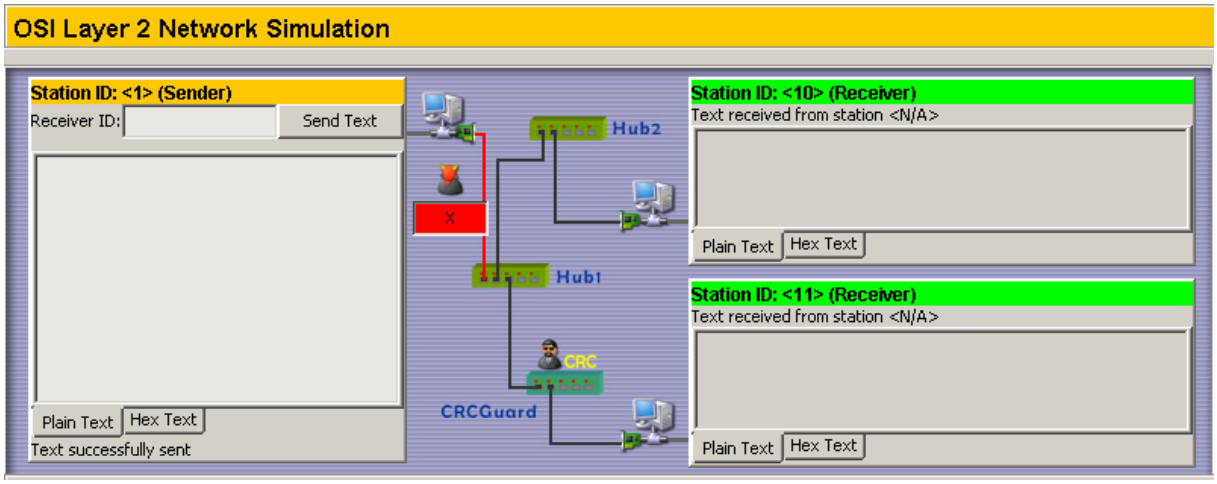
Ü Netzwerke und verteilte Systeme	Übung #3	WS 2007/2008
Name:	Matr-Nr:	
Abgabe: 30.10.2007		Gruppe:

Einführende Informationen zur Übung

Für die folgenden Übungsaufgaben wird ein Java-Framework verwendet, mit welchem eine konkrete Netzwerk-Infrastruktur nachgebildet wird. Das Framework selbst ist nicht an spezielle Protokolle, Geräte oder ähnliches gebunden und ermöglicht die Simulation von Datenübertragung mittels einer ISO/OSI 7 Schichten-Infrastruktur. Auf jeder Schicht können eine Reihe so genannter „Filter" eingehängt werden, die als Entities auf den Schichten dienen. In dieser Übung liegt der Schwerpunkt auf Layer 2, also dem Data-Link-Layer.

Vorgegebenes Szenario

In dem vorgegebenen Übungs-Szenario existieren drei Rechner, welche mittels diversen Geräten in einem Broadcast-Netz miteinander verbunden sind. In dem Szenario existieren zwei (Layer 1) Hubs, mit Namen „Hub 1" und „Hub 2". Es wurde ein drittes Gerät konstruiert, welches im Szenario „CRCGuard" genannt wird. Die Rechner (stations) haben eine eindeutige Layer 2 Identifikationsnummer. Der Rechner links, auf welchem auf Layer 7 die „Sendeapplikation" läuft, hat die ID 1; mittels dieser Applikation können gezielt Nachrichten an andere Rechner versendet werden. Auf den beiden anderen Stationen läuft jeweils eine Empfangsapplikation, welche Text anzeigt, sofern dieser bis zur Applikation durchgereicht wird. Die IDs dieser Rechner sind 10 bzw. 11.
Nun ergibt es sich, dass Rechner 1 mittels fehlerhaftem Kabel mit dem Hub 1 verbunden ist, und sich auf dem weiteren Weg Fehler einschleichen können. In der konkreten Implementierung kippt immer genau 1 Bit (das Fehlverhalten des Kabels kann zu Testzwecken mit einem Klick auf den Button „X" deaktiviert werden).



EasyL2

Als Übertragungsprotokoll auf Schicht 2 wurde „*EasyL2*" spezifiziert. Dieses sehr einfache Protokoll enthält die typischen Datenfelder, die auf Layer 2 ausgetauscht werden. Der Aufbau ist wie folgt:

Datenfeld	Größe	Zusätzliche Information
Präambel	1 byte	„Magic number“, muss immer 0xFE sein
Empfänger	2 byte	MSB zuerst übertragen
Sender	2 byte	MSB zuerst übertragen
Größe	1 byte	Größe des gesamten Frames
Daten	0-100 byte	Nutzdaten, max. 100 byte
CRC	4 byte	Prüfsumme über Frame

EasyL2 spezifiziert KEINE Wiederholungsanforderung, wenn beispielsweise vom Netzwerk-Stack erkannt wird, dass der Frame fehlerhaft ist. Derartige Funktionalität muss von darüber liegenden Schichten implementiert werden. D. h., wenn beispielsweise die Länge des Paketes oder CRC nicht übereinstimmt, wird der Frame einfach auf Layer 2 vernichtet und nie in Richtung Layer 3 weitergereicht. Höhere Schichten könnten dies beispielsweise mittels Timeouts erkennen und müssten entsprechend reagieren (nicht zu implementieren).

Was ist ein CRCGuard?

Als CRCGuard wurde in diesem Beispiel ein Gerät bezeichnet, welches auf Layer 2 Pakete inspiziert und genau dann verwirft, wenn der CRC im Paket fehlerhaft ist. Ansonsten wird nichts geprüft.
Im Beispiel dürften also fehlerhafte Pakete den Rechner 11 gar nicht erreichen, da diese vom vorgeschalteten CRCGuard schon ausgefiltert werden. Zu Rechner 10 können schon fehlerhafte Pakete vordringen, da diese von Hub 2 nicht ausgefiltert werden können. Rechner 10 filtert diese im eigenen Netzerk-Stack aus.
Ein CRCGuard kann also als „Switch im Flooding-Mode“ gesehen werden (die eigentliche Switching Funktionalität wird *nicht* bereitgestellt).

Simulation des Netzwerk-Stacks

Ein wesentlicher Bestandteil einer Netzwerk-Simulation ist der virtuelle Netzwerk-Stack von Rechnern. Dieser besteht aus zwei eigenständigen Teilen: einem Stack für ankommende (*IncomingNetworkStack*) und für auszusendende (*OutgoingNetworkStack*) Daten.

IncomingNetworkStack

Vom Kabel (Wire) ankommende Pakete werden beim IncomingNetworkStack eines Rechners angeliefert. Die Verarbeitung beginnt bei Layer 1 (physisches Entgegennehmen von der Leitung) und endet bei Layer 6. Hat ein Paket Layer 6 (erfolgreich) überwunden, wird es an die Applikation ausgeliefert. In jeder Schicht können so genannte *DecodingNetworkFilter* installiert werden. Dabei handelt es sich um Filter, welche eine spezielle, protokollspezifische Aufgabe erledigen. Eine Aufgabe für einen Filter auf Layer 2 wäre z. B., nur Pakete durchzureichen, die wirklich auch an den jeweiligen Rechner adressiert sind. Weiters muss mittels Filter auf Layer 2 der EasyL2 Frame aufgelöst werden. An Layer 3 dürfen nur Layer 3 Nutzdaten weitergereicht werden, da EasyL2 für die oberen Schichten transparent ist.

OutgoingNetworkStack

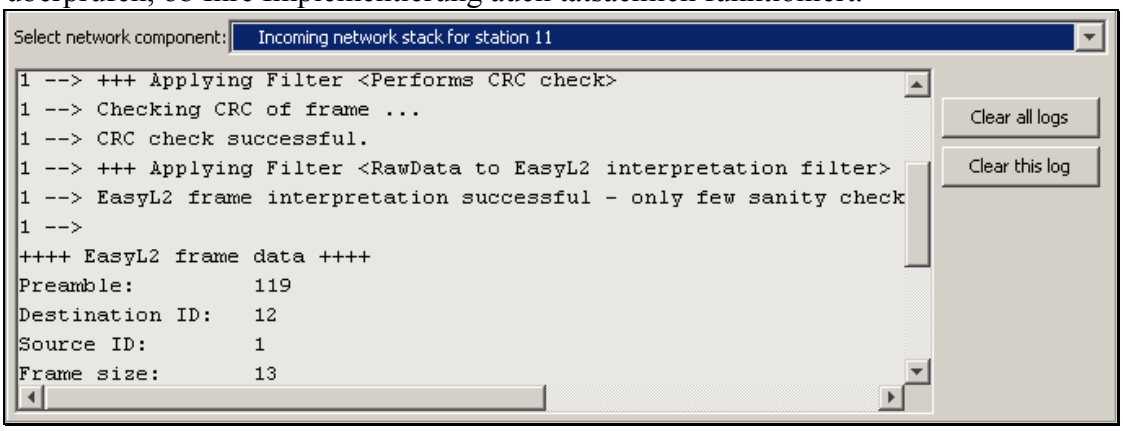
Möchte eine Applikation Daten an einen anderen Rechner versenden, so müssen diese den OutgoingNetworkStack des Rechners passieren. Die Daten werden sukzessive von einer oberen in eine untere Schicht weitergereicht, bis dass sie schlussendlich über das Kabel weitertransportiert werden. Dazwischen werden Verarbeitungsschritte der jeweiligen Schicht vorgenommen, welche in *EncodingNetworkFilter* gekapselt sind. In Layer 2 z. B. müssen die Daten u. a. mit dem EasyL2 Protokoll ummantelt werden.

Übungsmodalität

- Download des Frameworks und der Dokumentation von der Übungshomepage
- Abgabe des Programmcodes (nur geänderte Klassen im Quellcode) und der Ausarbeitung zusätzlich via E-Mail an
  `uebung3.netzwerke@fim.uni-linz.ac.at`
  (Deadline: 30.10.2007, 15:00 Uhr)
  Wird die Übung bis zu diesem Zeitpunkt nicht via Email abgegeben, können die Programmieraufgaben nicht gewertet werden.
- Alle abzugebenden Dateien müssen in eine ZIP-Datei zusammengefasst werden, welche mit Ihrer Matrikelnummer benannt wird (z. B. 0455100.zip).
- Im Subject Ihrer Email muss ebenfalls Ihre Matrikelnummer enthalten sein.
- Fragen und Probleme bzgl. der Übung? Mail an putzinger@fim.uni-linz.ac.at

Tipps und Hinweise

- Zu startende Klasse:
  `at.jku.fim.datalinksimulation.scenario.gui.SimulationApplication`
- Die einzelnen Komponenten und Stacks führen Ihr eigenes Log. Sie sind ein erster Anlaufpunkt für die Fehlersuche. Loggen Sie in den Übungsbeispielen ebenfalls alle wichtigen Aktionen, Zustände und Ergebnisse mit. In den Logs können Sie überprüfen, ob Ihre Implementierung auch tatsächlich funktioniert.



- Da Java keinen Datentyp unterstützt, dessen Wertebereich von 0-255 reicht, wird der Typ „int“ anstelle von „byte“ verwendet. Der Wertebereich ist aber trotzdem von 0 bis 255 beschränkt. D. h., für eine 16bit Zahl werden 2 int benötigt (dies ist für die Kommunikation im Framework selbst relevant, nicht jedoch für Ihre konkreten Aufgabenstellungen)
- Sehen Sie regelmäßig auf der Übungshomepage nach, ob es neue Hinweise oder Versionen des Frameworks gibt.
- Der Programmieraufwand dieser Übung ist nicht groß. Rechnen Sie aber etwas Einarbeitungszeit für das Framework mit ein.

Beispiel 3.1 (2 Kreuzerl – alle oder keine)

In der Implementierung des EasyL2 Protokolls fehlt die Generierung (beim Sender) bzw. die Überprüfung (beim Empfänger) des CRCs. Diese Funktionalität muss in der Klasse `at.jku.fim.datalinksimulation.utils.CRCUtils` ergänzt werden.

Implementieren Sie die beiden fehlenden Methoden in der genannten Klasse und verwenden Sie hierbei folgendes Generatorpolynom:

$x^{32} + x^{26} + x^{23} + x^{21} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

Implementieren sie die Berechnung des CRCs mittels Durchführung einer *Polynomdivision* (also nicht mittels simuliertem Schieberegister oder Lookup-Table).

Wie lautet der berechnete CRC für ein gültiges EasyL2-Paket, wenn als Nutzdaten folgende Zeichen übertragen werden: `EASYL2`

Beispiel 3.2

Mittels der Stations-ID „128“ im Empfänger-Teil eines EasyL2-Frames soll ab sofort ein „Broadcast“ durchgeführt werden können. Lokalisieren Sie die dafür verantwortliche Stelle im Code, und ändern Sie diese entsprechend ab. Deaktivieren Sie zum Testen die Leitungsstörung.

Beispiel 3.3

Berechnen Sie den CRC Wert für folgende Parameter:

Generator-Polynom: $x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^{8} + x^{6} + x^{5} + x^{2} + 1$

Daten: 0xFACE

D.h. berechnen Sie den folgenden Ausdruck:

$0xFACE0000 / (x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^{8} + x^{6} + x^{5} + x^{2} + 1)$

Geben Sie insbesondere nicht nur den CRC Wert (=Rest der Division) an, sondern auch das Ergebnis der Division!

BONUS (2 Bonus-Kreuzerl)

Wegen des großen Erfolgs von EasyL2 soll das Protokoll auf mehrere physikalische Übertragungsmedien erweitert werden. Dazu wird *EasyL2v2* spezifiziert, welches denselben Protokollaufbau wie EasyL2 hat. Zusätzlich, nach der Bildung des Frames, wird noch die Rahmenmarkierung `01111110` am Beginn und am Ende des Paketes hinzugefügt. Dazu muss in den Rahmen-Daten Bit-Stuffing durchgeführt werden – nach 5 gesetzten Bits wird also jeweils eine 0 gestopft. Die Netzwerk-Stacks der Geräte und Rechner sind bereits auf EasyL2v2 vorbereitet. Sie müssen jedoch noch einen Software- bzw. Firmware-Upgrade durchführen, indem Sie die eigentlichen Worker-Methoden implementieren.

1. Es sind bereits Kodier- und Dekodier-Filter an den richtigen Stellen in den Netzwerk-Stacks installiert. Diese rufen Methoden in der Klasse `at.jku.fim.datalinksimulation.layer2.protocols.EasyL2v2Utils` auf. Implementieren Sie die beiden Methoden in dieser Klasse.
2. Da jetzt auf Bitebene operiert wird, werden die Daten vor Übergabe an die genannte Klasse in eine BitSet-Struktur konvertiert. Mit frame.get(8) bekommen Sie beispielsweise das niederwertigste Bit des vorletzten Bytes im Frame, mit frame.get(32) das erste (niederwertigste) Bit des 5. letzten Bytes im Frame (entspricht in EasyL2 dem letzten Daten-Byte).
3. Geben Sie die Log-Ausgaben des Outgoing Network Stacks von Station 1 sowie die Log-Ausgaben des Incoming Network Stacks von CRCGuard mit ab. Die Magic-Number 0xFE (dezimal „254“) hat 7 Bits in Folge gesetzt und muss deshalb im Beispiel wie folgt codiert werden:

   11111110 → 11111011 | 0