Software Streaming via Block Streaming

Pramote Kuacharoen*, Vincent J. Mooney III[+] and Vijay K. Madisetti[&]
{pramote, mooney, vkm}@ece.gatech.edu

[+]Assistant [&]Professor, *School of Electrical and Computer Engineering
[+]Adjunct Assistant Professor, College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA

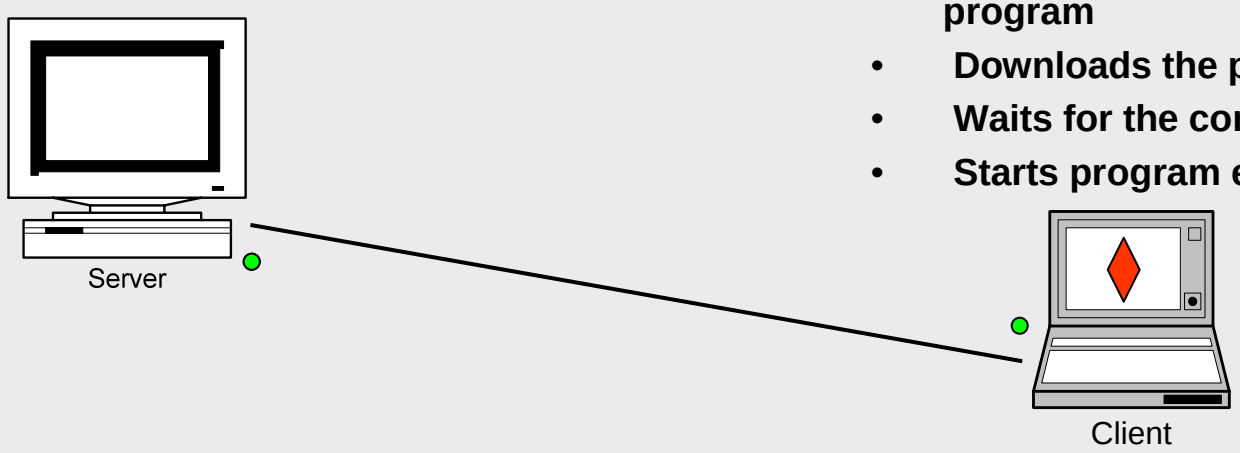
This research is funded by the State of Georgia under the Yamacraw initiative.

DATE Conference, March 6, 2003

Outline

- Introduction
- Objectives
- Related Work
- Block Streaming
- Performance Analysis
- Simulation Results
- Conclusion

Introduction



- **Sends a request to download a program**
- **Downloads the program**
- **Waits for the completion of download**
- **Starts program execution**

- Download time can be very long, delaying the program execution
- Resource Utilization may not be efficient since unneeded parts are also downloaded

Objectives

- To stream software (especially embedded applications) to remote devices
- To reduce the amount of time from when the application is selected to download to when the application can be executed (application load time)
- To reduce the amount of time when the application is suspended or stalled during execution due to missing code (application suspension time)
- To efficiently utilize resources such as bandwidth and memory
- To support a wide range of applications (real-time and non-real-time applications)
- To facilitate software updating since the latest version of software is always downloaded to the client device

Applications for streaming

- Likely to change over time to support new functionality such as game software
- Have many features, only a few features are needed such as financial software
- Run on a device which has limited resources

Related Work (1)

- Java applet implementation
  - Requires JVM
  - Allows applets to run without obtaining all classes
  - Uses on-demand downloading for each class file, potentially making too many connections and causing the application to be suspended while a class file is being sent
  - Can avoid making too many connections by bundling and compressing class files into one file, which in turn delays program execution
- Our method uses both background streaming and on-demand streaming

Related Work (2)

- Raz, Volk and Melamed [7]
  - Divides an application into a set of modules
  - Replaces functions in each module w/ stub functions
  - The streaming behavior varies depending on the size of each module, causing difficulties in predicting application suspension time
  - Target: general purpose computers
  - Streaming at the level of modules
- In our method, the size of delivery units can be fixed and we support embedded devices

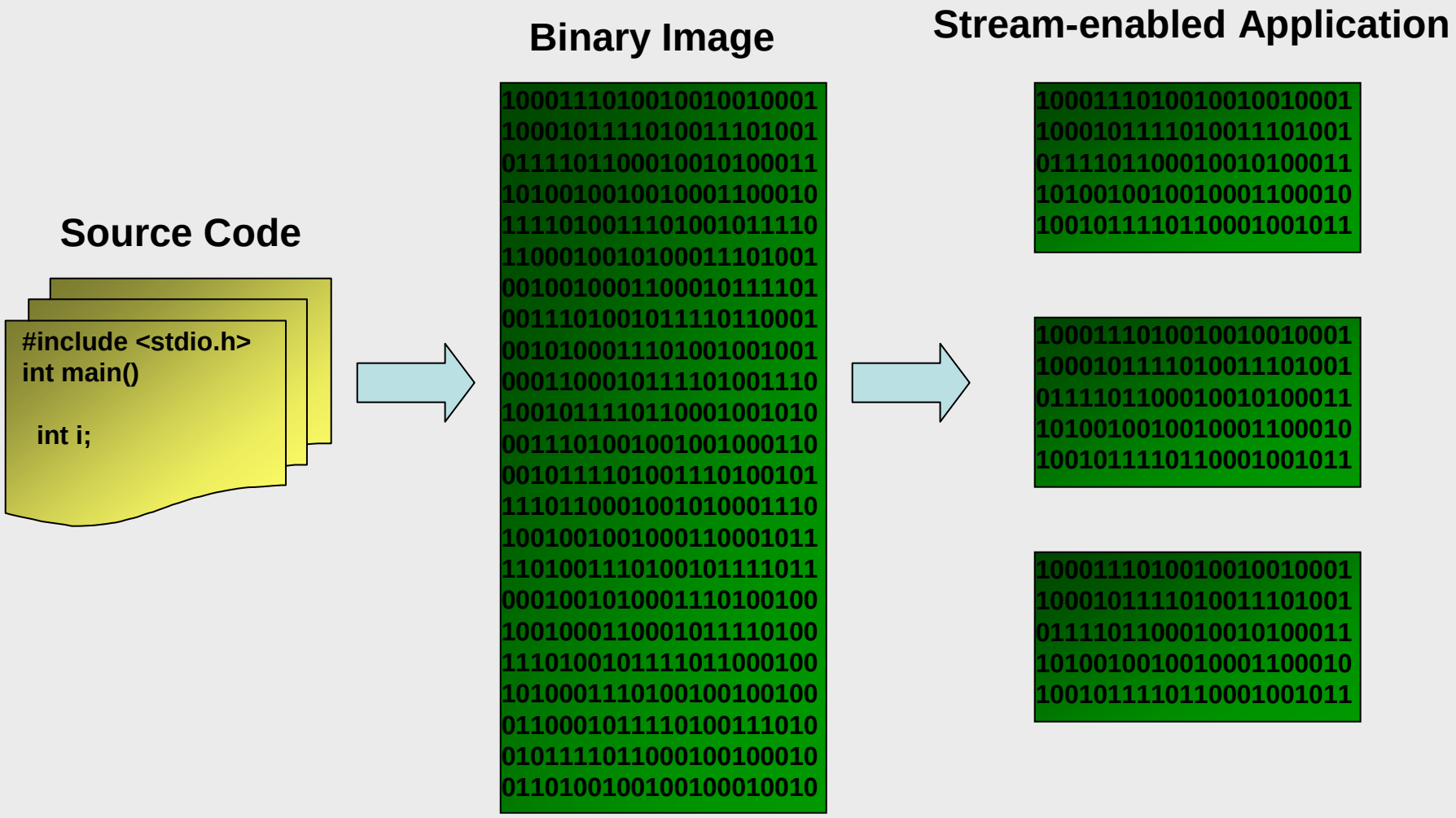
Related Work (3)

- Eylon et al. [8]
  - Divides an application into fixed size of streamlets
  - Requires virtual file system
  - Requires OS support
  - May not be applicable for a small memory footprint and slower or lower-power processor embedded device
- Our method does not need a virtual file system

Related Work (4)

- Source code streaming [9]
  - Sends the application source code to the devices, exposing the intellectual property contained the source code
  - Compiles the application at load time, increasing the application load time
  - Requires a compiler at the client device which occupies a significant amount of storage space
- Our method does not expose the intellectual property and does not need a compiler to reside at the client

Block Streaming Concept



Block Streaming: Binary Instructions

- Instruction categories
  - Arithmetic (e.g., add, subtract)
  - Logical (e.g., and, or, shift)
  - Data transfer (e.g., load, store)
  - Conditional branch (e.g., branch less than, branch on equal)
  - Unconditional branch (e.g., return, branch)
- Exiting and entering a block
  - A branch or jump instruction
  - A return instruction
  - An exception instruction
  - After executing the last instruction of the block

Block Streaming: Handling Branches

- We called a branch instruction that may cause the processor to execute an instruction in a different block an *off-block* branch
- Code modification
  - Branches: modify if is an off-block branch
  - Last instruction of the block: if the following block is not yet loaded **and** the last instruction is not a goto or a return instruction, add an instruction to stream in the following block
  - Return instruction: do nothing, caller code already in memory
  - Exception instruction: stream exception code prior to the current block

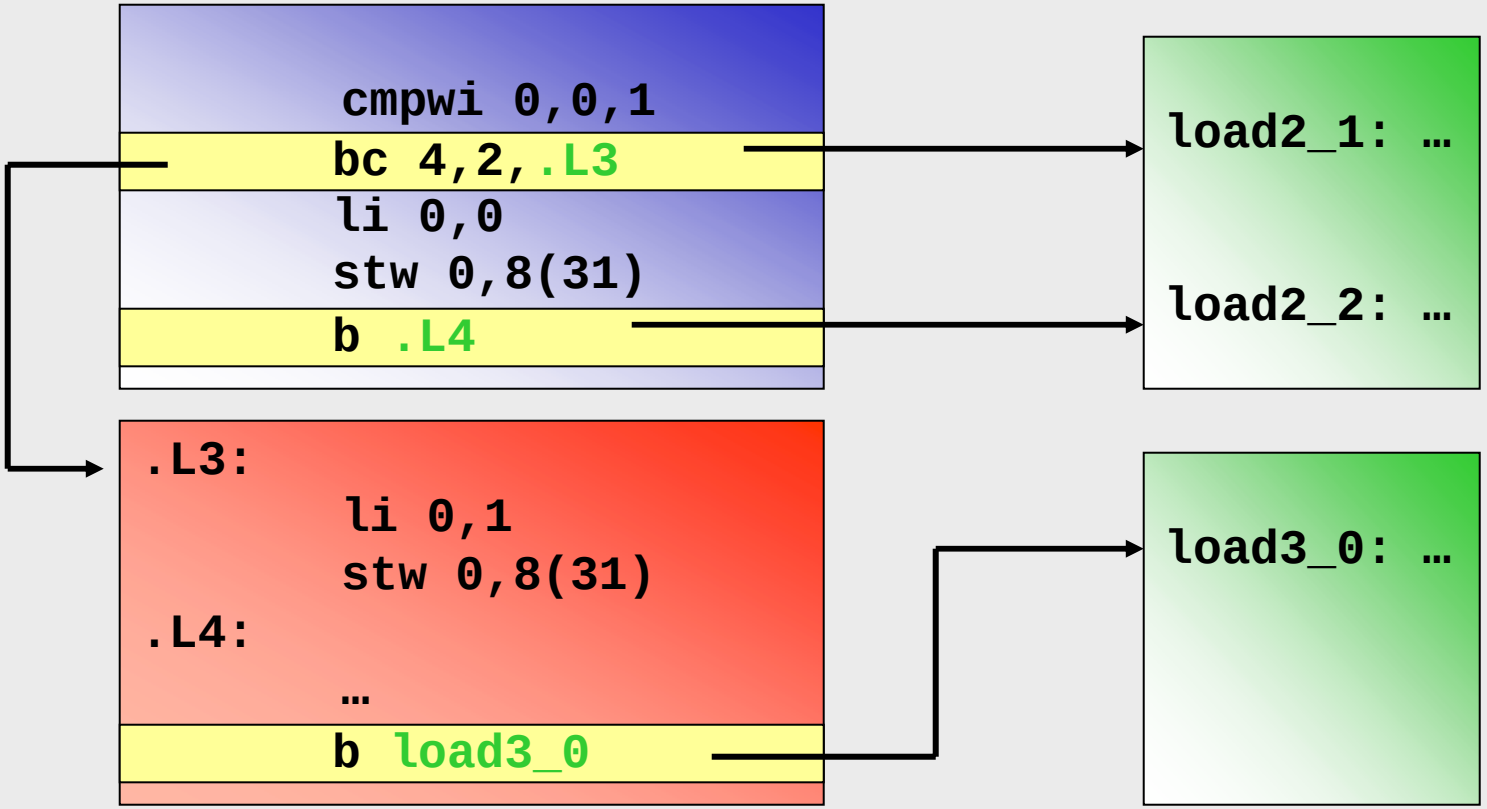
Block Streaming: Example (1)

```
if (i==1)
  i=0;
else
  i=1;
```

```
        cmpwi 0,0,1
        bc 4,2,.L3
        li 0,0
        stw 0,8(31)
        b .L4
```

```
.L3:
        li 0,1
        stw 0,8(31)
.L4:    …
```

Block Streaming: Example (2)



Performance Metrics

- Overhead
  - Transmission and memory overhead (12 bytes/off-block branch). However, if some blocks are not loaded, software streaming saves resources
  - Runtime overhead
- Application load time
- Application suspension time

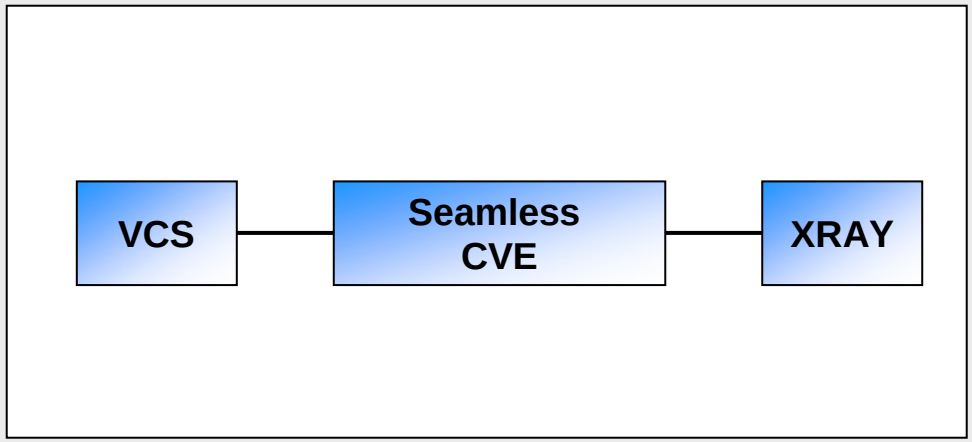
Performance Enhancement (1)

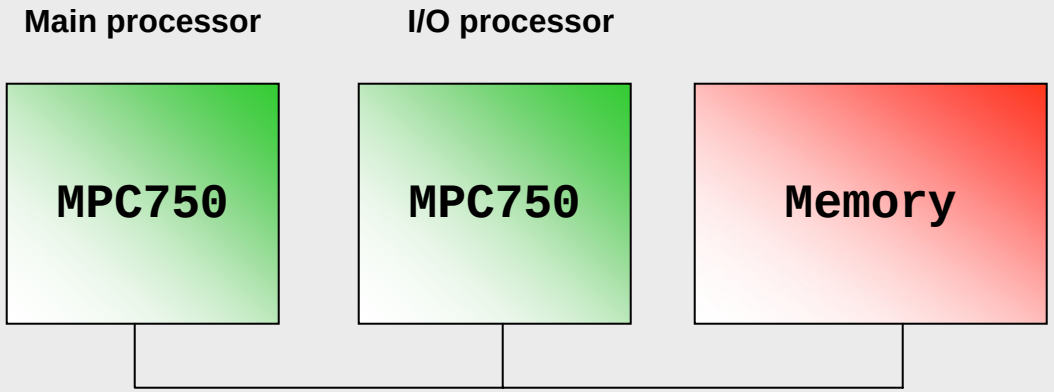
Background Streaming

Performance Enhancement (2)

On-demand Streaming

Simulation Environment





Softstream Tools

- Softstream Generator
  - Generates stream-enabled application from the original binary image
- Softstream Loader
  - Loads blocks into memory and invokes the application
- Softstream Linker
  - Modifies code at runtime

Simulation Scenario

- Adaptive autonomous robot exploration
- Impossible to write and load software for all possible environments
- The mission control needs to update the robot software over a 128Kbps link
- The new code is 10MB
- The robot must run the software to react to the new environment within 120 s

Simulation Results (1)

Block size (bytes)	Total # of blocks	Added code/block	Load time (s)
10M	1	0.0003%	655.36
5M	2	0.0007%	327.68
2M	5	0.0017%	131.07
1M	10	0.0034%	65.54
0.5M	20	0.0069%	32.77
100K	103	0.0352%	6.40
10K	1024	0.3516%	0.64
1K	10240	3.5156%	0.06
512	20480	7.0313%	0.03

Simulation Results (2)

- Sending the whole software takes over 10 minutes: the deadline is missed
- Using software streaming with the first blocks of size of 1MB, the new software can be executed within 66 seconds: the deadline is met
- The application load time improves by a factor of more than 10X

Conclusion

- Embedded software streaming allows an embedded device to start executing an application while the application is being transmitted.
- Our streaming method can lower application load time, bandwidth utilization and memory usage
- We verified our streaming method using a hard/software co-simulation platform

Future Work

- Branch prediction algorithm
- Software profiling
- APIs for controlling background streaming