Vernetzte Embedded Systems und Visualisierung mit Java

Dipl.-Biol. Rainer Becker

Fraunhofer-Institut für Solare Energiesysteme ISE
Heidenhofsstr. 2, 79110 Freiburg
Tel.: 0761/4588-5409, Fax.: 0761/4588-9000
Rainer.Becker@ise.fhg.de, http://www.ise.fhg.de

Stand der Technik und Entwicklungspotential bei modernen Heizsystemen

In den letzten Jahren hat es im Bereich der Heizungstechnik zahlreiche technische Neu- und Weiterentwicklungen gegeben. Moderne Brennwerttechnik soll Energie einsparen und die Umwelt weniger belasten. Eine der in der Zukunft sicher wichtigsten Energiequellen ist die Sonne. Thermische Solaranlagen werden heute in viele Heizsysteme integriert. Wärmepumpen und Lüftungssysteme kommen oftmals hinzu, wodurch der Komplexitätsgrad der Anlagen ständig zunimmt.

Heizsysteme werden heute aus einzeln geregelten Subsystemen zusammengestellt. Jedes dieser Subsysteme hat eine eigenständige Regelung, die für diese Hardwarebasis angepasst wurde und auch meist gut funktioniert. Probleme und Widersprüche gibt es erst beim Zusammenspiel der Subsysteme. Da keiner der Regler über die Zustände der anderen Teilsysteme der Anlage Bescheid weiß, wird immer davon ausgegangen, dass sich die anderen Teilregler so verhalten, dass die eigene Regelung sinnvoll ist. Diese Annahme stimmt jedoch meistens nicht und die einzelnen Teilsysteme sind so stark voneinander abhängig, dass sie sich deshalb auch sehr schnell gegenseitig negativ beeinflussen (Bsp.: In einem Heizsystem mit thermischer Solaranlage kann diese aus Mangel an kaltem Wasser nicht energieeffizient arbeiten, wenn der zentrale Warmwasserspeicher von der Gastherme komplett durchgeheizt wurde.)

Um die Kommunikation zwischen den Reglern für die verschiedenen Anlagenkomponenten zu ermöglichen, liegt es nahe, diese miteinander zu vernetzen. Dazu bietet sich die Verwendung von herkömmlicher Netzwerktechnik aus der EDV an. Mit der reinen Infrastruktur der Vernetzung ist das Problem jedoch noch nicht gelöst. Das Geflecht gegenseitiger Einflüsse muss so aufgelöst werden, dass sich jeder einzelne Regler in Abhängigkeit seines Subsystems von anderen Anlagenkomponenten sinnvoll verhält. Dazu ist zunächst einmal die Verfügbarkeit aller für eine energieeffiziente Regelung benötigten Informationen zu gewährleisten. Die einzelnen Komponenten der Anlage müssen ein ihnen allen gemeinsames,

standardisiertes Datenformat verwenden und es müssen Software-Schnittstellen vorhanden sein, um Daten untereinander auszutauschen. Ein vernetztes Regelungssystem kann in die Lage versetzt werden, mit einer zentralen Instanz zur Datensammlung und -auswertung das Nutzerverhalten selbständig zu beurteilen und sein Regelverhalten daran anzupassen. Auch durch Interaktion mit dem Nutzer, z.B. über einen programmierbaren Wohnungsregler, kann eine höhere Energieeffizienz erreicht werden (Während die Bewohner eines Gebäudes abwesend sind, kann der mit starken Wärmeverlusten verbundene Betrieb der Zirkulation für heißes Wasser eingestellt werden.). Durch Zusatzinformationen, wie z.B. Prognosen für die zu erwartenden Außentemperaturen und die Sonneneinstrahlung (z.B. aus dem Internet) kann das Regelverhalten zusätzlich verbessert werden. Wenn alle vernetzten Regelungskomponenten eines Heizsystems die wichtigen Kenndaten und Betriebsbedingungen ihrer Baugruppe/Verbraucher kennen, so kann daraus im Netzwerkverbund ein sinnvolles, energieeffizientes Regelungsverhalten errechnet und die Anlage entsprechend betrieben werden.

In Abb. 1 ist ein Beispiel für die Nutzung verschiedener Informationsquellen bei der Berechnung eines Regelverhaltens zu sehen. Zwei von drei Wohnungen melden Bedarf nach Heizwärme beim Schichtenspeicher, dem Speicher für heißes Wasser an. Dieser Speicher aktiviert daraufhin die beiden Wärmeerzeuger Gastherme und Solaranlage. Wetterdaten aus dem Internet sorgen bei entsprechender Einstrahlungsprognose für eine Aktivierung der thermischen Solaranlage. Da durch den Betrieb der Solaranlage ausreichende Wärmemengen zur Verfügung gestellt werden können, hemmt die Solaranlage den Betrieb der Gastherme.

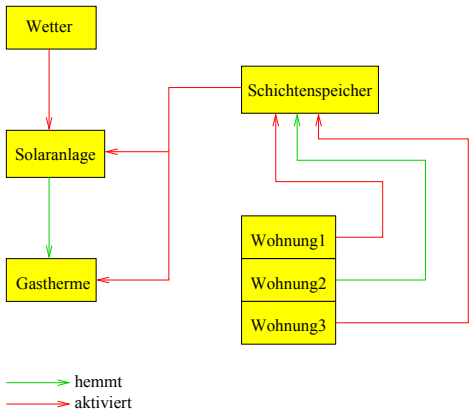

Abbildung 1: Regelungsoptimierung in einem Heizsystem durch vernetzte Informationsverarbeitung

Die Technische Basis für vernetzte Strukturen

Die Hardware für die in Firmennetzen und im Internet vielfach bewährte Vernetzung mittels Ethernet ist mittlerweile so preisgünstig geworden, dass sie auch im Bereich der Heizungstechnik sinnvoll anwendbar ist. TCP/IP bietet sich als Übertragungsprotokoll an, weil es für fast jede vernetzbare Hardware zur Verfügung steht. Als zentraler Rechner eines größeren Heizsystems für die Datenhaltung und zentrale Regelungsaufgaben erscheint ein sogenanntes *embedded system* geeignet (Bsp.: siehe Abb. 2). Dabei handelt es sich um einen netzwerkfähigen Kleinrechner. Er kann mit allen vernetzten Anlagenkomponenten kommunizieren, deren Verhalten überwachen, Daten über ihre Betriebszustände sammeln und bereitstellen sowie ihnen Anweisungen für die Regelung geben. Als Betriebssystem kann das aus der Open-Source-Szene stammende Unix-Derivat Linux zum Einsatz kommen, da es die benötigte Internet-Funktionalität standardmäßig beinhaltet, stabil läuft und kostenlos ist. Fertige embedded systems oder auch Prozessorboards werden durch die zahlreichen Consumer-Produkte immer preisgünstiger.

Abbildung 2: Ein am Fraunhofer-ISE entwickeltes embedded system mit x86-kompatiblem Prozessor, 8MB RAM und Netzwerkanschluss

Die einzelnen Anlagenkomponenten können mit preiswerten, netzwerktauglichen Mikrocontrollern bestückt werden (Bsp.: siehe Abb. 3). Sie können die konventionelle Regelungshardware ersetzen oder aber diese steuern. Die Controller haben zahlreiche Ein- und Ausgänge (analog, digital, RS232, RS485, Ethernet,...),

so dass damit die Messdatenerfassung, die Kommunikation mit dem Embedded-Linux-Rechner und die Regelung für die entsprechende Anlagenkomponente realisiert werden können. Derartige Mikrocontroller sind für ca. 70 Euro erhältlich.

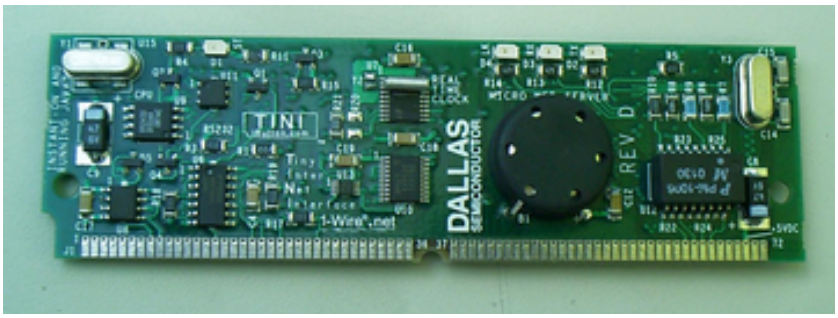

Abbildung 3: Ein netzwerkfähiger Mikrocontroller (TINI von Dallas Semiconductor) mit 1MB RAM und integrierter Java-VM

Die Netzwerktopologie

Innerhalb eines Heizsystems werden die Mikrocontroller der einzelnen Subsysteme der Anlage sowie der Embedded-Linux-Rechner sternförmig über einen Hub/Switch miteinander verbunden (Siehe Abb. 4). Jede Komponente kann mit jeder anderen Komponente über den Hub/Switch kommunizieren. Der Embedded-Linux-Rechner soll Informationen aus dem Internet laden und gleichzeitig Daten für Logging- und Visualisierungsfunktionen liefern. Die Mikrocontroller hingegen sollen keinen direkten Zugang zum Internet haben. Um das Netzwerk aus Embedded-Linux-Rechner und Mikrokontrollern vor unbefugtem Zugriff zu schützen, wird es durch eine Firewall zum Internet hin abgeschirmt. Der Zugang zum Internet erfolgt dann über einen ISDN/DSL-Router.

Sollen mehrere räumlich dicht verbundene Heizsysteme (Siehe Abb. 5) von verschiedenen Orten aus überwacht werden, so bietet sich eine Anbindung an das Internet über einen Router sowie ein zentraler Server (PC-Basis) für das Datenlogging sowie die Bereitstellung der Visualisiserung an. Die Heizsysteme sowie der zentrale Server sollten auch wieder durch Firewalls zum Internet hin abgeschirmt werden, da sonst Manipulationen an den Reglern der Anlagen und den Messdaten nicht auszuschließen sind. Durch den zentralen Server können die mit nur relativ leistungsschwacher Hardware ausgestatteten embedded systems entlastet werden. Auch komplexere Visualisierungs- und Auswertungsfunktionen sind

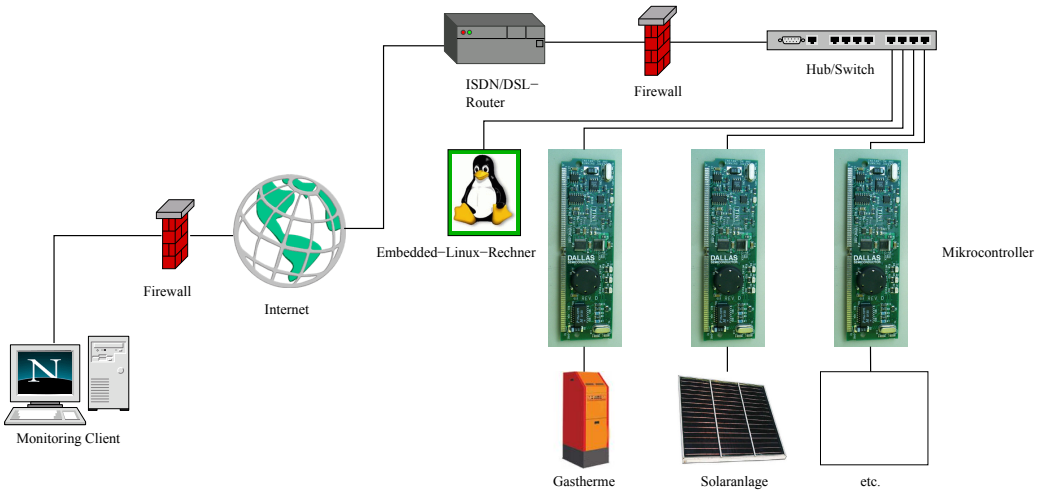

Abbildung 4: Vernetzung eines einzelnen Heizsystems. Sämtliche Server-Funktionen müssen vom Embedded Linux Rechner bereitgestellt werden.

mit einer solchen Infrastruktur besser möglich. Ein zentraler Server kann auch Plausibilitätsprüfungen der Messdaten durchführen und aufgrund der Auswertung im Fehlerfall Wartungspersonal per EMail oder SMS benachrichtigen. Auch eine Energieflußkontrolle ist damit möglich, die Rückschlüsse auf die Energieeffizienz der einzelnen Anlagenteile zulässt.

Visualisierung mit Java und XML-RPC

In Abb. 6 ist ein Screenshot von der Online-Visualisierung der Solaranlage des Fraunhofer ISE in Freiburg zu sehen. Die Visualisiserung wurde mit der Programmiersprache *Java* realisiert. Java geht auf ein Projekt aus dem Jahr 1991 zur Entwicklung einer einfachen und plattformunabhängigen Sprache für Geräte der Konsumelektronik zurück. 1996 wurden die Webbrowser der Firmen Netscape (Netscape) und Microsoft (Microsoft InternetExplorer) so gestaltet, dass sie Java-Programme aus dem Netz laden und im Browser-Fenster ausführen konnten. Diese Programme, sogenannte *Applets*, haben den Vorteil, dass sie lokal auf dem Rechner laufen, auf dem auch der Webbrowser ausgeführt wird. Der Server, von dem das Applet stammt, muss so keine Rechenleistung für die Ausführung des Applets zur Verfügung stellen. Will man Prozesse visualisieren, die von einem

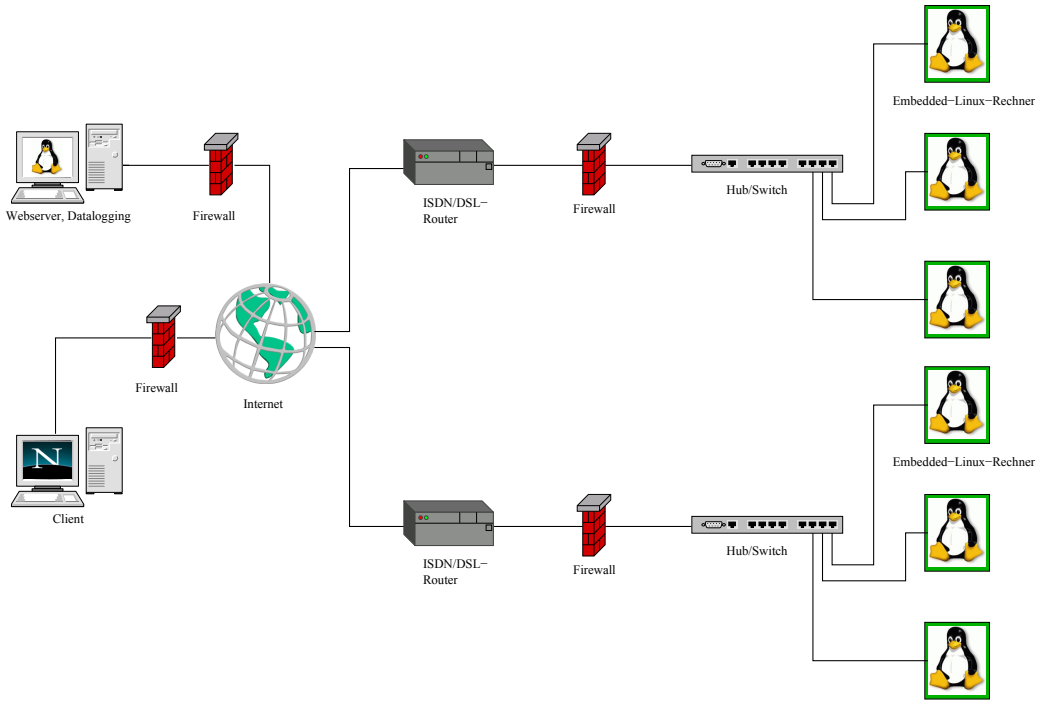

Abbildung 5: Mehrere Heizsysteme werden an das Internet angebunden. Ein Webserver hostet Java-Applets zur Visualisiserung der Anlagenzustände und loggt Daten über die Betriebsführung der Anlagen.

embedded system gesteuert werden, so kann dieses als Server fungieren, der das Applet an den Webbrowser des Clients überträgt und der außerdem die zur Visualisierung benötigten Daten zur Verfügung stellt. Die aufwendige grafische Darstellung für die Visualisierung übernimmt komplett der Client-Rechner, so dass die Ressourcen des Embedded System geschont werden. Im Gegensatz zu Visualisierungsprozessen, bei denen der Server die grafische Aufbereitung der Daten durchführt und in Form von statischen Webseiten, die in festen Zeitrastern neu generiert werden, an den Client überträgt, kann ein Java-Applet selbst bestimmen, wie oft neue Daten angefordert und visualisiert werden. Mit der zweiten Generation der Programmiersprache Java stehen neue grafische Möglichkeiten wie z.B. Antialiasing (Kantenglättung) und DoubleBuffering zur Verfügung, die auch anspruchsvollere Visualisierungsprogramme ermöglichen. Bei sehr komplexen Visualisierungen oder zur Visualisierung von mehreren Anlagen empfiehlt sich die Nutzung eines leistungsstarken zentralen Servers, um die embeddes systems der

Anlagen zu entlasten. Der Server kann jeweils einmal die von den möglicherweise vielen Clients benötigten Daten holen und diesen dann zur Verfügung stellen.

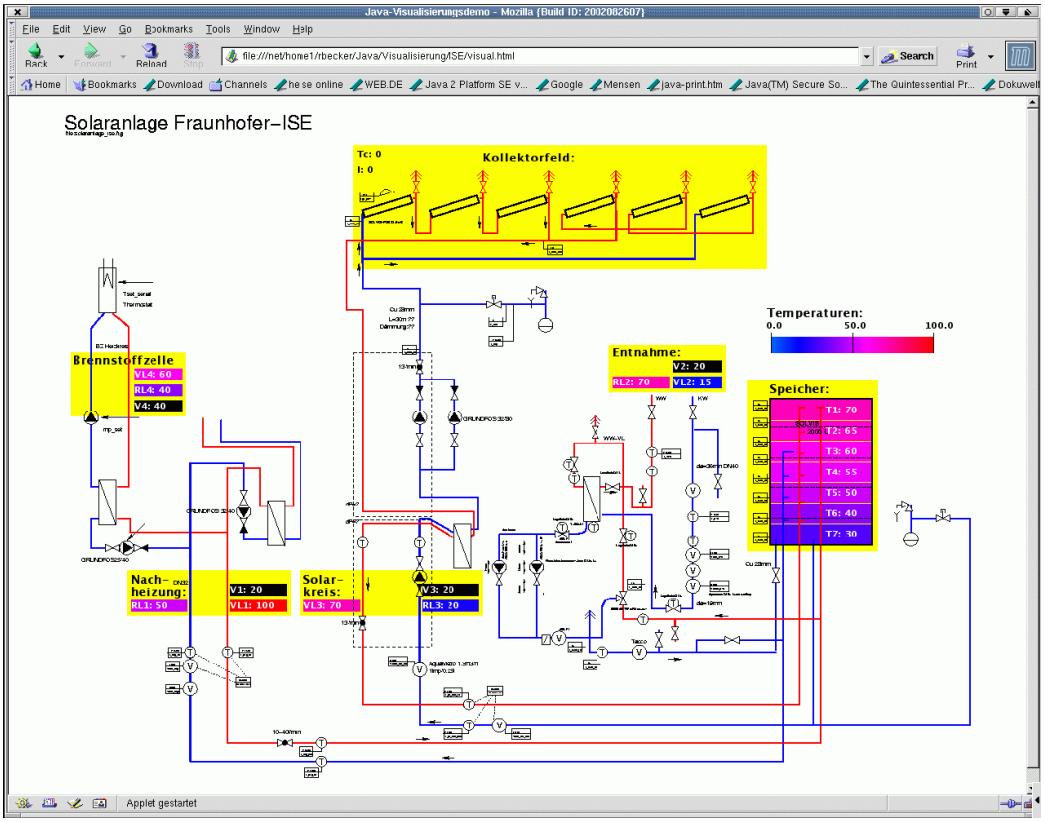

Abbildung 6: Visualisierung der Solaranlage des Fraunhofer ISE in Freiburg im Webbrowser Mozilla

Bleibt die Frage zu klären, in welcher Form die Datenübertragung zwischen einem Java-Applet und dem Server stattfinden soll, der die zu visualisierenden Daten liefert. Die programmiertechnisch einfachste und die Ressourcen des Servers am wenigsten belastende Variante ist sicherlich die Übermittlung von einfachen Strings ("Zeichenketten"). Das Embedded System "lauscht" auf einem Server-Socket auf eingehende Verbindungen. Wird von einem Client eine solche Verbindung aufgebaut und eine Anfrage in Form eines Strings gestellt, so wird der String ausgewertet und eine entsprechende Antwort generiert, die auch als String an den Client übergeben wird. Danach baut der Client die Verbindung wieder ab. Die Kommunikation über Sockets funktioniert jedoch nur dann, wenn die ent-

sprechenden Ports im Netzwerk freigeschaltet sind. Im Zuge des zunehmenden Bewusstseins für Datensicherheit werden immer mehr Rechner durch sogenannte Firewalls abgeschottet, die nur Kommunikation auf standardisierten Ports für bestimmte Internet-Dienste zulassen (Bsp.: Port 80 für WWW, Port 22 für SSH). Damit ist eine Kommunikation über Sockets auf anderen Ports unmöglich.

Eine etwas aufwändigere, aber dafür auch überall funktionierende Variante der Kommunikation zwischen einem Java-Applet im Webbrowser und einem embeddes System ist die Nutzung von *XML-RPC*s. XML steht für Extensible Markup Language, RPC für Remote Procedure Call. XML-RPC-Datentransfers werden auf dem gleichen Port (Port 80) abgewickelt, wie Anfragen nach Webseiten. Es wird auch das gleiche Übertragungsprotokoll (HTTP=HyperText Transfer Protocol) genutzt, so dass die Kommunikation selbst über sogenannte Proxy-Server funktioniert, die in vielen Firmennetzen nicht umgangen werden können.

In Abb. 7 ist beispielhaft der Ablauf der Kommunikation zwischen einem Client, einem Server und zwei embedded systems dargestellt. Der Webbrowser (im Bespiel *Netscape*) des Clients fragt zunächst eine Webseite beim Webserver (im Beispiel *Apache*) an. Eingebettet in die Webseite ist das Visualisierungsapplet, welches vom Server an den Client übertragen und auf diesem ausgeführt wird. Das Java-Applet fragt mittels XML-RPC beim Webserver Daten von den zu visualisiserenden Prozessen an. Der Webserver leitet die Anfrage an ein sogenanntes *CGI* (=Common Gateway Interface), ein kleines Programm weiter, welches entweder auch wieder über XML-RPCs, oder aber in der einfacheren Variante über Sockets von den embedded systems die benötigten Daten anfordert. Liegen diese vor, so werden sie vom Webserver an das Java Applet weitergegeben und dort visualisiert.

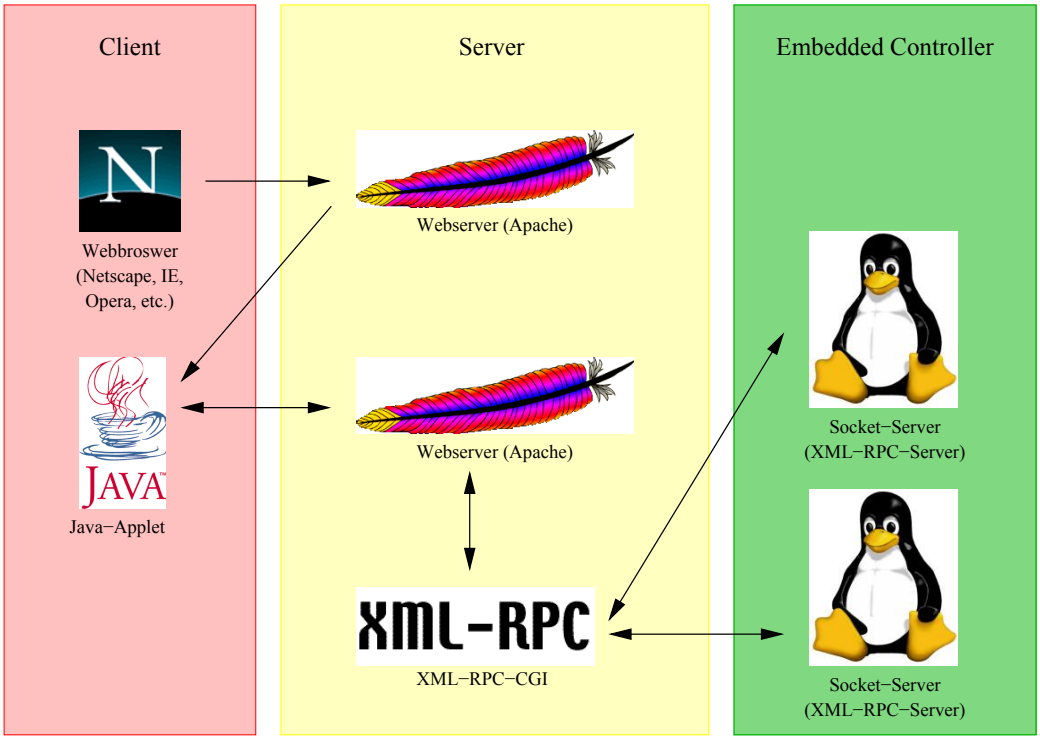

Abbildung 7: Ablauf der Kommunikation zwischen Visualisierungsrechner (Client), zentralem Server und embedded systems