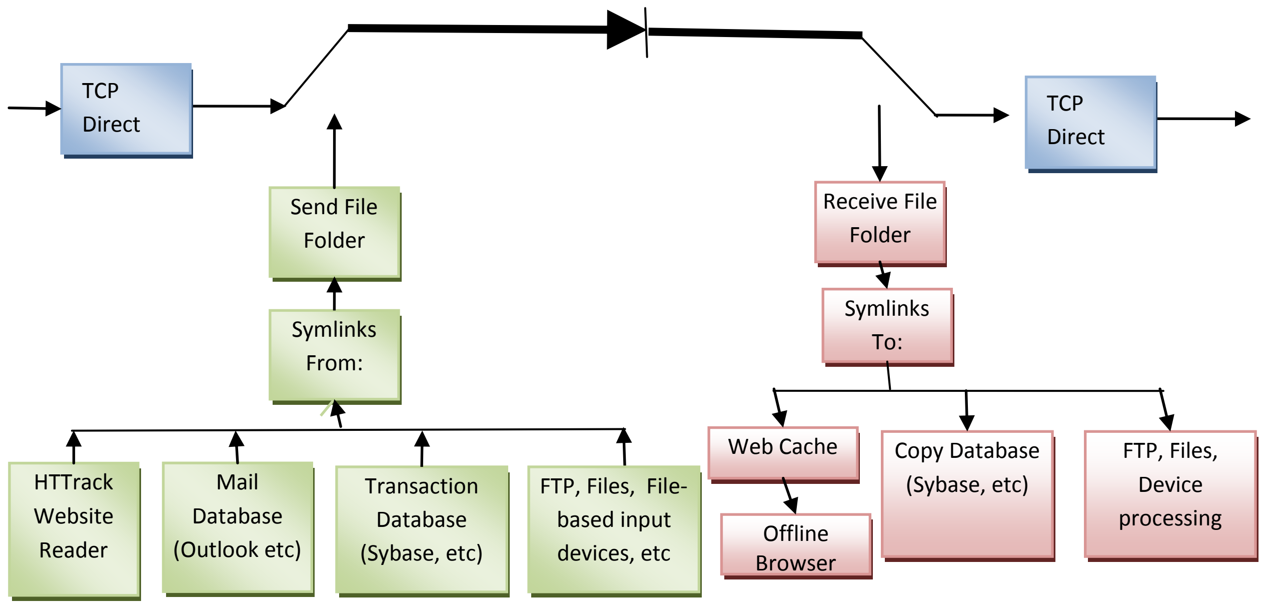
AROW Data Diode

AROW Data Diode Software Structure

Data Diodes can perform most of the same functions as normal routers but usually with the aid of some support software. This software is used to marshall the data so that it can be accessed as if it were presented by a normal router.

System Administrators must make data available to users in response to requests that are extra to the normal user interaction with applications, for example, requesting web pages or database access. Marshalling software usually makes copies of data on the uncontrolled side of the diode, perhaps with extra filtering and data inspection. The basic form of software applications supporting data diodes is shown here:



TCP Direct
Send File Folder
Symlinks From:
HTTrack Website Reader
Mail Database (Outlook etc)
Transaction Database (Sybase, etc)
FTP, Files, File-based input devices, etc
Receive File Folder
Symlinks To:
Web Cache
Offline Browser
Copy Database (Sybase, etc)
FTP, Files, Device processing
TCP Direct

Data can be sourced directly from and to a TCP stream, or from and to management software. The principle of AROWBftp management software is the transfer of files.

Data in file format can be transmitted across the AROW and reconstructed seamlessly on the protected side. So application support software consists of conversion to a consistent file structure.

somerdata

Somerdata Ltd
1 Riverside Business Park
St Annes Road
Bristol
BS4 4ED
UK

Phone: +44 (0)1179 634050
Fax: +44 (0)1173 302929
E-mail: sales@somerdata.com
Website: www.somerdata.com

Operating systems and Languages

AROW software needs to support computer operating systems of all flavours. the most popular in enterprise network infrastructures used in secure networks are UNIX and Linux. Linux is rapidly gaining popularity because its open source nature allows very fine control and intimate auditing of its performance. However, less critical networks are the domain of Microsoft's Windows system, especially in office-data type networks with relatively simple but large databases of information.

Designing software specifically for an operating system leverages the native speed, features and application integration of each OS, but at the cost of having to write and maintain applications across each. This is a slow and expensive process, that in many cases results in constant system maintenance, upgrades, flaws and slower operation.

It also means that users are in the hands of their suppliers, so individual tailoring of applications is expensive or impossible.

AROW software uses the Python language. This is a scripted, interpreted language that operates at a high level through a local interpreter for the underlying operating system. This means that it is notionally intrinsically slower than software written for the native platform. However, the Python language has evolved considerably over the last few years and its interpreters have been extensively optimised, such that operational speed differences are now very close to native speeds. Coupled with the improvements in hardware platforms, the use of multiple cores and clustered systems, the issue of speed in Python has largely disappeared. The Python language is now used by many leading service providers to deliver high speed high quality and fast developed products.

The advantage of using Python to support AROW is that these scripts are open – anyone can examine them tio ensure that they are only doing what they should be, and they can be easily modified by users or system integrators to tailor the application to a specific users system.

AROWBftp

AROWBftp is the name given to the collection of AROW support scripts that can perform the data management tasks required to use AROW.

The fundamental requirement of AROWBftp is to serialise data from a number of sources, marshal it into a transmittable form, then recover the data back to its original form. The goal of AROWBftp is to make the data appear to the protected user as if they were connected to the unprotected network.

So how is this achieved?

The first requirement is to protect the data from network congestion corruption. In a normal TCP layer, this is achieved by a 3-way handshake – data is sent by the sender and acknowledged by the receiver. If no acknowledgment is received, the sender knows that its data has failed and re-sends the data. This method ensures that received data is always correct.

This option is not available to Data Diodes since there is no return path to provide the acknowledgment. The sender has no way of knowing that data has been successfully received. AROW mitigates this problem by providing considerable internal buffering and TCP layer server firmware that at least ensures that connections into and out of the Diode are error-free, with the internal buffers being used to allow re-send of data in times of network congestion.

AROWBftp includes a number of strategies to enhance data integrity further.

Redundancy

The most obvious way of ensuring that data is received correctly is to send it more than once. A comparison can then be carried out to check to see if the multiple copies agree. This is a crude but effective method, often used with UDP data. AROWBftp *frames* data to be sent using an internal format that includes a header with such information as file and frame size, timestamp and a crc checkword for each frame that allows each transmission frame to be checked. The receive software examines all of the frames for integrity and holds a local copy of each frame until all of the frames in the file have been received correctly, then it recreated the file, stripping the header and framing information. In this way, if one or more frames are detected as errored, equivalent frames from the next redundant transmission are used to re-construct the original file.

The choice of redundancy, whether to use it, how many times the same file should be re-transmitted is down to the network conditions and the required integrity of the data. It might reasonably considered that transmission of a database containing valuable financial or personal data is more important than the transmission of a video link or web page, therefore more redundancy might be employed. In general, if redundancy is employed, a number between 2 and 4 is sufficient. Of course, the level of redundancy directly affects the transmission bandwidth of the data link, so in the case of a 4 times redundancy transmission, the data link effective bandwidth drops to one-quarter of the raw bandwidth.

Manifest Checking

While there is no direct path from the protected to unprotected sides of AROW, it may be appropriate to generate a manifest of files sent across the link, and create a manifest of files actually received, then compare the two manifests. Discrepancies detected can then be alerted to the network administration who can then re-send any files that were not received correctly. This necessitates establishing some ling between the protected and unprotected sides that potentially could be exploited, but could take the form of a simple text file emailed back to the unprotected side administrator, or another form of protected data transfer.

AROWBftp supports the generation of such a manifest, as a XML file that can be compared to a manifest generated on the received side. The format of this manifest includes timestamping, number of times sent and other management information that can be used in a file comparison. A provided rest tool can then be run on the unprotected side, to reset file attributes in AROWBftp so that it knows to resend files that it previously had marked as already sent, and therefore excluded from transmission.

This reset can be done by date, by file type, by comparison with a received manifest or by searching on particular file attributes such as name or type wildcards etc.

Data Framing

Data framing used in the redundancy reconstruction process in itself provides a powerful error checking capability. Equally important as providing correct data across the link is *not* providing *incorrect* data. In most cases it is better to have no data than wrong data. the framing scheme employed by AROWBftp file data includes a CRC check, which if failed means that that frame, and therefore its content, will be discarded. AROWBftp recognises several types of data –Link keep-alive ('heartbeat'), file data, tcp stream data and udp stream data. In the case of files, wrong data is unacceptable, and therefore should be discarded, but in the case of stream data, there may be higher (application) level protocols that include error correction or concealment. In these circumstances, it may be appropriate to pass through apparently errored content and so this option is used in AROWBftp receive software.

Python Scripts

AROWBftp consists of a number of Python scripts and so it is necessary for the operating platforms to install and have available the standard Python interpreter (Python 2.7). The scripts are divided into Send and Receive functions, although many of the helper scripts are common to both. AROWSend.py is responsible for analysing the data to be sent, formatting it into frames, managing AROW TCP socket connections and controlling the rate of transmission. The main scripts expect parameters to be provided that describe the required operating conditions (AROW's TCP/IP address for example) and these are detailed in the operating manual. Once started, the AROWSend script can be left to continuously monitor and manage the file and data transfer process. AROWSend expects to find data files in one location (folder /directory) and, will transmit and re-transmit all folders and files found in and under this root folder. Clearly it is not practical to move all user data to a single folder, so the normal practise is to create a series of symbolic links to folders and files on the source network that are required to be accessible on the protected side.

AROWSend regularly scans the root folder to check for changes, additions or deletions and adjusts its transmission tree appropriately. Once sent, (subject to redundancy settings) a file or folder structure will not be transmitted again unless the file is changed. The frequency of scanning can be set by the user. Files that are symlinked will also be sent only when they have changed.

AROWSend also includes TCP and UDP data ports that can be used to send these data types multiplexed into the serial data stream, such that TCP, UDP and File data can be sent, sharing the available bandwidth.

The AROWRecv.py script is responsible for accepting data from AROW and therefore continuously monitors the socket connection for new data. On receipt of data, it is checked for frame integrity and data type, segregated and de-framed before being presented. For file data, this takes the form of a single reception folder/directory, where the sent file structure is duplicated in exactly the same form as that originally sent. Clearly it is necessary to ensure that this file structure, with its symlinks, is capable of being written to the appropriate locations, so file permissions will have to be managed appropriately. For TCP and UDP data, AROWRecv creates a local server that allows the connection of the appropriate client for further data processing.

It should be noted that all data transmission is transparent – if management or error correction information is added to the source data it is removed before being presented for use on the protected side.

Any of these scripts can be adjusted to suit individual needs, or extended to add functionality. Typical customisation might be

file filtering, to check that files have not been tampered with;
batch processing as part of a cron job,;
multiple stream handling;
unusual protocols and bespoke encryption;
integration into existing security applications

Symlinks

Use of Symlinks (Symbolic Links). Standard on Unix/Linux systems and available on Windows NTFS systems since Server 2003, Vista and some versions of XP, these are simply links to the files in a system that are treated as if they were the files themselves. Thus access, copying, deleting, moving etc can be accomplished (subject to permissions) as if directly operating on the files. For example, a symlink can be created to a mail database, placed in AROWBftp's Send folder, and all of the database files will be copied to AROWBftp's Receive folder on the protected side of AROW. Symlinks placed in the Receive folder can be used to direct the received database files to their final destination for user access.

Web Access

Similarly Web pages can be transmitted after conversion to a file format by applications such as HTTrack. Again, symlinks can be used to control source and destination paths, and a standard web browser in offline mode can be used to access the cached files on the protected side of AROW.

Other sources

This principle can be applied to any file-based transaction including hardware input devices, such as webcams and data acquisition devices.

Where possible, we recommend open source, cross-platform software that can be examined and audited

Useful open source applications: HTTrack (for website copying)
, FFmpeg, (for multimedia streaming)
Netcat (and it's Windows port ,Ncat) for direct data streaming
Winbolic (symlink manipulation tool).

somerdata

Somerdata Ltd
1 Riverside Business Park
St Annes Road
Bristol
BS4 4ED
UK

Phone: +44 (0)1179 634050
Fax: +44 (0)1173 302929
E-mail: sales@somerdata.com
Website: www.somerdata.com