5K-4

ネットワークを利用したカーネルデバッガ対応コンソールの設計と実装

濱 善幸† 佐藤 喬† 多田 好克†
†電気通信大学 大学院情報システム設計学専攻

1 概要

情報家電などに組込み計算機が利用されている。このような組込み計算機上で動作するカーネル等の開発には、コンソールやデバッガの用途でシリアル端子が使われてきた。その小型さゆえ、ビデオ出力やキーボード入力を持たない場合が多いからである。

しかし、近年組込み計算機が普及する中で、さらなるの小型化や低コスト化の要求がある。これに伴いシリアル端子も省かれるようになり、カーネル等の開発に問題が生じている。

そこで、本研究では組込み計算機でも普及が進むイーサネット端子等を利用したカーネルデバッガ対応コンソールの設計と実装を行った。具体的には、カーネルデバッガとコンソールのいずれにも接続可能な I/O モジュールを作成した。

2 システム概要

2.1 システム全体像

本システムは、シリアル端子を持たない組込み計算機での開発で使用することを目的に、シリアル端子経由で行われていた通信をイーサネット経由の通信に置き換えるものである。今回の実装では、コンソールとカーネルデバッグに対応することとする。

コンソールとして使用するには、実行する命令の入力や実行結果の表示を行う入出力機構、命令を実行する機構、両者を繋ぐ通信経路が必要である。またカーネルデバッグを行うには、メモリ内容の解析などを行う target 上で動作するカーネルデバッガ、カーネルデバッガで実行する命令の入力や実行結果を表示する機構、両者を繋ぐ通信経路が必要である。

本研究では、組込み計算機で使用されてきたシリアル端子を経由する通信経路に代わり、イーサネットを経由する通信経路を新たに作成した。

2.2 システム構成

図 1 に示すように、開発用計算機 (以下 host と呼ぶ) と開発対象計算機 (以下 target と呼ぶ) が 1 対 1 で存在し、UDP/IP を利用できるようなネットワーク接続がなされている状態で実装を行った。OS は Linux である。

本システムの主な構成要素は、appli_connector、netpoll_connector、netpoll、netcontroller_io である。イーサネットによる通信経路の置き換えを行うには、パケットの送受信機構と target にて受信したパケットからメッセージをアプリケーションに伝える機構が必要である。この前者の機構を構成するのが、appli_connecter、netpoll_connector、netpoll であり、後者の機構を構成するのが netcontroller_io である。

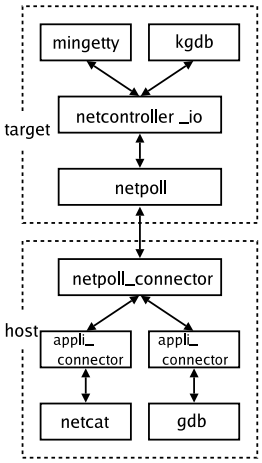

図 1: システム構成

2.3 通信の流れ

通信の流れには、host から target に通信する送信と target から host に通信する受信の 2 種類がある。本節では、これらの処理の流れを示す。

なお本システム構成要素である netpoll には、指定した IP や port 番号としか通信できない、同一ホスト内で netpoll を重複起動できない、等の問題がある。これにより、複数のアプリケーションと通信する場合に不都合が生じている。

そこで、パケットの本文中にパケット送信元の port 番号を埋め込み、識別することで複数のアプリケーションと通信可能にした。

"Design and implimentation of console for kernel debugger using network"
†Yoshiyuki Hama, Takashi Satou, Yoshikatsu Tada
†Graduate School of Information Systems, The University of Electro-Communications

2.3.1 送信

1. appli_connector : host 上のアプリケーションからのメッセージ先頭に、当該 appli_connector の port 番号を付加し、netpoll_connector に送信する。

2. netpoll_connector : appli_connector から受け取ったメッセージを、netpoll へ転送する。

3. netpoll : netpoll_connector から受け取ったメッセージから port 番号を読み取り、取り除いた後、target 上の対応アプリケーション用の受信バッファにメッセージを積む。このメッセージは netcontroller_io の read 関数などにより、対応アプリケーションに届けられる。

2.3.2 受信

1. netpoll : target 上のアプリケーションからのメッセージ先頭に、対応する appli_connector の port 番号を付加し、netpoll_connector に送信する。

2. netpoll_connector : netpoll から受け取ったメッセージから port 番号を読み取り、対応する appli_connector に送信する。

3. appli_connector : netpoll_connector から受け取ったメッセージから port 番号を削除した後、host 上の対応アプリケーションに送信する。

3 netpoll

netpoll とは、完全なネットワークサブシステムと I/O サブシステムを利用できない状況で、カーネルがパケットを送受信する事を目的とした I/O である。このような有意な特徴からパケット送受信には netpoll を使用することとした。

パケットを受信した時、netpoll の受信パケット処理関数は、メッセージ中の port 番号に基づき、対応するアプリケーション用のバッファにメッセージを積む。

またパケットを送信する時、メッセージの先頭に、対応する appli_connector の port 番号を付加して送信する。

4 connector

appli_connector と netpoll_connector は、host 上のアプリケーションと target 上の netpoll 間のパケット中継アプリケーションである。

先に示したように、本システムではパケットに port 番号が必要であるが、アプリケーションから送信されるパケットには付加されていない。そのため、アプリケーションからのパケットを受け取り port 番号を付加する機構が必要である。

また taregt から送られてくるパケットが、どのアプリケーションに宛てたパケットなのかをパケットに付加された port 番号から識別し分類する必要がある。

そこで、これらの機能などを有する 2 つの connector を作成した。なお、target 上で動作させると netpoll 導入のメリットがなくなる等の理由から、connector は host 上で動作させることとした。

5 netcontroller_io

netcontroller_io とはファイルなどを操作する関数へのポインタを保存する構造体である。Linux では、デバイス操作はこのような構造体に示される関数を経由して行われる。

コンソール、カーネルデバッガとして利用するには file_operations と、kgdb_io が必要である。この 2 つを別々に用いることを考えていたが、最後に有効化した構造体が他方の構造体への操作要求を奪うという現象が発生した。そのため、この 2 つを統合した netcontroller_io を作成した。

6 動作確認

基本的な動作確認として、コンソールとカーネルデバッガを 1 つずつ動かした場合に正常動作するかを確認した。netcat の仕様に基づく操作困難性はあったが、それ以外の反応速度や表示速度に問題は無かった。

また、コンソールとカーネルデバッガの 2 つを動かした場合に正常動作するかを確認した。カーネルデバッグ中はカーネルデバッガがコンソールの動作を止めるためメッセージを送っても動かないが、メッセージ自体の受信は行われておりデバッグ終了後正常に動作した。

7 関連研究

イーサネット経由でコンソールやカーネルデバッグを行う Etherconsole[1] や kgdboe[2] など、各機能を単体で実現するものは多いが、拡張性のあるシリアル端子の置き換えとなるものはない。

参考文献

[1] Kistler,M., Hensbergen, E.V. and Rawson, F.: Console over Ethernet,in Proceedings of the FREENIX Track:2003 USENIX Annual Technical Conference

[2] kgdboe http://kgdb.linsyssoft.com