Eine dazugehörige Antwort kann dann das folgende Format haben:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 ...
</soap:Envelope>
```

16.2.4 UDDI

»Universal Description, Discovery, and Integration« war ursprünglich Teil einer noch längeren Bezeichnung: »Universal Description, Discovery, and Integration Business Registry« oder etwas kürzer »UDDI Business Registry« oder einfach »UBR«. Dahinter steckt die ursprüngliche Absicht, mit Web-Services einen Marktplatz für Services zu schaffen. Dabei sollte es drei fundamentale Rollen geben: Anbieter, die Services bereitstellen, Nutzer, die Services benötigen, und Vermittler/Broker, die Anbieter und Nutzer zusammenbringen. Dieser Ansatz führte zu dem in Abbildung 16-2 dargestellten bekannten Dreieck, das oft gezeigt wird, wenn Web-Services erklärt oder eingeführt werden.

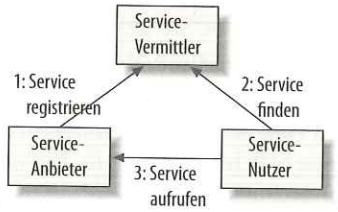

Abbildung 16-2: Anbieter, Nutzer und Vermittler von Services

Das UBR sollte dabei die Rolle des zentralen Vermittlers für einen weltweiten Marktplatz von Web-Services übernehmen. Dies ist mit so etwas wie weltweiten gelben Seiten für IT-Dienstleistungen zu vergleichen. Anbieter von IT-Dienstleistungen sollten ihre Web-Services dort einfach registrieren und Nutzer diese dann anhand bestimmter Kriterien dort finden können. UDDI war das Protokoll, das dabei zum Registrieren und Finden von Web-Services dienen sollte.

Es stellte sich allerdings heraus, dass die Idee des UBR nicht angenommen wurde, weshalb die UBRs als weltweites Zentralregister im Januar 2006 abgeschaltet wurden. Die offizielle Begründung lautete wie folgt (siehe [*UBR*]):

> *IBM, Microsoft und SAP haben den Status der UDDI Business Registry neu bewertet und sind zu dem Schluss gekommen, dass die Ziele des Projekts erreicht wurden. Aus diesem Grund wird die UDDI Business Registry am 12. Januar 2006 abgeschaltet.*

Dies war im Grunde ein bemerkenswertes Beispiel für eine Geschichtsverfälschung. Anstatt zu erklären, dass die Idee der weltweiten gelben Seiten für Web-Services (noch) nicht angenommen worden war, definierten die Betreiber der UBRs kurzerhand das ursprüngliche Ziel um und behaupteten, es wäre nur darum gegangen, einen Prototyp für ein derartiges Zentralregister zu betreiben, um den dazugehörigen Standard zu entwickeln.

Der Fehlschlag des UBR bedeutet natürlich nicht, dass UDDI selbst ein Fehlschlag war, aber es gibt in der Praxis Zweifel, wie sinnvoll UDDI wirklich ist. Mag sein, dass die Zeit einfach noch nicht reif war und einfach nur die richtigen Services fehlten, aber es lag auch an nicht gemachten »Hausaufgaben«. In [*NewcomerLomow05*] heißt es dazu:

> *UDDI erreichte sein Ziel, das Zentralregister für alle Metadaten zu Web-Services zu werden, allerdings aus verschiedenen Gründen nicht. Einer davon war die Schwierigkeit, sinnvolle Kategorien für die dazugehörigen Informationen zu finden. Im Ergebnis stellte sich UDDI für die Mehrzahl an Web-Services-Interaktionen über das Web als wenig hilfreich heraus.*

Web-Services benötigen allerdings aus technischen Gründen oftmals einen Vermittler, da es sich bei den Service-Aufrufen inhärent um Punkt-zu-Punkt-Verbindungen (englisch: point-to-point connections) handelt und man mit Hilfe des Vermittlers vermeiden kann, dass Empfangsadressen bei den Web-Services-Nutzern fest kodiert werden (siehe Abschnitt 5.3.1 auf Seite 67). Dafür fanden sich allerdings auch andere Lösungen. Ich zitiere noch einmal [*NewcomerLomow05*]:

> *Generell sieht es so aus, dass einige Firmen zwar UDDI verwenden, andere erweitern dagegen existierende Lösungen für Zentralregister wie LDAP, JNDI (»Java Naming and Directory Interface«), relationale Datenbanken oder den CORBA-Trader. Und in begrenztem Umfang kann auch eine »ebXML-Registry« hilfreich sein.*

Inzwischen gibt es mehrere UDDI-Versionen und andere Web-Services-Standards, die sich teilweise überlappen. Nach [*NewcomerLomow05*]:

> *sind diese Überlappungen zwischen den Spezifikationen ein weiteres typisches Kennzeichen für den Grund, dass es bei der Standardisierung von Web-Services an Führung und Architektur mangelt.*

Alles in allem ist die Situation in Bezug auf UDDI ein ziemliches Chaos, und erst die Zeit wird zeigen, wie sich die Dinge weiterentwickeln.

Wie in Abschnitt 5.3.3 auf Seite 71 erläutert, legt ein protokollgetriebener ESB nur ein Protokoll fest, an das sich alle Beteiligten halten müssen. Ansonsten sind die einzelnen Anbieter und Nutzer aber zunächst einmal autark. Letzteres ist auch der Vorteil dieses Ansatzes. Es müssen keine Bibliotheken ausgetauscht werden, sondern die Beteiligten müssen nur sicherstellen, dass sie sich an das Protokoll halten. Unglücklicherweise ist Interoperabilität beim Web-Services-Protokoll aber nach wie vor ein Thema. Infolgedessen kann man nicht unbedingt davon ausgehen, dass wenn System A mit System B und System B mit System C kommunizieren kann, dann auch System A direkt mit System C kommunizieren kann. Mit anderen Worten reicht es für Interoperabilität nicht aus, dass jedes System am ESB hängt. Um sicherzugehen muss man jede einzelne Kommunikation zwischen zwei Systemen verifizieren. Das hebt den eigentlichen Vorteil eines ESB zumindest teilweise wieder auf.

Die Tatsache, dass es sich bei Service-Aufrufen inhärent um Punkt-zu-Punkt-Verbindungen handelt, bedeutet, dass zunächst jeder Nutzer beim Aufruf wissen muss, wo der Service physikalisch angeboten wird (siehe auch Abschnitt 5.3.1 auf Seite 67). Um für Load-Balancing- oder Failover-Zwecke erst zur Laufzeit entscheiden zu können, wo ein Service aufgerufen wird, muss einiges an Aufwand getrieben werden. Man kann UDDI einsetzen, um vor dem eigentlichen Aufruf herauszufinden, wo ein Service angeboten wird, oder man kann innerhalb des ESBs Mechanismen einbauen, die eine Indirektion ermöglichen. Alles das muss man aber explizit veranlassen bzw. jeder Nutzer ggf. ausprogrammieren.

Am besten ist es, wenn beide Probleme mit entsprechenden Konzepten so gelöst werden, dass sich die einzelnen Anbieter und Nutzer auf die fachlichen Aspekte konzentrieren können. Dazu kann man den Web-Services-Ansatz zum Beispiel zu einem API-Ansatz ausbauen, was bedeutet, dass von zentralen Teams auch die Tools bereitgestellt und gepflegt werden, die das Web-Services-Protokoll auf plattformspezifische APIs abbilden.

Ein anderer Ansatz, der mit dem vorherigen kombiniert werden kann, besteht darin, die in Abschnitt 5.3.2 auf Seite 69 angesprochenen Interceptoren bereitzustellen. Diese kapseln den eigentlichen ESB nach außen über die angebotenen Web-Services-Schnittstellen (also ein SOAP-Protokoll). Intern lassen sich dann aber beliebige Mechanismen etwa zum Protokollieren oder zum intelligenten Routen zur Laufzeit einbauen. Ein derartiger Ansatz sieht dann zum Beispiel so wie in Abbildung 16-4 aus.

Konkret kann das so aussehen, dass der ESB intern aus ein oder mehreren Application-Servern besteht, die als zentrale »Hub-and-Spoke«-Server fungieren. Wenn es sich um mehrere Server handelt, können diese mit Protokollen wie JMS, MQ oder JBI verbunden werden. Jeder Application-Server bietet nach außen zu einzelnen Anbietern und/oder Nutzern lokale Web-Services-Schnittstellen, die das SOAP-Protokoll verwenden und mit WSDL-Dateien beschrieben werden. Dabei kann ein und derselbe Service den einzelnen Nutzern unter verschiedenen Adressen angeboten werden (nach außen teilt man ja nur die Adresse des jeweiligen Interceptors mit; wohin die Aufrufe intern geleitet werden, ist für die Nutzer irrelevant). Bei einem Kunden haben wir derartige Systeme zum Beispiel so gebaut, dass pro Land ein zentraler Server existierte, sodass innerhalb des ESB Nachrichten nur dann über verschiedene Server liefen, wenn es sich um länderübergreifende Service-Aufrufe handelte.

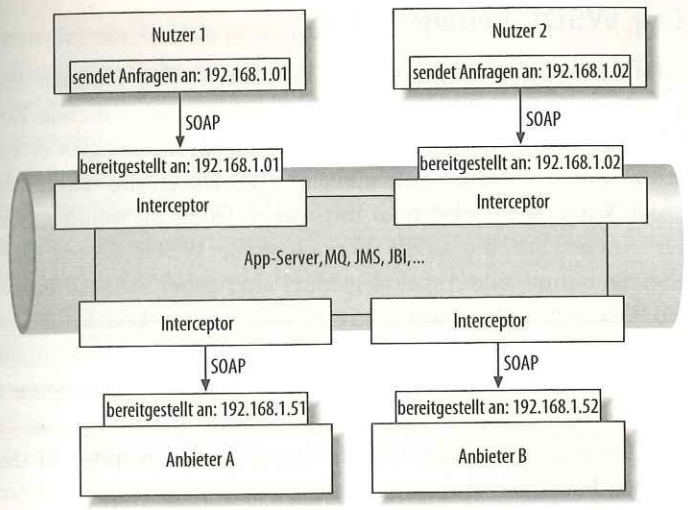

Abbildung 16-4: Ein Web-Services-ESB, der intern andere Protokolle verwendet

Mit einem derartigen Ansatz kann man auch die inhärenten Probleme in Bezug auf Sicherheit (siehe Kapitel 14) und Zuverlässigkeit (siehe Kapitel 10) in den Griff bekommen. Jeder Anbieter und Nutzer kann mit seinem Interceptor über eine HTTPS-Verbindung kommunizieren (wozu er ggf. ein entsprechendes Zertifikat benötigt), und innerhalb des ESB werden Protokolle verwendet, die im Gegensatz zu HTTP eine Auslieferung von Nachrichten garantieren.

Man beachte, dass man mit einer derartigen Ausbaustufe eines ESB bei Web-Services aber nicht von vornherein beginnen muss (und auch nicht sollte). Wenn man mit Web-Services beginnt, reicht es völlig aus, wenn es eine Indirektion zwischen Anbieter und Nutzer gibt, sodass Nutzer die physikalischen Adressen für die konkreten Aufrufe nicht direkt von den Anbietern beziehen. Stattdessen werden Stellen oder Mechanismen definiert, die die Adressen zur Startzeit liefern. Im ersten Wurf sind diese Startadressen dann in der Tat die tatsächlichen Adressen der Anbieter. Später, wenn man den ESB dann ausbaut, ersetzt man diese Adressen durch die Adressen der Interceptoren, die die Aufrufe dann weiterleiten. Ist das einmal der Fall, kann man innerhalb des ESB immer bessere höherwertige Dienstleistungen einbauen (siehe Abschnitt 5.4 auf Seite 74). Durch einen derartigen inkrementellen Ansatz kann man vor allem sicherstellen, dass man aus praktischen Erfahrungen lernt, was notwendig und angemessen ist.