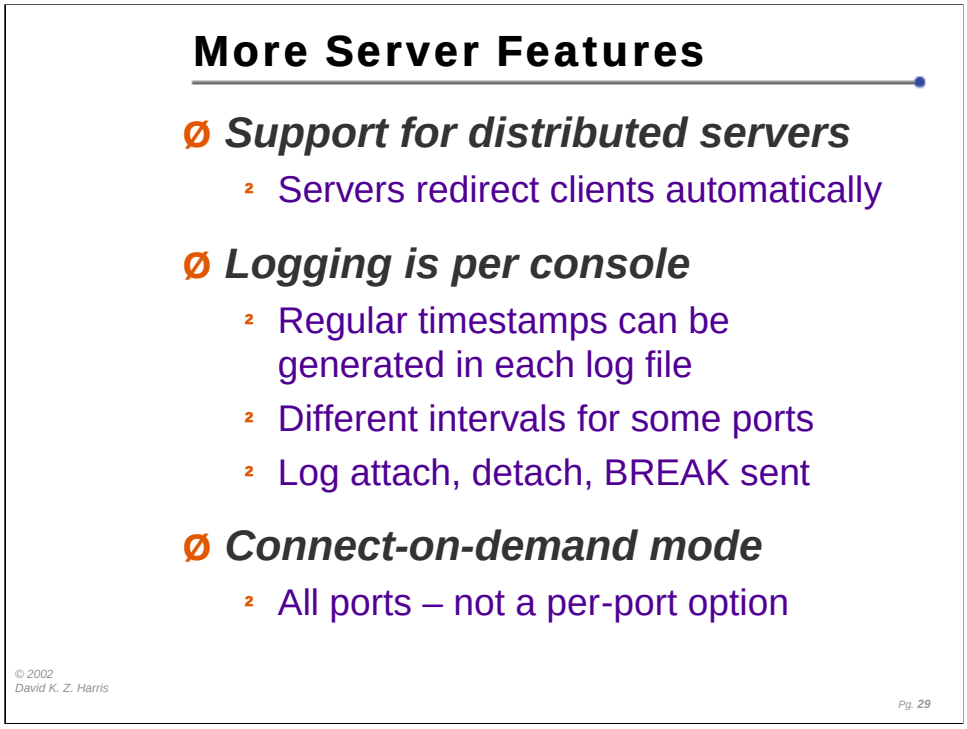


In a complete power failure, the console server should be one of the first hosts turned back on (just after you turn all of the terminal servers, and basic network gear). You want the console server to boot quickly, and to try to establish the reverse telnet sessions to the terminal servers before you start bringing up your main infrastructure hosts, and peripheral network gear.

The server should not depend on any other machines in order to log. In the event that some of your network or infrastructure servers fail, you want the console server to be capturing log data for as long as possible during a failure.

If you are planning to use distributed console servers (putting some in remote offices, or distributed data centers), you should create one central console server file, which would include data for which servers will log for which ports. It’s a great idea to keep the master file(s) under a change control system.

By keeping one file, under change control, you will have a single place to make changes, and only one place to check to see if a device is listed. Adding a new server later becomes easy, since you can edit one file to make the additions, and then push that file to all of the console servers, including the new one.



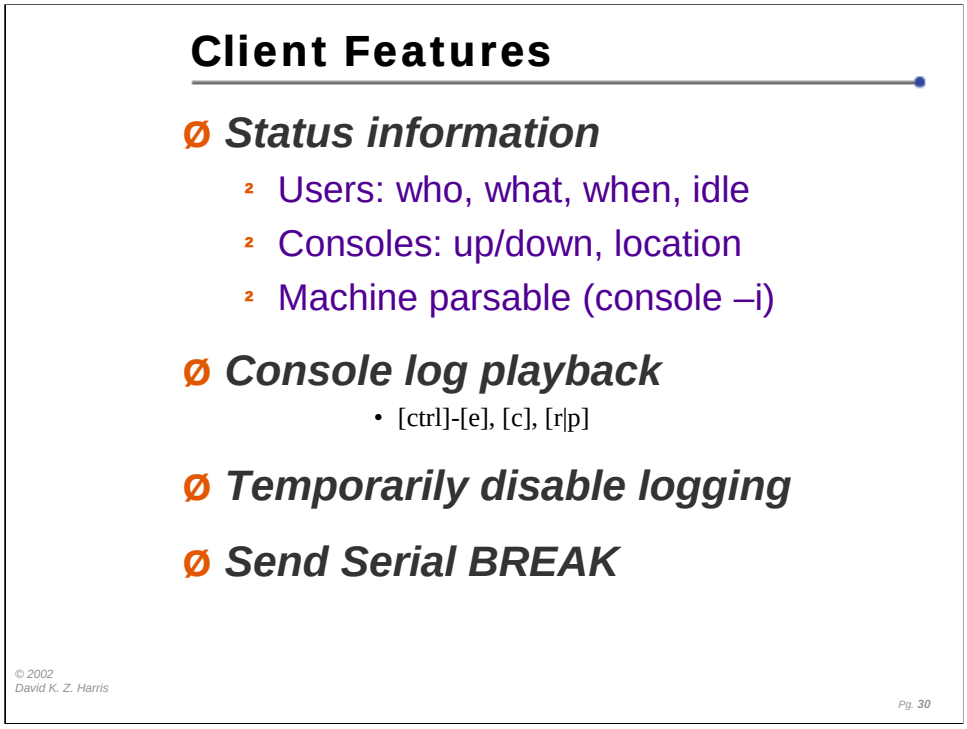
Multiple distributed console servers are supported. If there are no security zones restricting access, each server should use the same conserver.cf file. The conserver software will automatically redirect the clients to the appropriate console server.

Logging and regular timestamps ("marks") can be enabled on a per-console basis, through the entries in the conserver.cf file.

You may not want to log certain ports, and that's just fine. You can determine which files are logged by the entries for each host in the conserver.cf file.

Regular timestamps are useful for providing time bounds on console output for consoles that don't automatically timestamp their messages. You can set any port to provide a date/timestamp into the log file for that port by including the desired interval for each device that you want to have conserver timestamps.

Connect-on-demand mode will open a connection to a serial port when a user makes a connection to the console server. Connections are closed when all users have disconnected from that console. All logging outside of when connections are established are lost in this case. (This is a 'global option'. That is, if you enable this feature by invoking conserver with the "-I" flag, all of the consoles will work this way…if nobody is watching, then no logging occurs either.)



These are really implemented within the server, but they are seen as the user experience.

Various status information regarding users and consoles can be retrieved. Things such as who's connected to a particular console, what mode (read/write or read/only) the user is in, when they connected, and how long they've been idle are available. Console status, such as whether the port is connected or not and where the port is located (terminal server, local port, or command) can be retrieved using the client as well. (Issuing the command `console –i will return 'machine-parsable' versions of this output, for scripting.)`

Temporarily disabling log files is a nice feature for when you know you'll be displaying unencrypted passwords or other sensitive information. When the active read/write user disconnects, logging is automatically re-enabled.

Issuing the command `console -w` will display which users are connected to which consoles (across all servers), and which machine they are connecting from. (Use console –W to show connection details for the Master only.)

Issuing the command `console -V` will display the version of Conserver Client running, the initial master server (for that client), and the default escape sequence for that client.

Issuing the command `console –r will show you the version of Conserver Server running.`

More Client Features

- ***One-write, many-read***
  - Force read/write mode
  - Control goes into a stack
  - When someone disconnects, control reverts to the person who had it before.
- ***Broadcast messages***
  - Similar to ‘wall’ command

Connecting and disconnecting from serial ports helps reset things to a known state. Sometimes you can run into a software flow-control problem or general port lockup and dropping/closing the connection and re-establishing it can usually clear things.

The server enforces a policy of at most one client being in read/write mode for each console at any one time. Other clients attached to the console are in read-only (or “spy”) mode. A user can “bump” others into “spy” mode and promote themselves to read-write at any time. Being able to forcibly acquire read/write mode is useful in emergencies and for mentoring purposes. If a less experienced admin is about to do something dangerous or an admin has left his keyboard but is still connected to the console, you can “steal” the connection and work on the issue.

Broadcast messages can be sent via the client “`console -b`” option. It’s similar to a “wall” (write to all) message, but it is directed only to clients currently connected to a console. The message is sent to users connected to all conservers, in a distributed conserver implementation.

Where To Get Conserver

- *Download the tar file*
  - http://www.conserver.com/
  - ftp://ftp.conserver.com/conserver/
- *Distributed freely*
- *Support options*
  - Email lists
  - Documentation
  - Console Guide Pages

The primary distribution site for conserver is http://www.conserver.com/.

Currently, there is no commercial support offering for this application. We're trying to improve the documentation, and we are always interested in hearing comments from users, including suggestions for improvements or new features.

You can find additional supporting information about connecting devices to terminal servers at http://www.conserver.com/consoles/.

We have established email lists for new-version announcements, as well as a BOF-like discussion group. Subscription information can be found at the conserver.com web site (http://www.conserver.com/).

Conserver Distribution

CHANGES	INSTALL
LICENSE	README
FAQ	TODO
autologin/	conserver/
conserver.cf/	console/
contrib/	configure
install-sh	Makefile.in

And a few other files used during the configurations process…

When you untar the tarfile, you will find a directory named the same name as the tarfile (without the .tar extension). The directory will include a bunch of files, including the then-current Conserver.com main web page (conserver.html), and Bryan Stansell's TODO list, so folks can read what is still slated for future releases.

The README file at the top level will guide you through the distribution. Most things should be obvious. Here are some of the things that may not be.

The `autologin` directory is full of code that I haven't even tried to compile or use. Check out the `README` files if you're interested, but this code is designed to provide a shell on a host instead of a login prompt – so you don't have to log into your consoles.

The `contrib` directory will contain various contributions from the community. Currently it has instructions and support files for building a Solaris package, as well as clues for setting up an RPM package for RedHat Linux..

The `conserver` directory contains the source code for the conserver application.

The `console` directory contains the source code for the console application.

The `conserver.cf` directory contains sample conserver.cf and conserver.passwd files as well as the associated man pages.

The Important Documents

- *The INSTALL file*
- *The CHANGES file*
- *The Man Pages*
  - console(1)
  - conserver.cf(5)
  - conserver.passwd(5)
  - conserver(8)

The INSTALL file is your best (current) information to install the version you have. There are the ‘quick, default’ installation instructions, as well as some more detailed information in this file.

The CHANGES file is a version history (what was added and/or fixed) in the various versions.

The man pages provide the most complete information regarding the various flags available in the version you have.

Easy Default Installation

- *Uses auto-configuration*
  - Detects/reports system settings
- *Custom configuration also easy*
  - Flags replace older cons.h file
- *The install is verbose*
  - 'console -V' also prints settings
- *You need config files*
  - conserver.cf and conserver.passwd
  - Copy and modify samples, or create your own versions.

Read the INSTALL file for details, since it always includes 'current' info for the distribution version that you have.

The 'default' installation is;

unpack the tar file

'cd' to the distribution directory

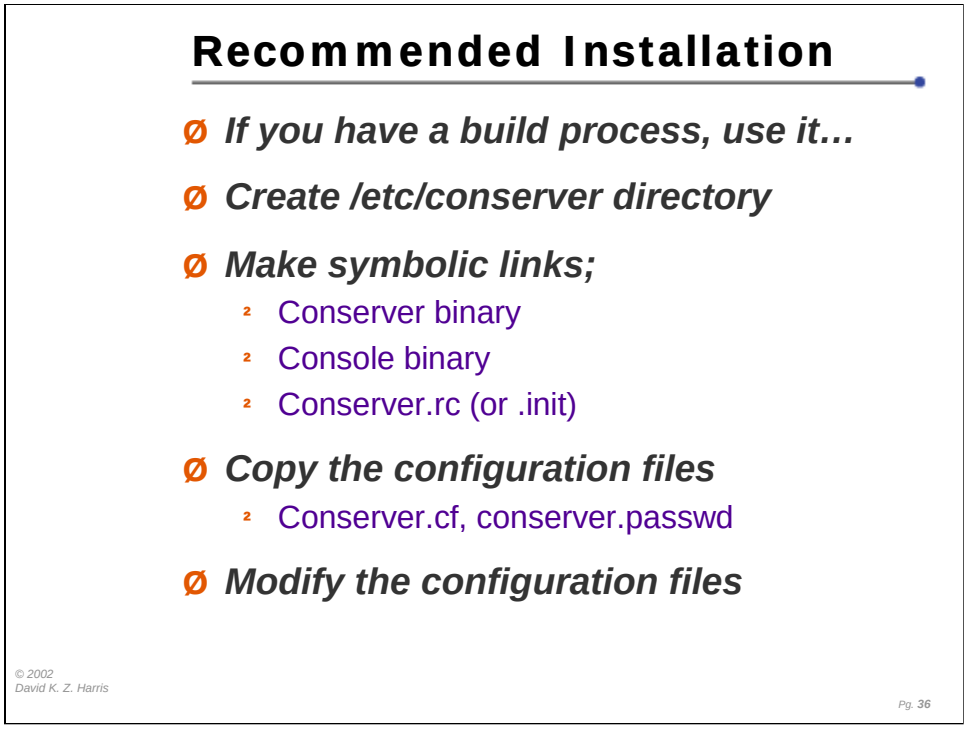
run './configure' for the auto-configuration

fix any reported problems

run './make'

fix any reported problems

run './make install'



If you already have a process for installing packages, by all means USE IT. This page is a suggestion for sites without such a formal infrastructure in place.

The default installation will put the conserver and console binaries in different places. You can either use flags with the 'configure' command, or you can leave them in the default locations and use symbolic links to make them appear in a single directory. In the places where I've used conserver, it was common to find all of the current commands and files in /etc/conserver. In some cases, these were sym-links to the directories where the current versions were stored. (Using 'conserver -V', you can view the installed configuration.)

Using a consistent directory for your binary files and configuration files will make it easier for users. As you upgrade versions, you can change the sym-links, and the users (and scripts) don't need to do things differently.

If you already use a change control system, consider using it to track the changes to the configuration files. Even if it's something as simple as RCS, using a change control application will pay benefits later on, by preventing accidental changes, and giving you the chance to fall back to an earlier version if some changes cause you problems.

Consider who needs to have access to the configuration files for editing. Consider restricting even read-access to 'the world', since you may add accounts and passwords (crypted) in the .passwd file.

conserver.cf console entries

- ***The console.cf [name] entries…***
  - Conserver clients connect to the [console name]
  - Log files can inherit the [name] with “&”
- ***Console entries allow for:***
  - Remote Access (terminal servers)
  - Physical Access (serial ports on the server)
  - Communications with shell programs
- ***Optional log time stamping***
  - Log attaches/detaches, and sent BREAKS.
  - Timestamp interval can vary on per-port basis
  - Great for use with devices that don’t timestamp their log messages!

Read the conserver.cf man page for more detailed information.

The first half (before the “%%” line) of a conserver.cf file contains console entries and, optionally, special keyword assignments.

Console entries are made up of 6 fields.

The console name

the location of the serial port

a secondary location value

a logfile name

an optional mark time interval

and an optional BREAK sequence

If the LOGDIR relative path has been defined at the top of the file, and the [logfile name] contains the ‘&’ character, the logfile will be created in the LOGDIR with the [console name].

Mark intervals, if used, are a numeric value followed by ‘m’, ‘h’, or ‘d’, for minute, hour, and day (4h = 4 hours).

For terminal server-connected ports, the syntax looks like this:

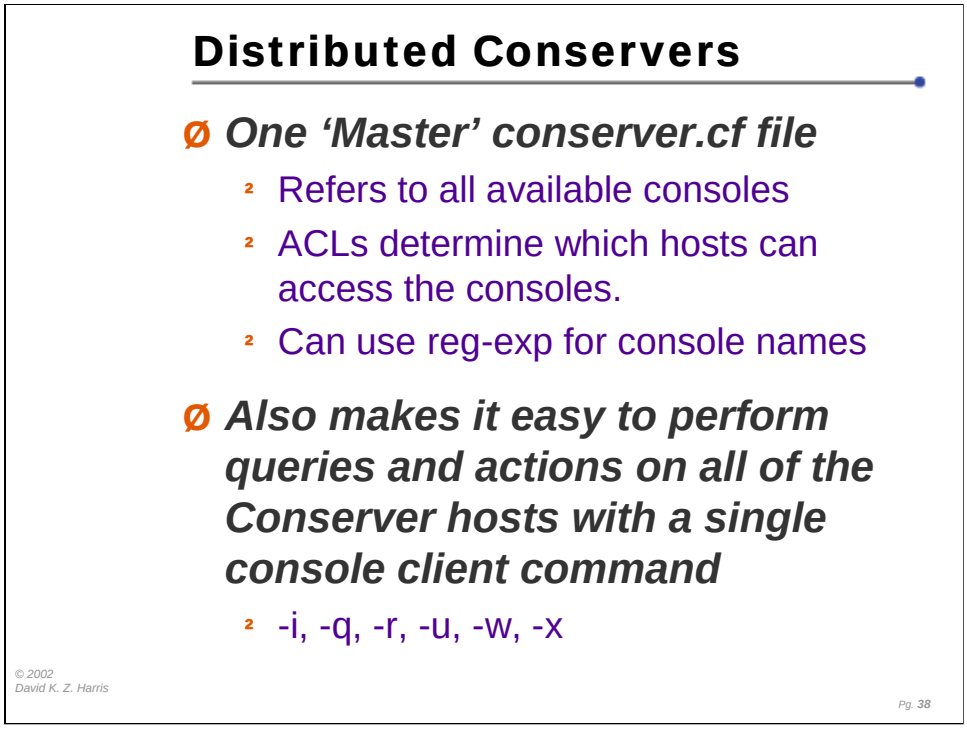
name : !host[@host] : port : logfile : mark-interval[m|h|d]:

For connections on a built-in serial port, the syntax looks like this:

name : tty[@host] : baud[parity] : logfile : mark-interval[m|h|d]:

For invoking a shell program, the entry looks like this:

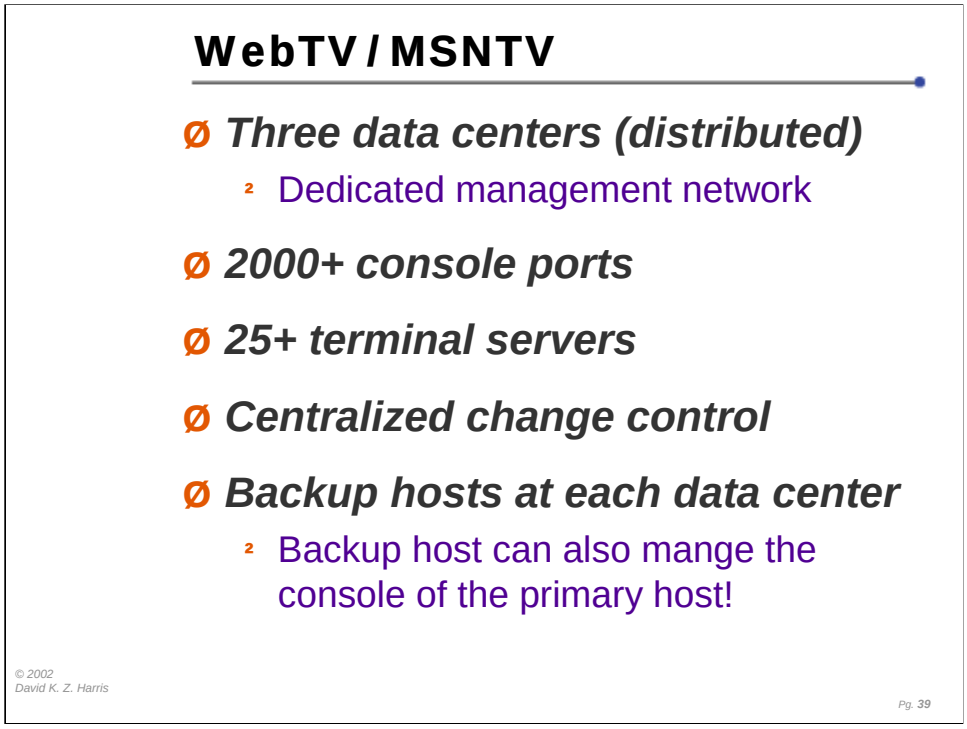
name : |command : : logfile : mark-interval[m|h|d]:



Running a "Distributed" Conserver means using a single conserver.cf file that includes all of the consoles that you care about, no matter which of the Conserver hosts is responsible for logging each port, and then you distribute that single file to all of the hosts.

One benefit of this method is that it's easier to manage one file, instead of many. (Otherwise, you have the problem of needing to grep many files to find the one file with the information for the single console port you wanted to check or modify.

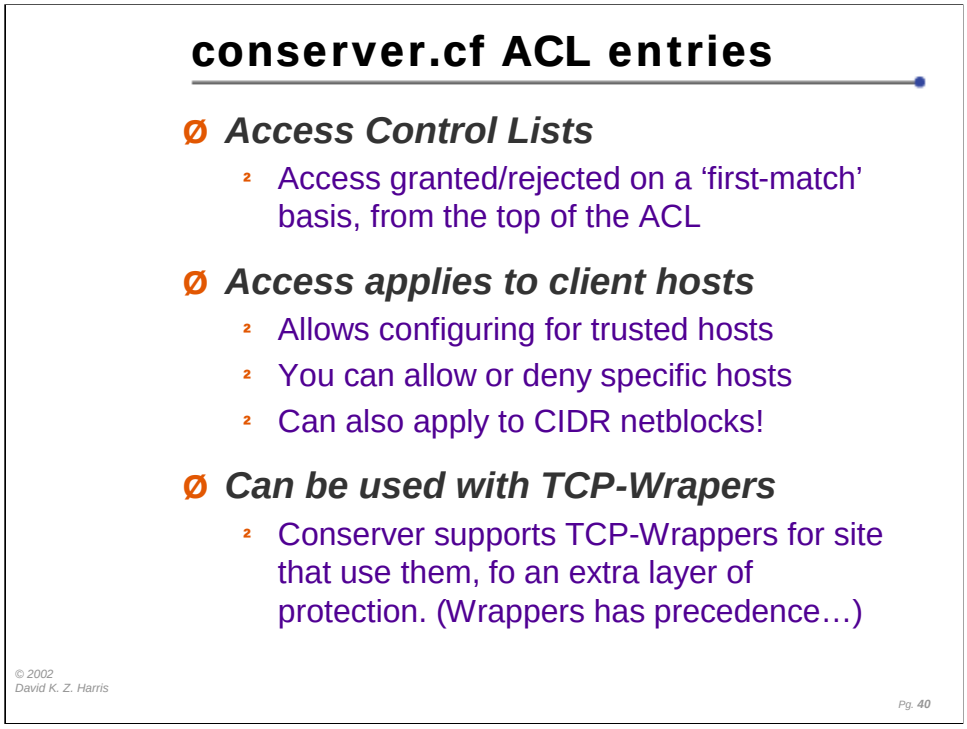
Another benefit is that the single file can tell you which conserver application hosts holds the logfile for any given monitored console.



Conserver followed a few senior Network Admins from Synopsys to WebTV. Microsoft later acquired WebTV, and later still Microsoft morphed the service into MSN TV.

During the growth of this ISP service, they added more data centers, and the number of hosts and conserver users grew significantly. The Network Operations folks changed out most of the Console Server hardware to Cisco 3600 series, except for a few nodes where serial BREAK would not be a problem. All of these console servers require SSH to access and manage the console servers.

Conserver is tightly integrated to monitoring tools, and is a core tool of the NOC operators. As a result, the change control processes for their distributed mode deployment also copies the files onto a second host in each data center. If the conserver host in a data center fails, it is an easy process to bring up conserver on the secondary host, and then continue capturing logs from other hosts while the primary host is rebuilt.The secondary server can also allow admins to access the console on the primary server.



The second half of a `conserver.cf` file (before the "%%" line) contains access control list (ACL) entries. The basic syntax of the command is;

```
{trusted|allowed|rejected} : {host|ip|cidr}
```

The ACL entries are two fields; the keyword "trusted", "allowed", or "rejected" separated by a colon from the hostname, ip address, or CIDR address block.

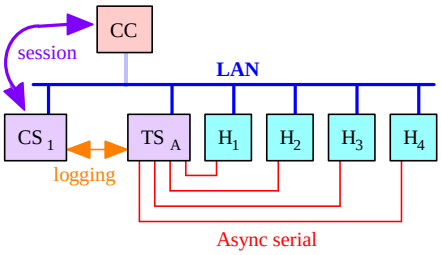
Access will be granted (or rejected) on a first-match basis, so the order of the entries is important.

> The `trusted` tag means that no authentication is needed if console client is from that host, address, or network. You should ensure that these trusted hosts require strong authentication before users can access the console client on these hosts. After all, the client allows you to specify a username, in case you are using the client under someone else's shell…so, someone could specify someone else's username, with no need for authentication, on a trusted host.
>
> The `allowed` tag means that connections are allowed, but still require username and password authentication.
>
> The `rejected` tag means that no connections are allowed from that host, address, or network.

For the second half of each entry, {host|ip|cidr}, you can actually have a list of host names (or host addresses, or CIDR blocks) on the same line, separated by spaces, if you prefer. You can decide which way is easier to read and understand.

> Example: `allowed : 192.168.9.0/24 192.168.42.0/23`

Single Conserver Example



ø conserver.cf file

```
LOGDIR=/var/console
h1:!ts1:2001:&:1h
h2:!ts1:2002:&:1h
%%
allowed: cc1
trusted: cs1
```


The configuration file is very small, but it's specifying a lot of information.

First of all, the default location for log files is being set to /var/console.

Next, two consoles are being defined: h1 and h2. Each of the consoles are hosted on the terminal server named ts1. h1 is associated with TCP port 2001 and h2 is associated with TCP port 2002. Each console log file will be time-stamped hourly.

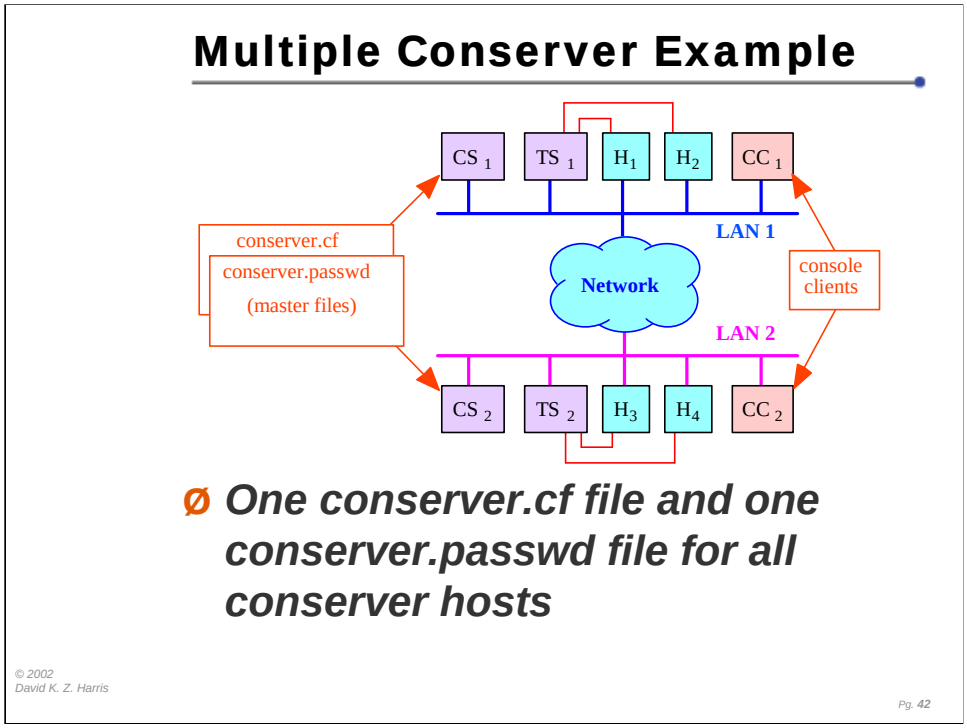
The log file itself is specified as "&". This special character is replaced with the console name ('h1' or 'h2' respectively). Since that is a relative path, the `LOGDIR` prefix is applied and you end up with "/var/console/h1" as the full log file name.

Then we see the "%%" line. This separates the console names from the ACLs.

The Access Control List is stating that client connections from cc1 are allowed (must succeed with password authentication) and connections from cs1 are trusted (no authentication is necessary).

Additionally, you can use the "#" (pound sign) to add comments to your configuration files, to help keep things clear. (Consider adding a line before each terminal servers entries, giving the hostname, ip address, and physical location of the device. You can also consider noting what type of hardware it is.)

```
# ts-1 (c3620)
# Data center 1, bldg. B, top of rack 3
```



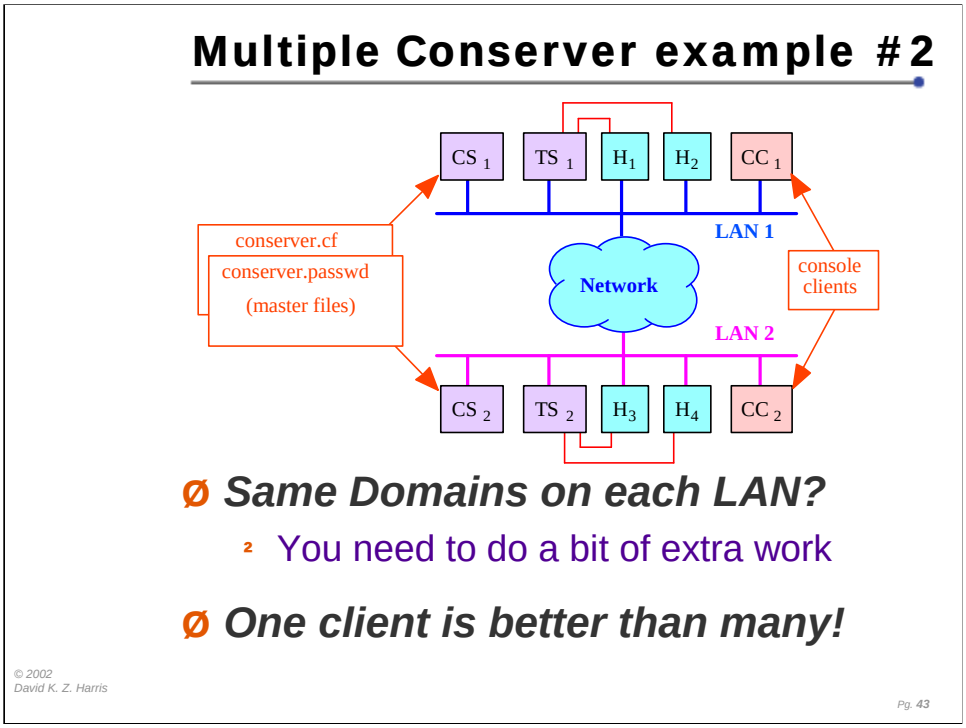
This example is a basic representation of a multi-conserver setup. Here is a sample configuration file for the diagram:

```
LOGDIR=/var/console

h1:!ts1@cs1:2001:&:1h
h2:!ts1@cs1:2002:&:1h

h3:!ts2@cs2:2001:&:1h

h4:!ts2@cs2:2002:&:1h

%%
allowed: cc1 cs1

allowed: cc2 cs2
```

Notice the "`@cs*`" tags used in the second field. This specifies which console server host should manage which console. Any time you have multiple console servers, every line in the configuration file should contain a "`@cs`" tag.

You should create one `conserver.cf` file for use with all of your distributed conserver hosts, putting the `@` tag in every host entry.

The result of this file is that cs1 will make two TCP connections to ts1, and cs2 will make two TCP connections to ts2. If either of them is contacted by a client and asked to be connected to a console they aren't managing, they will redirect the client to the appropriate console server. If you did not distribute one master configuration file using this syntax, the various servers would not know about the others and the redirection would not occur.

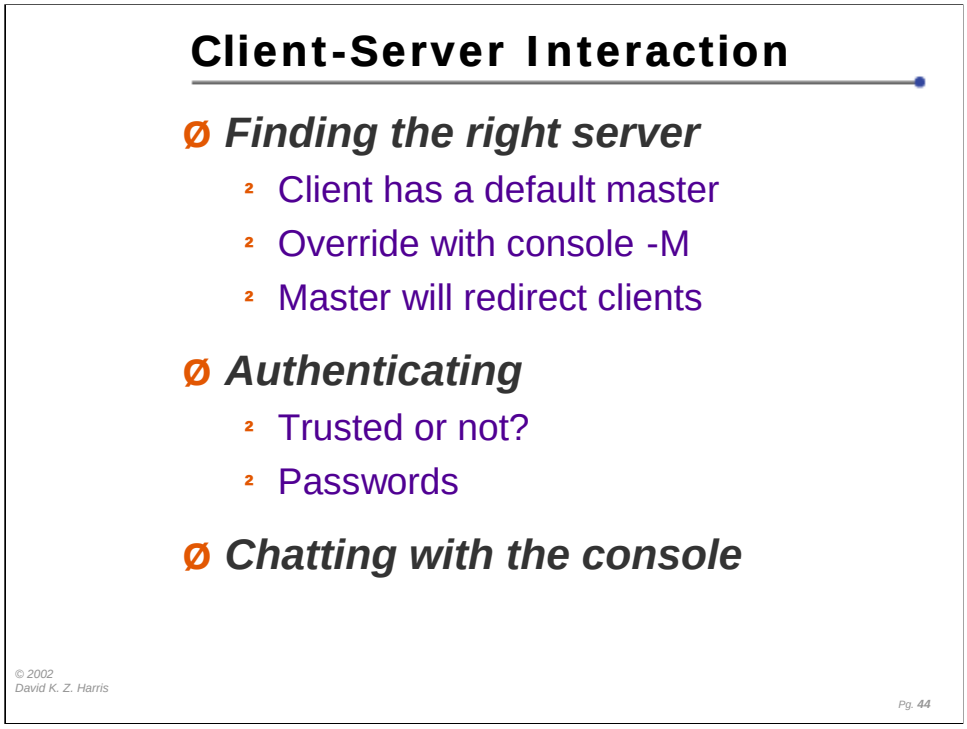


If you only ever run "console" on hosts that are running conserver, compile in "localhost" as the default master hostname. You do this by adding an option to your "configure" command:

```
./configure --with-master=localhost
```

If you want to be able to run "console" on non-conserver hosts, you can leave the default master hostname as "console" (no special action required at compile-time), and then, on each conserver host, make "console" an alias for localhost in /etc/hosts (assuming your name service switch configuration specifies /etc/hosts lookups before DNS or other services).

If in your flat namespace you need to be able to run "console" from any host on any LAN, and always connect first to the conserver host on the same LAN, then it might not be unreasonable to have separate binaries on each LAN, especially if those LANs are administered by different groups who maintain their own collections of tools anyway. But that's the exception to the rule, done only as a last resort. Even in this case, it would be more elegant -- and simpler in the long run -- to install wrapper scripts on each LAN to invoke the real console binary (which would be installed under a different name, like "console.real") with the appropriate -M option for that LAN; that way you wouldn't need to build separate binaries every time you upgraded the conserver package.



The first step a client takes when connecting to a console is to find the right server. The client is compiled with a default server name and it will default to that server, unless you override it with the –M flag.

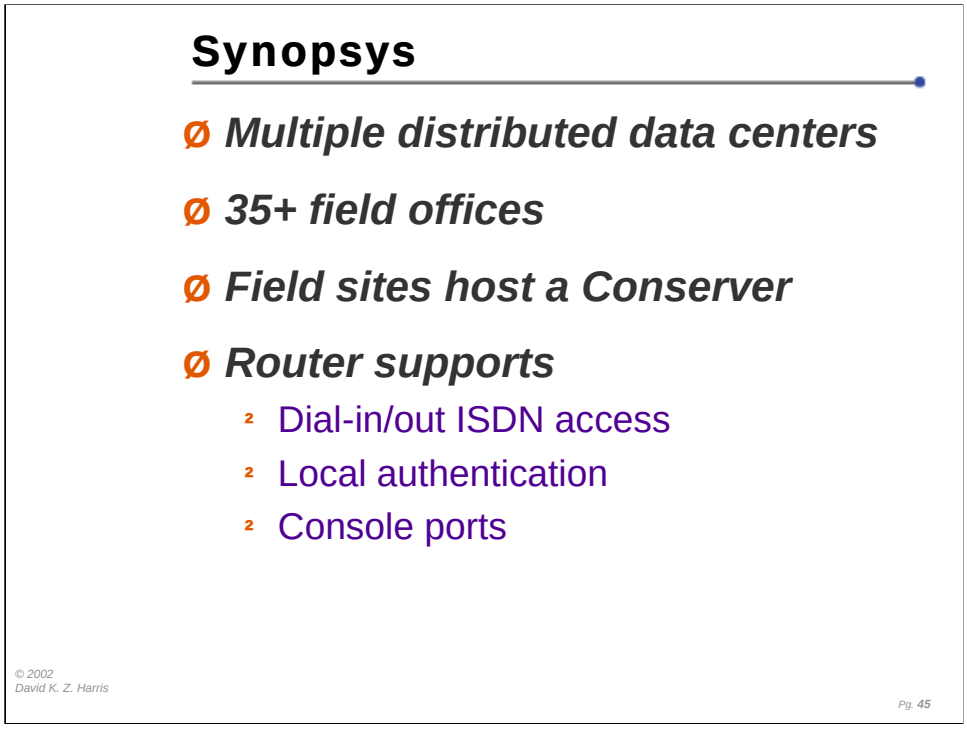
Example: `console -M cs3`

This can be useful if you're testing a new conserver host, or moving your conserver application to another machine during an emergency and don't have time to rebuild the client, or your default console server is down in a distributed setup and you need access to a console managed by another server.

If you have a distributed console server setup, you can connect to any server and it will redirect the client to the server managing the console requested.

Once the proper server is found, access is either granted immediately (to "trusted" hosts), granted after authentication (for "allowed" hosts), or immediately rejected (for "rejected" hosts), based on the ACL entries in the `conserver.cf` file.

If access is granted (immediately or after a password check), the connection is made to the requested console. Anything typed on the console will be delivered to the attached console, and the client will begin seeing the console output. (The traffic coming from the attached console is logged, as well as being passed to any connected conserver clients.



Synopsys has been using Conserver for more than a decade. They had expanded their implementation across the major data centers for the company, and during 2000, they replaced older ANNEX terminal servers for all Cisco 3600-series hardware, to eliminate Serial BREAK problems.

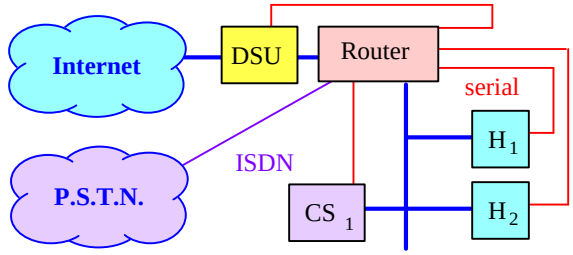
After attending the LISA 2000 tutorial on Conserver, Jeff Komori architected a transition to distributed mode multiple Conserver hosts, and they deployed it across all of their medium and large field offices.

The model for a field office was a Cisco 3640 chassis, with Ethernet, WAN WICs (Serial and ISDN links), MICA modems, and NM-32A serial modules. The High-speed serial WAN link and Ethernet continued to play their traditional WAN/LAN router role. The ISDN and MICA modems provided high-speed modem dial-up access into the field office, reducing the corporate dialup costs. And the async serial card provided console access.

The big win was during a WAN link outage, where a Network Admin could see both sides of the failure, and where console logs turned up configuration needs on field office hosts when the WAN link was down.

Synopsys Basic Field Office

- *WAN for main traffic*
- *PSTN (ISDN) for field dialup*
  - (Public Switched Telephone Network)
- *Local Conserver Host*



In the field office, the field office router also became the Console Server, and the ‘local dialup connection’ for staff near the field offices.

By adding a local Conserver host in each field office, the system administrators now had a tool to understand what happened to the servers in the field offices when the WAN link failed. This helped uncover system dependencies (such as libraries mounted over the WAN, or DNS dependencies) that would hamper ‘local’ work if the WAN was down. Knowing about a problem is the first step in fixing it.

The dialup access saved significant ISP costs each month. By using an ISDN line into each office, the dialup connections could support speeds greater than 28.8 and 33.6 Kbps. However, this also provided the network administrators a method to ‘be on both sides of a broken WAN’. With the primary link down, they could now perform diagnostics from the field office, as well as from their NOC. This required a local authentication host at the field office, which could be managed from the main office while the WAN was working normally.

Starting Conserver

- *Conserver flags*
  - -C : Configuration file override
  - -d : Become a daemon
  - -D : Enable debugging output
  - -i : Initialize consoles on-demand
  - -M : listen only on interface with this specific IP address
  - -u : send unloved to stdout
  - -v : Verbose output
  - -V : Version information

Conserver has many flags for modifying it's default behavior. Some of these flags are crucial, others are not. For normal operations you want to use the –d flag.

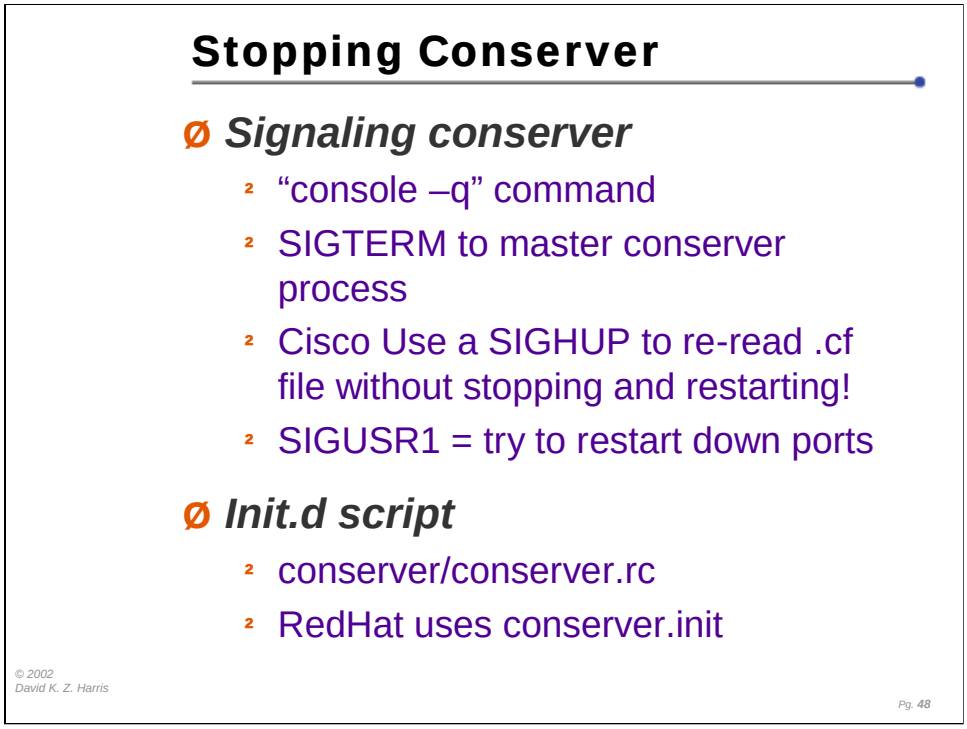
The –u flag sends output from 'unloved' consoles to stdout. This is meant for on-duty staff to monitor for output thaat isn't being dealt with by someone else.

The –D and –v options are useful if you want to troubleshoot startup problems or other aspects of the server.

The –C flag allows you to override the location of the conserver.cf file. The –i flag enables the "initialize-on-demand" mode, which makes conserver connect to console ports only when a client connects, and disconnect from the console ports when all users have disconnected.

The –V flag provides version information as well as compile-time settings.

The conserver/conserver.rc startup script is a working startup/shutdown script written for a Solaris environment. Use it as a guide for generating your own script or /etc/rc.local entry.

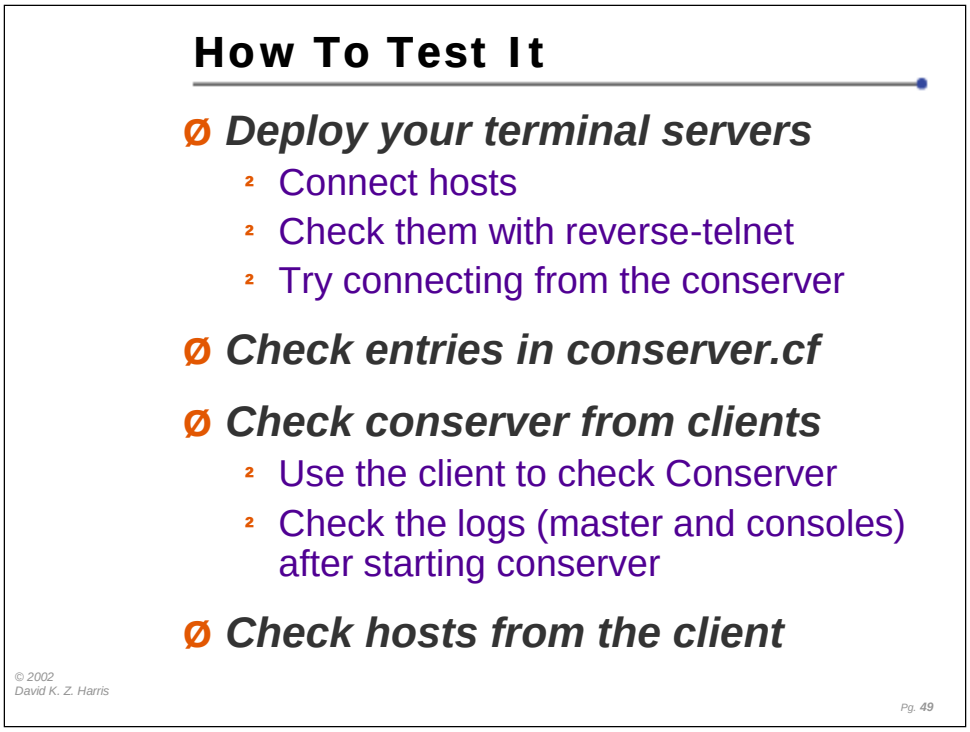


There are two ways to gracefully signal conserver to shut itself down. The first method is using the console client with the –q flag. If you're asked for a password when you use this method, you must provide the root password of the console server host. If no authentication is done (you're coming from a "trusted" host), conserver will shut down without a password.

The second method is to send a SIGTERM to the master conserver process. When it receives this signal, it signals all the child processes to exit and then exits itself.

The distribution comes with an init.d script in conserver/conserver.rc. This script (modeled for a Solaris environment) uses the SIGTERM method of shutting down the software.

You can find a model for the conserver.init script in the conserver contrib/redhat/ directrory.



Terminal server setup was covered briefly in the Basic Serial Tutorial.

After basic serial setup, try connecting to the consoles you have attached to terminal servers by using the reverse-telnet capability of the terminal server. You should test consoles connected to local serial ports via “tip” for built-in serial ports, or by telnet to particular addresses and TCP ports for terminal server and console server hardware attached ports.

* If you cannot connect to the ports manually, Conserver won’t be able to connect to them automatically.

** Testing manually will find out if other applications (getty, etc.) is already attaching to certain port, and will turn up DNS and routing problems on hosts and console server hardware.

Once you’ve verified connectivity, populate the conserver.cf file with the appropriate entries.

For consoles on devices that have DNS or NIS hostnames, we recommend that you use the same hostname for the name entries in `conserver.cf`. This usually makes it easier to remember the name to use to reach those consoles. .For devices without DNS names, you can name them for the DNS devices they attach to (the DSU for a router, or the drive array for a host, etc.)

When running conserver, verify that connections are made to the various ports and/or terminal servers, and that no errors are being produced during startup. Check the main log (by default, /var/log/consrever) for errors. Typing ‘`console -V`’ on the conserver host will produce compile-time options and information that conserver is looking for.

Client Commands

- *The default escape sequence is:*
- *[CTRL]-[e] then [c] then:*
  - ? : brings up help menu
  - r : replays last 20 lines of the log
  - . : disconnects from the session
  - f : forcibly take over a session
- *Other commands are available*

All interaction is done from the client. Most interaction is done when connected to a console.

When you are connected, the default escape sequence is “CTRL-e” then “c”.

After the escape sequence, you use one or more characters that implement various functions.

A `"?"` will bring display a list of all available functions. The most popular are `"."` to disconnect, `"z"` to suspend, “`f`” to bump someone into “spy” mode and force you to be read-write, “`r`” to replay the last 20 lines, and “`l1`” to produce a BREAK signal.

You can use “w” to see who else is also attached to the console that you are on (handy to see who is spying on you ;-).

You can even send any character you like to the host using a 3-digit octal code (useful for sending control characters and other strange bytes).

Depending on what mode you’re in, you’ll see different lists of functions. When you are in Spy mode (read-only), you have no write capabilities, so some of the normal functions will be missing.

Updating Conserver Files

- *When do I need to restart it?*
- *Modify conserver.cf?*
  - Send SIGHUP
- *Modify conserver.passwd?*
  - Do nothing
- *Roll console log files?*
  - Send SIGHUP
  - Check the master conserver log

Restarting the conserver host means a short bit of downtime. The time will vary, but a rule of thumb is the more console names you have, the longer a restart will take, as the processes are spun up, and each reverse-telnet session is opened, and the associated logging is restarted. (But, it will take significantly longer if you are trying to reach a Console Server device that is not reachable on the network!) The solution in version 7.2.0 and later is to send SIGHUP (kill –HUP `cat /var/run/conserver.pid`).

Any changes to the conserver.passwd file are used immediately. The file is opened and read every time a conserver client authenticates with conserver. If you have a large conserver.passwd file, this could result in a detectable delay when authenticating.

Sending the conserver a SIGHUP will cause it to re-read the conserver.cf file, without touching any active client sessions. This is useful if you roll your console logs or have some need for a “semi-restart”, or you have added or removed console ports in the conserver.cf file.

Time Synchronization

- ***Important for logging***
  - backup and file sharing too
- ***Comparing logs from many devices after an 'event'?***
  - Security devices
  - Hosts, servers
  - Network (routers, switches, load balancers)
  - Check non-network devices often

In a distributed file system, time synchronization is an important issue. It is also important if you are going to be comparing logs between distributed machines.

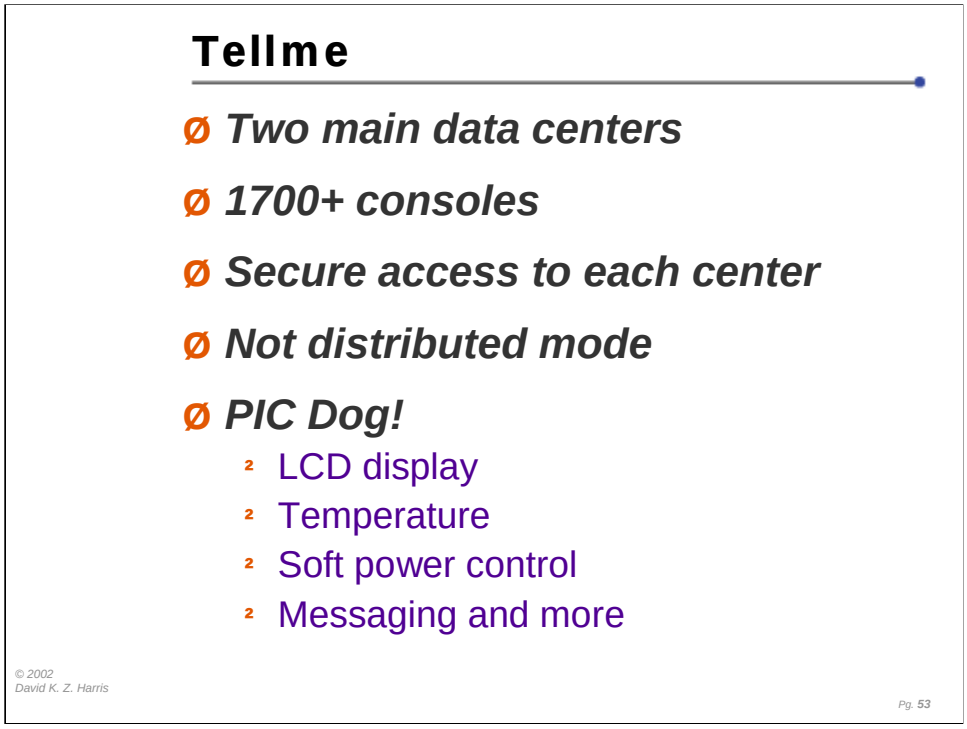
NTP is a practical solution for synchronizing the time between various machines and the logging server.

Network devices and servers should be configured to include date/time information in the messages sent to the serial console. If you are logging devices which cannot include timestamps in the messages, you can ask conserver to include periodic timestamps from the server in specific log files, which can provide a basic reference timestamp. (But the conserver host will need to be set to the same time as the hosts that can properly timestamp.

Consider the issues of comparing logs from machines across multiple time zones. If the machines keep their time in the 'local' time zone, the administrators will have to mentally compensate for the time differences when searching through the log files.

However, if you decide to sync all devices to a single time zone (GMT perhaps?), you then have the problem that admins still need to "do the math" and convert to their local time zone for resolving single-machine issues.

The bottom line here: Someone will have to convert timestamps some of the time. You should decide when it would be better; during a crisis, or day-to-day.



Tellme has been using Conserver for more than two years.

Since naming between the two data centers was very similar, they opted to not use distributed mode, in an effort to reduce the chances for typo mistakes, such as rebooting the wrong host.

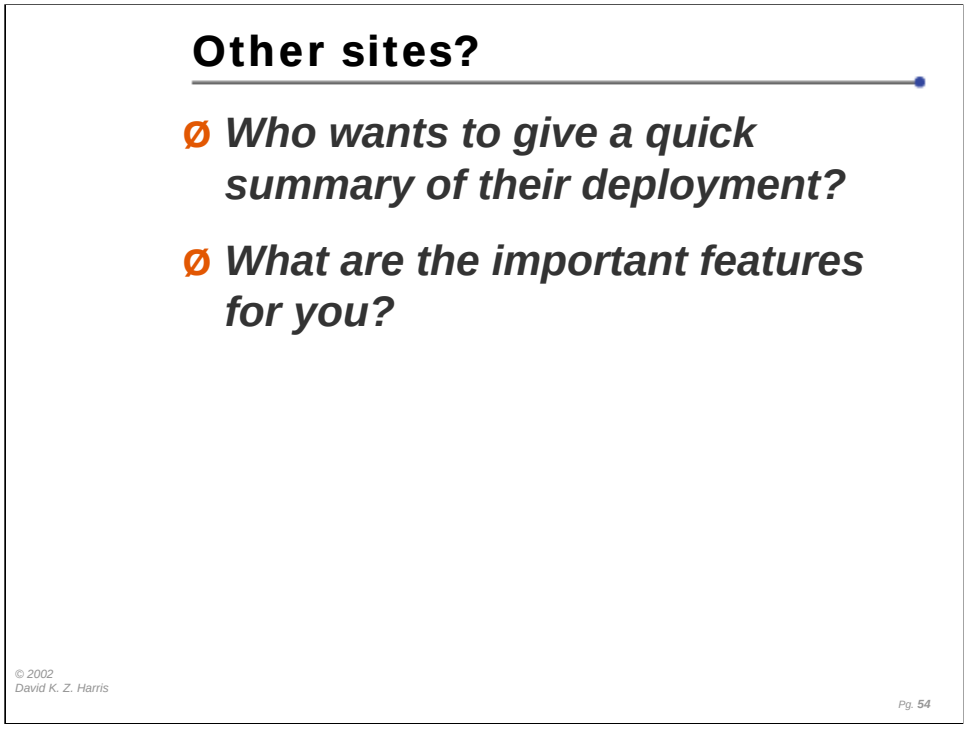
Only the Network Operations folks can make changes on the Conserver hosts (including adding access and privileges for users, and modifying the hosts/console lists).

Cary Roberts has developed a special interface board for their servers, called “PICDog”, and this option is being resold now by Rackable Systems in their 2-U tall “1000 series” rack-mount PC chassis (the option is called “Phantom”). It uses phantom power from the server, has an LCD display (2 lines, 20 characters), and connects as a pass-through device on one of the COM ports, providing a variety of features, similar to the control you can achieve with the PC Weasel cards. It adds temperature, and other signaling as well. The PC BIOS also supports console redirection.

http://www.rackable.com/

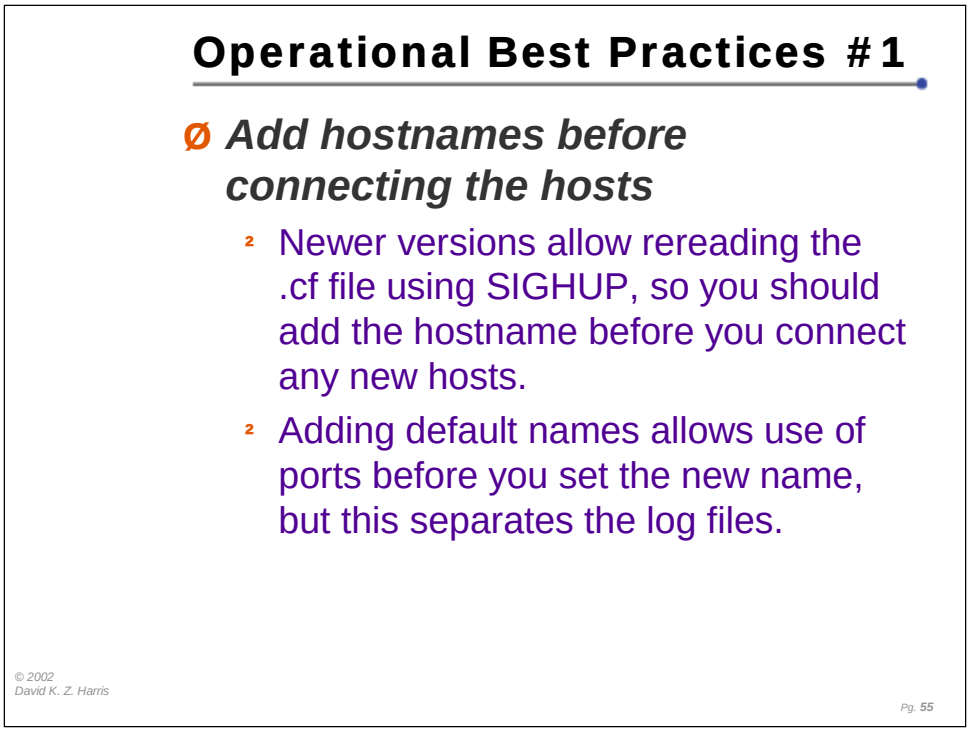
http://www.rackable.com/lightsout.html

http://www.rackable.com/advantages.html



The Conserver Users list has been a good place to read about some other deployments, as well as reading about some users problems and getting the answers to their problems.

I also get a lot of email from folks who have found my console pages, and they thank me, but they "just have one more question"…and in our exchanges, many folks will tell me something about their deployments. Bryan also gets the domain name statistics from folks downloading the Conserver code, and it's been interesting to see who has pulled the distributions down.



I suggest adding default names for unused/unassigned ports. ([ts-name]-[line-number] was a good naming convention.) This allows you to use a port before you send Conserver a SIGHUP, since the port is already named something intuitive.

However, using this method, the initial set-up of devices would end up under the log for “ts-16-39”, and then under the new host name after the SIGHUP.

Whether you set up default names, so that you can use the ports easily before you configure the new name for the host

Operational Best Practices #2

- ***Mentoring***
  - One person can control a session (read-write), but many can watch.
  - Juniors can watch seniors in action.
  - Seniors can watch juniors in action!
  - Used with a conference call (for example, during a downtime), this provides a way for someone in a remote site to 'watch over the shoulder' of someone else making the changes.

This is an important feature. The simple concept is that only one person can control a port (that is, have read-write access) at one time, but remember these two other things;

1) many folks can be watching in "spy" mode at the same time.

2) control can be passed between all of the session attached.

Scenario:

The tech in a lab has shown some promise, so you have granted the tech non-privileged access on the local router, to reduce the number of calls about the network being down. Now, when there is a problem, the tech can investigate from the router, and perform limited testing.

One evening, the tech finds a problem, probably a routing entry, and you get the call. From home, you connect back to the network, and connect to the console.

Even though the tech is connected (in read-write mode), you can still attach in spy mode, and replay the last 20 or 60 lines of the log to see what has been viewed lately.

You're still on the phone, and the tech can show you what they have done, while you watch as well. When you agree that a routing entry needs to be modified, you take control of the session (while the tech watches), and you enter the admin password, and make the change. You exit command mode, and let the tech take the session back, and see that it all works now!

Operational Best Practices #3

- *Logging useful information*
  - Syslog can capture similar types of information, but the packets could get lost on the way to the syslog server.
  - Crackers can spot where syslog goes, and interrupt it. (They usually don't think about console messages…)
  - There is no pointer on a system that would tell a cracker where any console data may be logged.
  - After-the-fact training information

Logging is a valuable tool, as long as you don't get overwhelmed by the amount of data that is produced.

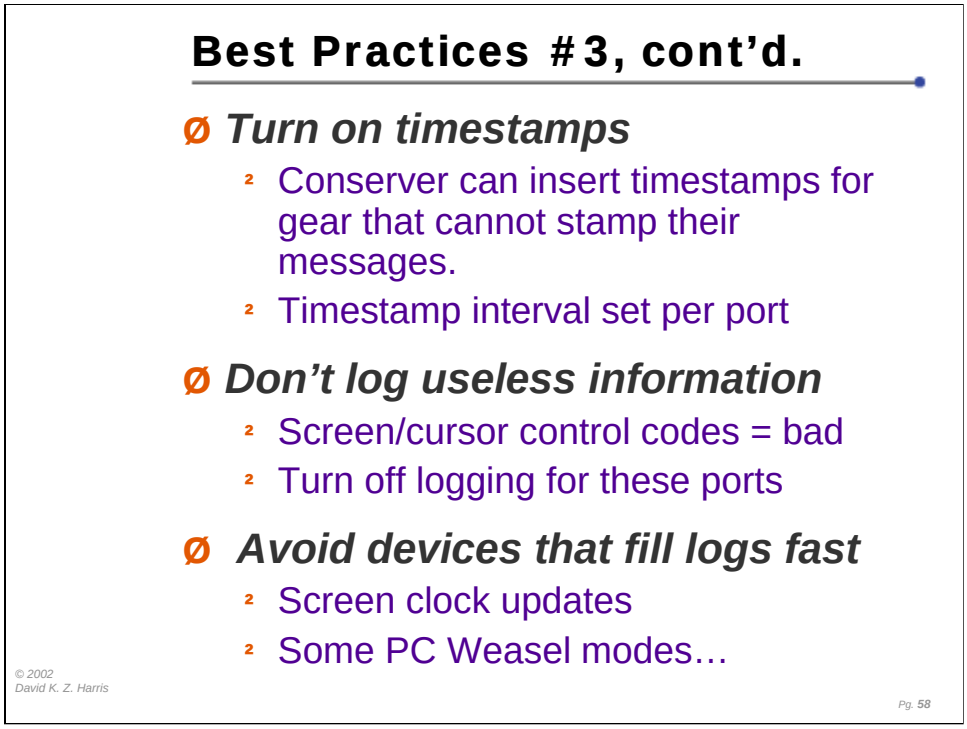
You can tune your logging, but you always walk that fine line between data overload, and making sure that you capture the important clues you need to monitor the systems, or detect failures (or attacks).

It's important to coordinate the clocks on your hosts, so that the times are useful when comparing events between different devices. (Can you correlate host failures with network events? Can failure on one host be traced to an event on another host?)

In many shops, the console is not connected to anything, or it is connected to a device that has little (or no) scrollback ability. Because of this, crackers tend to ignore that it may be attached to a write-only device (such as a printer). If they do think about it, they may try to deplete it by sending a ton of data to the console, hoping to use up the paper or ribbon.

With a conserver attached, all of that activity is logged, and `grep` will make it easier to sift through the extra noise to find your needle in that haystack!

You can also watch for Syslog to complain that it cannot log for any given reasons, and investigate those failures. (A lack of syslog data may not trip alarms on the monitoring system…)



Part of making your logs useful means striking the balance between getting enough practical information into your logs, while not filling your disks too quickly. (Of course, "disk space is cheap these days"…)

If you have gear that cannot put timestamps on it's output, you can ask Conserver to insert timestamps in individual logs, and you can also decide how often you want to insert the timestamps. You can make these choices on a port-by-port basis.

For a busy device, with lots of output, you may want timestamps every 5-10 minutes. For devices with rare output, you may want to put in timestamps only every 1-6 hours. (And you can change the granularity if you need it…if you're having trouble, you can increase the frequency until the problem goes away.)

You can have your console access without storing log files for certain ports. (That is, logging isn't "all or nothing", you can log some ports, and not log others.)

You may not want to log some devices, because the logs would be hard to read and parse later. Larscom Access-T family DSU/CSU products use cursor keys, and control codes to redraw screens. It's difficult to look through logs for these devices, trying to find what settings were changed. While console access is useful, devices that use screen controls may not warrant logging.

Operational Best Practices #4

- ***Proactive monitoring***
  - You can create your own scripts or use freeware packages to watch the most recent entries to a log file, to provide proactive alerts.
  - Some applications provide similar monitoring capability, and can send alerts to monitoring software, and/or Network Management Stations.

There are many packages available, both free and for sale, that will monitor log files and trigger alerts when things exceed pre-defined boundaries. Utilizing the log files that conserver provides can expand your view into health of your environment.

Netcool can be used to monitor the traffic in some of your logs, and incorporate that data into their system and network monitoring tools.

Swatch also plays nicely with the logs (http://www.oit.ucsb.edu/~eta/swatch/).

Scripts can be used to sift through the logs, searching for data.

You could also use scripts to watch for signs of trouble, by monitoring the log files themselves for unusual amounts of growth over a period of time. While not a perfect answer to monitoring, when a log file increases dramatically in size overnight, chance are that a sysadmin should be looking at the recent logs, looking for bad RAM, a failing disk, or just a lack of space in some partition.

Of course, it could also be important to note when a log file hasn't had any entries over a period of time. While normal for some devices, silence can sometimes be a bad thing for some systems or devices. Script monitoring can check to see when the last log entry was made. (For an active host, it could mean that a cracker has turned off logging to the console…)

Operational Best Practices #5

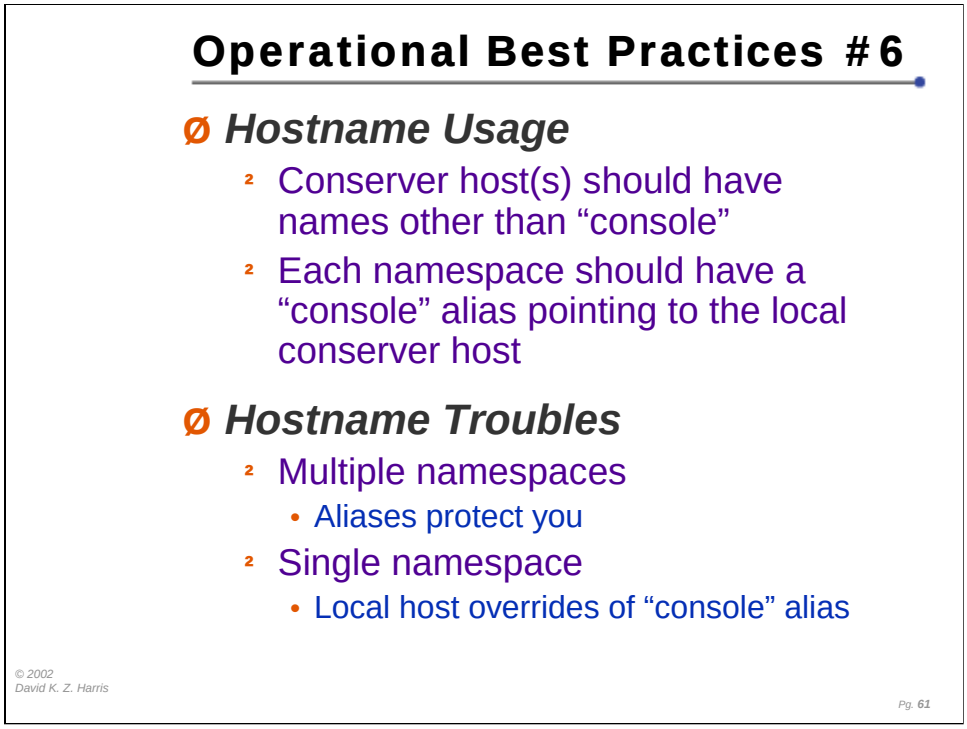
- *Forensics data*
  - When a machine crashes, your conserver log becomes your flight data recorder, capturing the last messages from the machine.
  - Looking at the recent log entries gives you some insight into what caused the failure.
  - Logs can tell you how long the failure has occurred. You can then look for similar messages to alert you to pending failures in the future.

Sometimes, devices fail. When you get the call that a device isn't responding, you may go to the console to check for a pulse. In the worst cases, you get on the console, hit the carriage return a couple times, and get nothing. Control characters don't help. "He's dead, Jim."

You could try to restart the machine immediately, but you still might not find out what killed it. (What do you tell the Director of Engineering when you are asked "What happened?" and "Well, how do you know it won't happen again?")

With conserver, before you hit the return key, you can play back the last 20 or 60 lines of the log. Think of this as the flight recorder, capturing the last statements of the pilot of a doomed flight. The log could tell you that this was a RAM problem, a full disk, or a faulty CPU. Each of these cases would require you to take different steps to get the machine back up to normal, and reduce the chances of another failure.

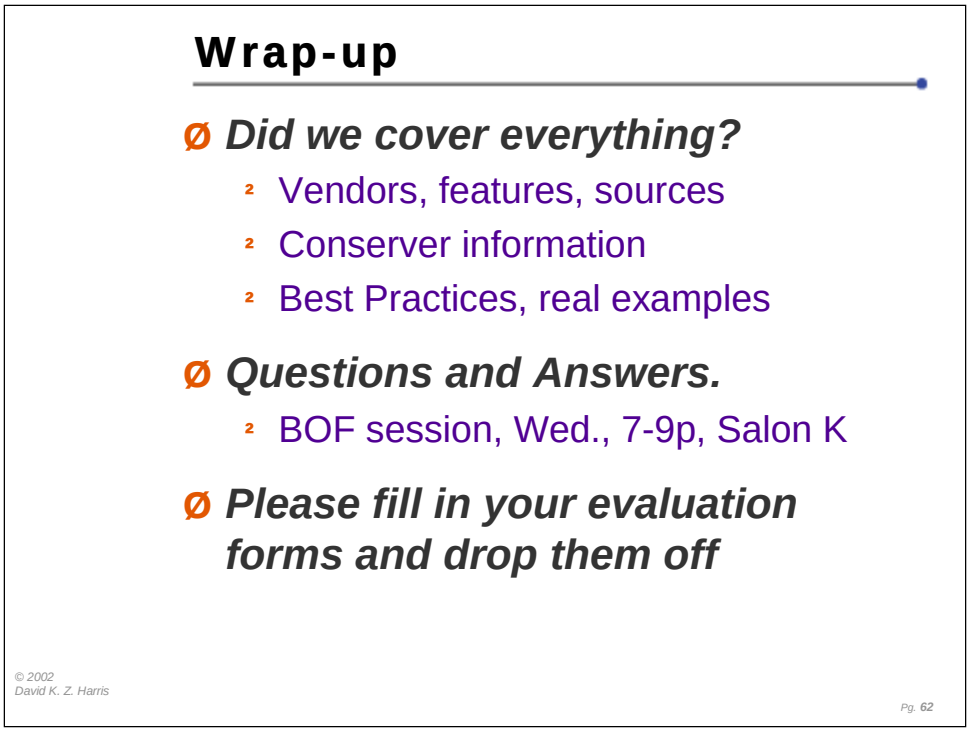
After the repair, you can use `grep`, or other tools, to search the logs for previous problems on this particular device, to see how many times it happened before, and over what period of time. You could then check the logs for similar hardware, to see if you find signs of a similar failure starting to show up on those devices!



There are a few naming conventions that you can use to ease the operation of conserver. The following assumes the default installation of “console” being the hostname used by the clients to connect to the server.

First, none of your console server hosts should have a hostname of “console”. “console” should always be an alias for the true name. This allows you to start with a single console server and add others in the future. In a multi-namespace environment, the alias “console” should exist anywhere a client might be located. For example, there should be a “console.eng.crtnty.com” and a “console.corp.conserver.com”. They both should point to their closest console server.

Where things can get tricky is when you have a single namespace and multiple console servers. You have two options for having clients find their local server: multiple console binaries (each with a different default hostname) and local alias overrides (local /etc/hosts entries for “console” can usually be made to override global definitions). Either solution will work, but you must decide on which is more difficult to maintain across your enterprise, and make sure hosts do their do hostname-to-address resolution consistently.



Hopefully I've done what I came to do, which is to teach you enough about Conserver and related Console Server hardware to decide if it can benefit you at your site. I also hope I've shared some useful practices that will improve the usefulness of your console server deployment. And I hope that I was able to answer some of the questions and concerns that the students had coming into the class.

We will have a BoF session tomorrow night, from 7p-9p, on the 5th floor, in Salon K. Bryan Stansell should be there, as should Dave Stuit. There are a few other interesting BoF sessions, but come to ours if you can!

I haven't convinced Lee Damon that Remote Access to Serial Consoles warrants a "Guru is In" session. If you have a feeling one way or the other, please let me know, or contact Lee, or the conference organizers, and let them know what you think.

While the LISA staff are interested in getting an evaluation form from all of you, I'm also interested in your responses, since I'll get some feedback from the LISA folks about how well you think I did.

On behalf of the Conserver development team, thanks for attending, and I thank you for turning in your tutorial evaluations.

David K. Z. Harris BigBand Networks

Suggested Reading

- ***Aurora Technologies***
  - http://www.auroratech.com/guide_request/guide-form.html
  - A good primer for console services, and an even-handed discussion of "Distributed Servers" versus "Console Servers plus Terminal Servers" topic
  - Email info@auroratech.com, and ask for the Guide to Multiport Connectivity for Solaris and NT.
  - Visit their vendor booth Wed/Thur

There are very few comprehensive works, explaining how to set up serial console services, which is why we developed this tutorial. However, Aurora Technologies has provided a good discussion of the topic, called "Guide to Multiport Connectivity for Solaris and NT"

The Aurora folks lean more towards a commercial server application (with support!), and they favor using many servers with multiple-port async cards installed in the servers.

While Conserver can support multi-port cards in the servers, I also feel that Terminal Servers are probably a better solution for large sites, or for sites with needs to connect many distributed ports.

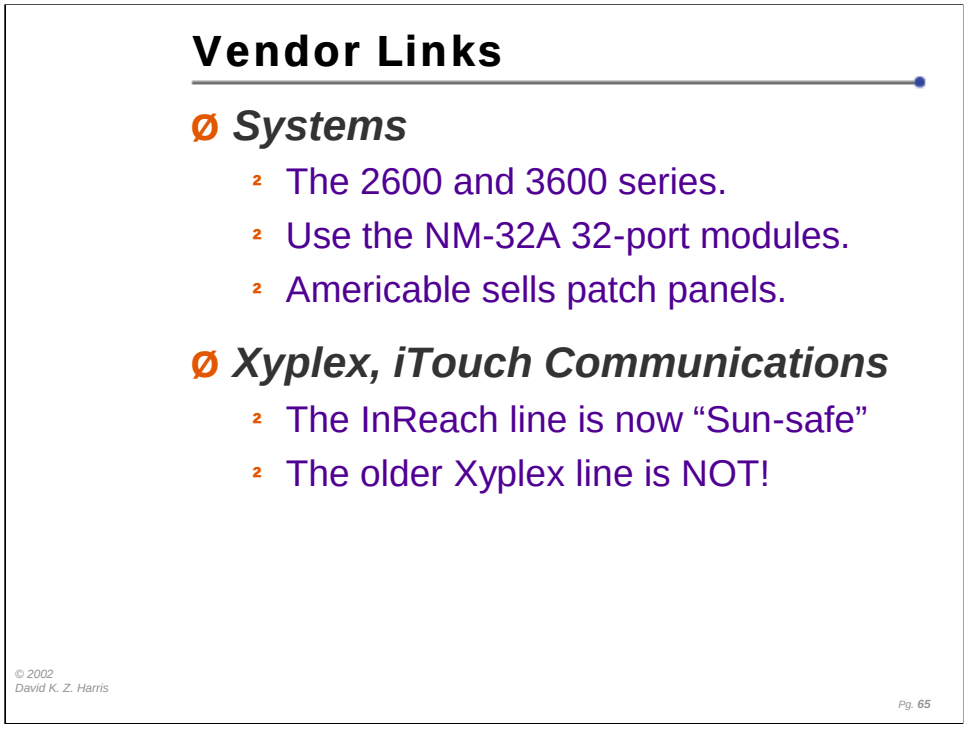
Rather than just taking my word for everything I have described here, I would encourage you to contact Aurora Technologies, and ask for a copy of their guide. Read through it, and get their side of the story as well.

Celeste Stokely has been providing a lot of useful information on her website, and you can find a good deal of information about the problem that Serial BREAK can cause a Sun CPU, as well as a vast array of systems administration topics and tutorial pages. (There are other useful pages as well. Check it out!)

Bryan Stansell has put Conserver.Com on the Internet, to make it easier for folks to find the most recent version of the Conserver code, as well as providing email lists for new version announcements, and for a users support forum.

David K. Z. Harris has been working on his serial console guides for 8+ years, and moved them to the conserver.com. These include Console connection guides, as well as some basic serial tutorial pages.

Both Bryan and David worked for Certainty Solutions (formerly GNAC), and both continue their support of Conserver while pursuing new efforts.



Terminal Server Vendors

Cisco Systems http://www.cisco.com

http://www.cisco.com/univercd/cc/td/doc/pcat/2600.htm

http://www.cisco.com/warp/public/cc/pd/rt/2600/prodlit/2600_ds.htm

http://www.cisco.com/warp/public/cc/pd/rt/3600/prodlit/seral_ds.htm

http://www.cisco.com/warp/public/cc/pd/rt/3600/prodlit/3600_ov.htm

Xyplex (was nBase, now iTouch Communications, becoming MRV)

http://www.itouchcom.com/products/index.cfm?cat=scs

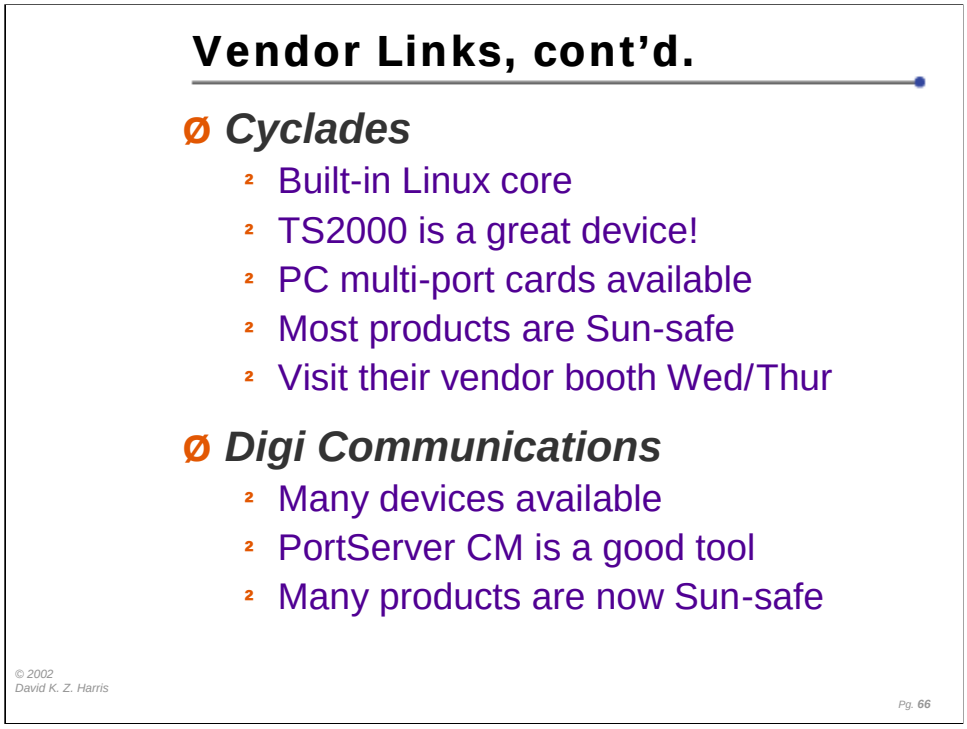
http://www.itouchcom.com/news/display.cfm?nwid=2002_07_01

Adapter Vendors

Americable 800-328-7954 http://www.americable.com

http://www.conserver.com/consoles/ciscokit.html

(also annexkit, iolankit, and xyplexkit)



Cyclades and Digi are both long-time makers in this arena.

Cyclades was the first (and only) console server hardware we found in our Serial BREAK testing that had a software-controlled BREAK problem. Once the recipe was sent on how to demonstrate the problem, we received a patch within a few days which resolved the issue. The TS2000 hardware was also enhanced during the design phase to ensure that the BREAK problem did not exist in the final product.

Digi has also entered the Console Server hardware arena, with the PortServer CM product line. The CM32 is also a good fit with Conserver. The existing PortServer Terminal Server line is also still available.

Cyclades

http://www.cyclades.com/products/ts_series.php

http://www.cyclades.com.pe/Productos/Anexos/ReleaseSerieTS.txt

Digi

http://www.digi.com/solutions/devtermsrv/cm/index.shtml

http://www.digi.com/solutions/devtermsrv/termsrv/index.html

http://support.digi.com/support/techsupport/hardware/portserver/sun_console.html

Vendor Links, cont'd.

- *Perle (Perle Systems Ltd.)*
  - CS9000 is Sun-safe
  - Cables, status LEDs on same side
    - Good or bad? You decide…
  - Good integration with MS Windows
    - May be useful in a mixed environment
- *Lantronix*
  - Still a workhorse in the industry

Perle System took on the Chase Research IOLAN line, as well as Specialix products. You can still get support, and buy hardware from Perle if you want these lines. (The IOLAN model 102 and 104 servers are Sun-safe.)

The Perle CS9000 is comparable to the Cyclades TS2000, and the Digi CM32, in that they are all 32 ports, Sun-safe, I rack-unit tall, lightweight. One difference that stands out, to me, is that the transmit and receive data LEDs on the CS9000 are integral to the RJ-45 jack, putting them on the same side of the unit as the cables. (On the Cyclades and Digi units, the LEDs are on the opposite side from the jacks.) Whether this is a benefit or a deficit in your mind will likely depend on how you decide to mount the units.

Lantronix has been making serial-to-network interfacing devices for over a decade, and should also be on your list of candidates to investigate.

Perle

http://www.perle.com/products/prod_family/console_server/cs9000.html

http://www.perle.com/products/resources/pdfs/CS9000%20Dis%20paper.pdf

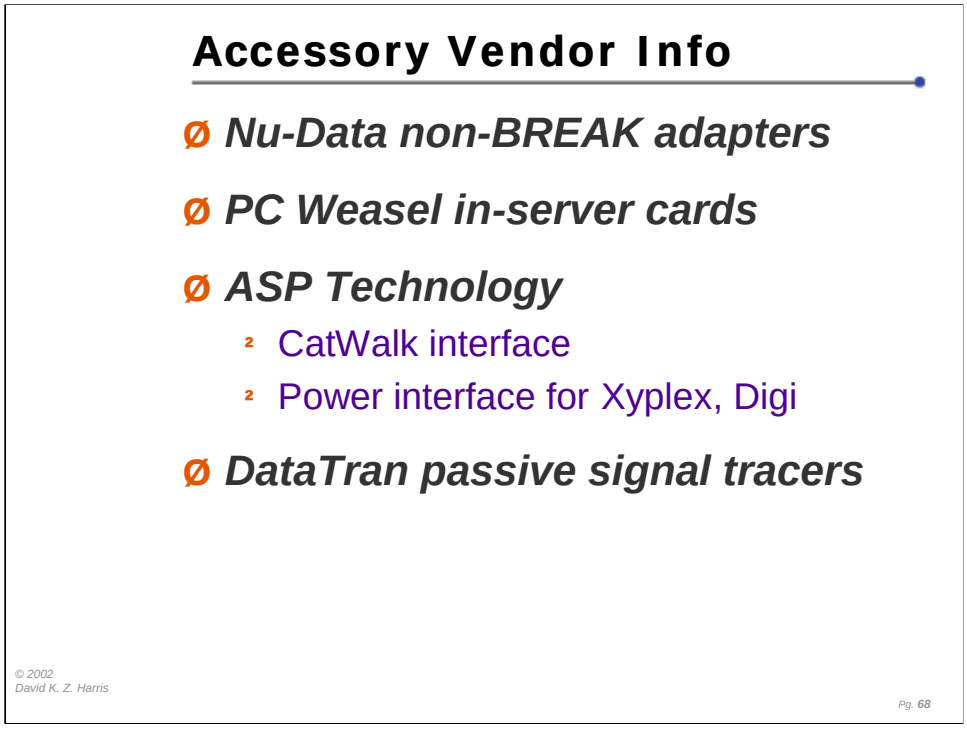
http://www.perle.com/products/prod_family/serial_servers/iolan_pl.html

Lantronix

http://www.lantronix.com/products/cs/index.html

http://www.lantronix.com/learning/tutorials/ds.html

http://www.lantronix.com/learning/wp/conserv_wp.html



Nu-Data is the sole source for a "Non-aborting Serial Console Adapter" (part number 4723), but their website PDF links have been broken for longer than I can remember. The units cost about $100 per port, which is OK if you only have a few devices that you need to protect. If you have 10 or more devices to protect, you should consider getting a Sun-safe console server device instead.

http://www.nudata.com/ v) 800.844.5757 f) 732.905.5708

http://www.nudata.com/workstationproducts1.htm

Real Weasel produces the PC-Weasel cards. This is the sweetest solution I know for MS-OS machines, and their PCI version has been out for about a year now. It gives you soft power control, and gets around the BIOS limitation in systems with Smart NICs and Smart drive controllers. They also have a couple other references on their web site for other hardware solutions, but none is as sweet, in my opinion. Clever Canucks! Try the web demo!

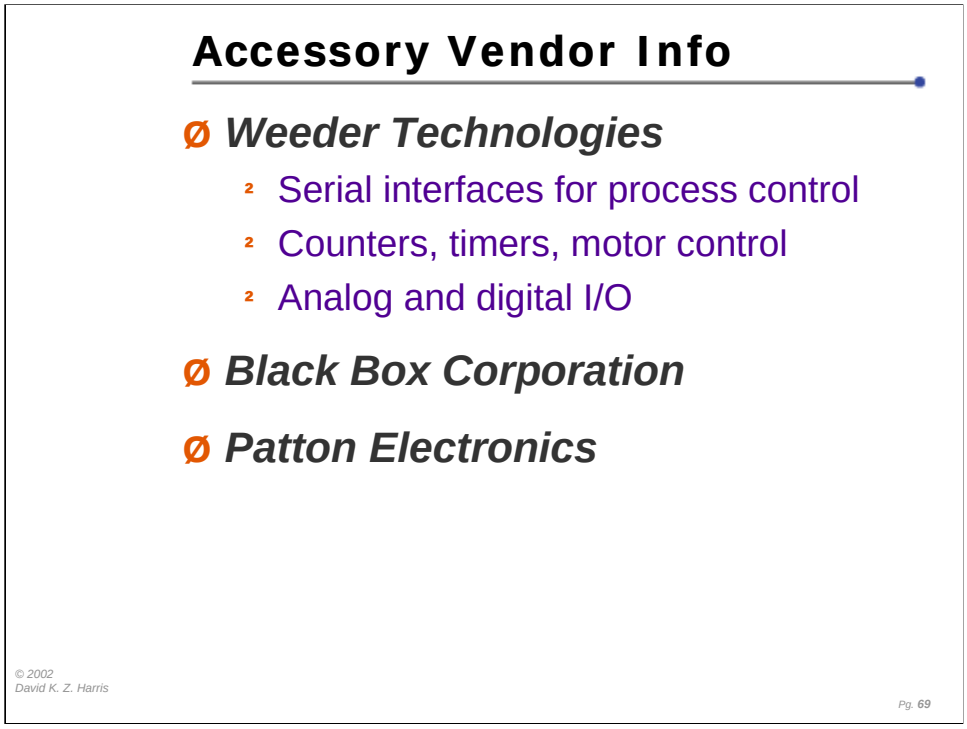
http://www.realweasel.com/ v) 403.705.2025 f) 403.705.2026

http://www.realweasel.com/oph.html

ASP Technology sells a console server application, but they are clever hardware hackers as well. The in-line dongles for the power leads on Xyplex and Digi product make them Sun-safe, but the CatWalk lets you put a local terminal on sensitive hosts, while leaving them connected for remote access by console server hardware. Very clever indeed.

http://www.asptech.com/ v) 970.686.1211

DataTran has passed on. Rest in peace. But if you can get your hands on their Passive RS-232 signal trackers, DO IT! The going cost was $25(US) for 8-signal DB-9 and DB-25 models.



Weeder Technologies

If you are looking for ways to get information from devices that don't have consoles, check out this site. While there are a few places to find process control interfaces, this is probably the best site. Useful info, compact products, and a good variety should fill many of your unusual needs, and help you put a console on hardware.

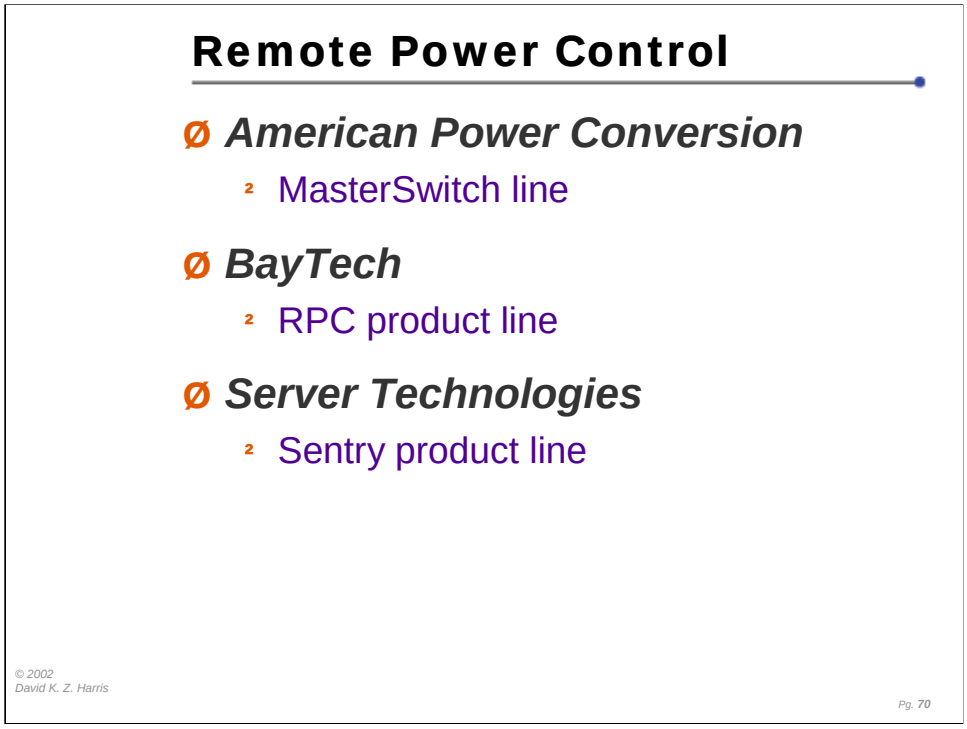
http://www.weedtech.com/ v/f) *850.863.5723*

Black Box has always been a favorite resource. They OEM and rebrand many products, and provide a wide array of interfacing devices. I've also used their catalogs as teaching materials, because they have useful block diagrams, and informational notation for using many of the products. Consider getting a printed catalog for your bookshelf.

http://www.blackbox.com/ v) **724.746.5500 f) 724.746.0746**

Patton Electronics is another vendor with a wide array. The problem with that is that it's hard to find a solution on a website if you don't know what the solution is already. Luckily, you can easily request a printed catalog from their web site.

http://www.patton.com/ 301.675.1000



American Power Conversion has been a long-time UPS maker, and a leader in developing SNMP management for UPS gear. Their MasterSwitch product line allows for serial port access to power outlets, as well as telnet and HTTP control.

http://www.apcc.com/

http://www.apcc.com/products/family/index.cfm?id=70

http://www.apcc.com/support/contact/index.cfm

BayTech has been developing power control products for nearly a decade, but they've been working on digital equipment for more than 25 years. Their remote control power strips can be controlled via a serial connection, or some have an integrated modem.

http://www.baytech.net/cgi-private/product/ v) 800.523.2702

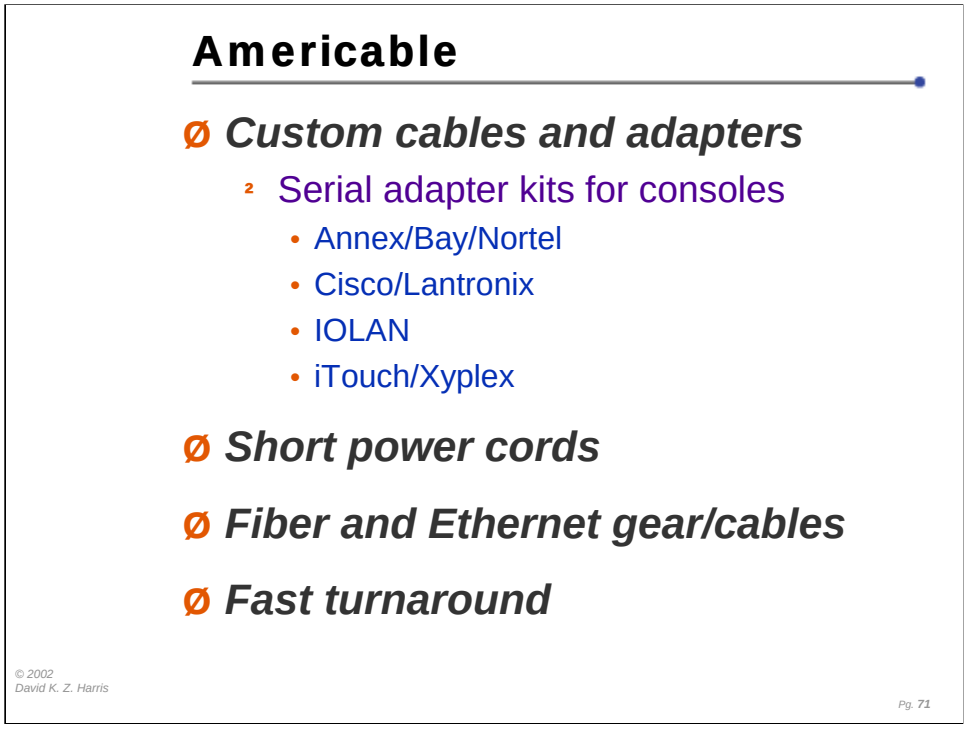
http://www.baytech.net/cgi-private/product?catagory=RPC+SERIES

http://www.baytech.net/support/demoinst.shtml

Server Technologies is new to me, but their Sentry line looks very interesting. It's larger than the APC and BayTech units, but it also can switch more power. (Both APC and BayTech have some units witch can switch lots of power. If you need to switch high-current loads, do your homework and check all three!) I like the Power Tower power strip, but other units also switch a console port as well as switching a power connection.

http://www.servertech.com/ v) **775.284.2000**

http://www.servertech.com/products/vac/vacvertical.htm



I've used Americable as a vendor for more than 6 years, and I've been very happy with them as a vendor. They've been able to turn some important orders around very quickly. Some of my other vendors haven't passed this test…

Americable has also worked with me to define and stock special serial adapters for Console Server use, as well as making special adapter and cable bundles for the Console Server vendors listed above. These kits have helped many folks get consoles connected to their networks quickly. But what is important to me is that they made the effort to define and build these parts and kits, before there was a demand, based on my requests. As a vendor, they make efforts to provide or acquire whatever we've been looking for. In many cases, buying something through Americable has been cheaper than buying it on my own from the manufacturer.

It's amazing how much short power cords and appropriate-length cables can clean up a rack!

My contact has been Steve Vacik (svacik@americable.com, x-3824), but I like everyone that I've met at Americable.

http://www.americable.com/ v) 800.328.7954 f) 952.944.8021

http://www.conserver.com/consoles/ciscokit.html