Behandlung von Netzwerk- und Sicherheitsaspekten in einem Werkzeug zur verteilten Paarprogrammierung

Sandor Szücs

Institut Mathematik und Informatik - Freie Universität Berlin

14. Mai 2009

Agenda

Softwaretechnische Betrachtung

Paarprogrammierung

- zwei Entwickler ein Rechner
- Vorteile: höhere Disziplin, Wissenstransfer, weniger Defekte, ..

Side-by-Side-Programmierung

- zwei Entwickler zwei Rechner mit Möglichkeit auf PP zu wechseln
- Vorteile: weniger Overhead als PP [1]

[1] Nawrocki et al, Software Process Improvement, volume 3792 of Lecture Notes in Computer Science, pages 28–38. Springer, 2005.

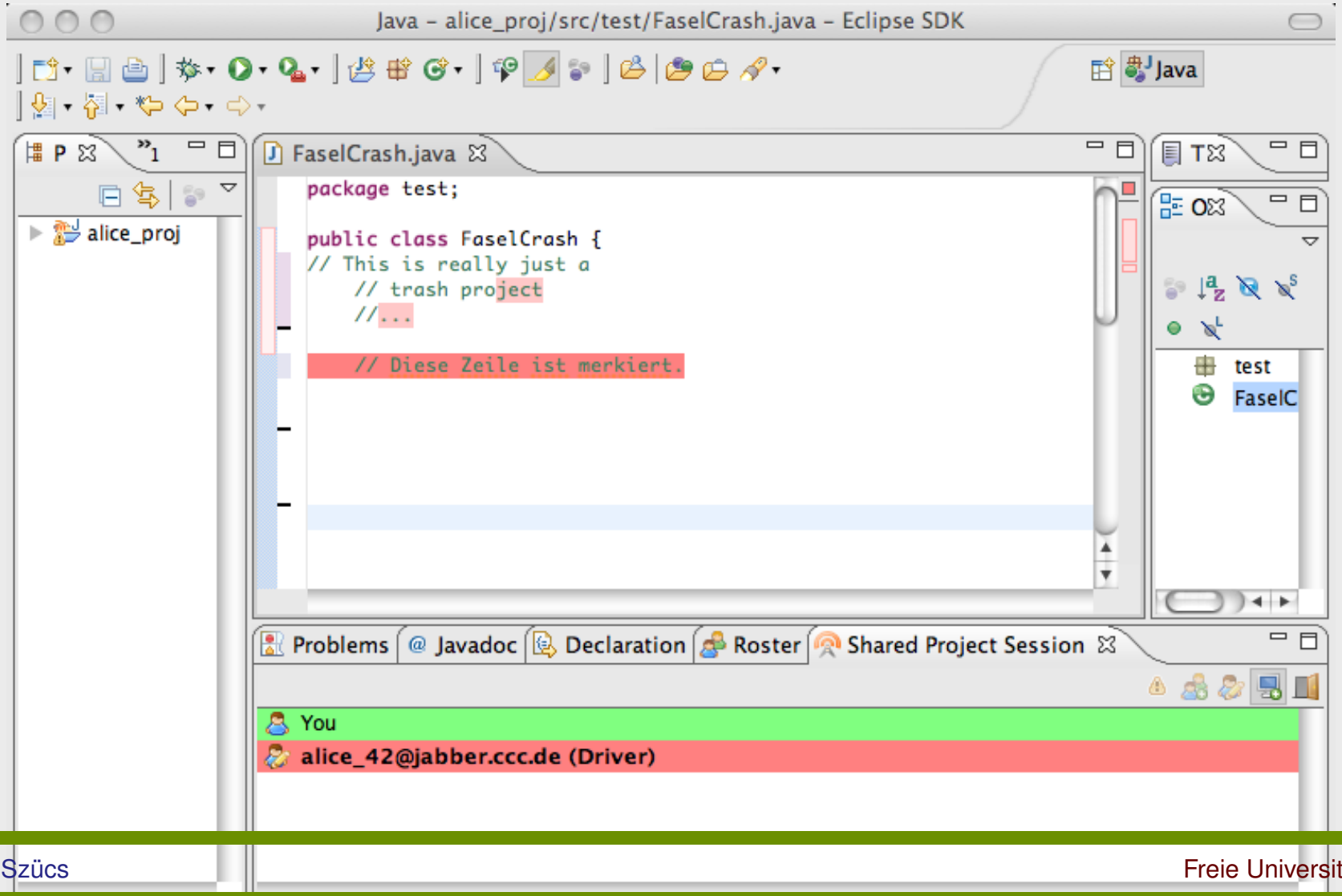
Saros

- Software die Paarprogrammierung und Side-by-Side-Programmierung in Eclipse unterstützt
- Modi in einer Sitzung:
  - exklusive Driver Rolle (klassische PP)
  - mehrere Driver gleichzeitig (side-by-side)
  - Verfolger Modus - unterstützt beide Varianten
- unterstützt mehr als zwei Teilnehmer
- einfache Installation per ‘update site’
- entwickelt als OpenSource
- real eingesetzt in der Firma Teles (www.teles.de)

Saros

- Software die Paarprogrammierung und Side-by-Side-Programmierung in Eclipse unterstützt
- Modi in einer Sitzung:
  - exklusive Driver Rolle (klassische PP)
  - mehrere Driver gleichzeitig (side-by-side)
  - Verfolger Modus - unterstützt beide Varianten
- unterstützt mehr als zwei Teilnehmer
- einfache Installation per ‘update site’
- entwickelt als OpenSource
- real eingesetzt in der Firma Teles (www.teles.de)

Saros Screenshot



Motivation

- Interesse an
  - IT-Sicherheit
  - Programmierung
  - Software Engineering
  - Netzen
- Diese Software habe ich im Studium gesucht!

Motivation

- Interesse an
  - IT-Sicherheit
  - Programmierung
  - Software Engineering
  - Netzen
- Diese Software habe ich im Studium gesucht!

Netzwerk

- Verfügbarkeit
- Authentifizierung
- Verlässlichkeit

Sicherheit

- Bedrohungen
  - Network items
  - Fälschbarkeit von Aktivitäten, Identitäten, ...?

Arbeitspakete

- AP1 Awareness über Netzlast, Verbindungsstatus, Latenz, Queuefüllstand
- AP2 Geschwindigkeit der Dateisynchronisation optimieren
- AP3 find matching project
- AP4 übertragen von Nachrichten via P2P
- AP5 Cancelation und Progress-Support von Dateiübertragungen
- AP6 Jingle-Übertragung (P2P) durch FW und NAT
- AP7 Smack-API-Liaison (Community Arbeit)
- AP8 Test und Optimierung von Saros in Netzen mit hoher Latenz
- AP9 Testsuite für die Netzkomponente entwickeln

AP1 - Awareness

- Netzlast ?
- Verbindungsstatus (Jingle-TCP/Jingle-UDP/IBB/Fehler)
- Latenz
  - Jingle-TCP: messen per RTT von TCP
  - Jingle-UDP: benutzt RUDP kein RTT verfügbar → selbst messen
  - IBB: messen der Beantwortung von XMPP Nachrichten
- Queuefüllstand (insbesondere im ActivitySequencer.activities und ActivityQueue.queuedActivities)
- Umfrage nach einer Sitzung für die Diplomarbeit möglich

AP1 - Awareness

- Netzlast ?
- Verbindungsstatus (Jingle-TCP/Jingle-UDP/IBB/Fehler)
- Latenz
  - Jingle-TCP: messen per RTT von TCP
  - Jingle-UDP: benutzt RUDP kein RTT verfügbar $\rightarrow$ selbst messen
  - IBB: messen der Beantwortung von XMPP Nachrichten
- Queuefüllstand (insbesondere im ActivitySequencer.activities und ActivityQueue.queuedActivities)
- Umfrage nach einer Sitzung für die Diplomarbeit möglich

AP2 - Dateisynchronisation

- Ziel: rsync schnell
- Wozu? Dateiabgleich nach Annahme der Einladung in eine Sitzung.
- aktuell: Diff + Zip
- Wie?
  - rsync Algorithmus [2] ist bekannt, könnte selbst implementiert werden
  - Jarsync [3], ist eine Java Implementierung und wird nicht mehr aktiv entwickelt
- Notwendigkeit fraglich, lösbar durch Einsatz einer Versionsverwwaltung
- abwägen: $\frac{Ertrag}{Aufwand}$

[2] PHD thesis und Techreport von Andrew Tridgell (rsync Autor)

[3] Version 0.3 http://jarsync.sourceforge.net/

AP2 - Dateisynchronisation

- Ziel: rsync schnell
- Wozu? Dateiabgleich nach Annahme der Einladung in eine Sitzung.
- aktuell: Diff + Zip
- Wie?
  - rsync Algorithmus [2] ist bekannt, könnte selbst implementiert werden
  - Jarsync [3], ist eine Java Implementierung und wird nicht mehr aktiv entwickelt
- Notwendigkeit fraglich, lösbar durch Einsatz einer Versionsverwaltung
- abwägen: $\frac{Ertrag}{Aufwand}$

[2] PHD thesis und Techreport von Andrew Tridgell (rsync Autor)

[3] Version 0.3 http://jarsync.sourceforge.net/

AP3 - find matching project

- Diff und Bewertung als notwendig
- hinreichend (?):
  - gleicher Projekt-Name
  - gleiche Pakethierarchie
  - ...
  - Diff über alle Dateien aller lokalen Projekte (~aktuell)
- Ziel: eine schlauere Heuristik als bisher
- Idee: Abbruch von der Vergleiche von Projekten bei hoher Gleichheit
- abwägen: $\frac{Ertrag}{Aufwand}$

AP4 - Nachrichtenübertragung via P2P

- Implementierung weitestgehend vorhanden
- noch zu leisten:
  - mehr paralleles senden und empfangen
  - Priorisierung
  - Blockierungsfreiheit
  - Durchsicht-Richtigkeit
    - Zustände: CONNECTED CONNECTING DISCONNECTED DISCONNECTING ERROR
    - Richtige Handlungen im jeweiligen Zustand und Übergang?
  - Durchsicht-Sicherheit
    - Authentizität (wie prüfe ich mein gegenüber?)
    - Unverfälschbarkeit von Aktivitäten wie Dateioperationen
    - Verfügbarkeit

AP4 - Nachrichtenübertragung via P2P

- Implementierung weitestgehend vorhanden
- noch zu leisten:
  - mehr paralleles senden und empfangen
  - Priorisierung
  - Blockierungsfreiheit
  - Durchsicht-Richtigkeit
    - Zustände: CONNECTED CONNECTING DISCONNECTED DISCONNECTING ERROR
    - Richtige Handlungen im jeweiligen Zustand und Übergang?
  - Durchsicht-Sicherheit
    - Authentizität (wie prüfe ich mein gegenüber?)
    - Unverfälschbarkeit von Aktivitäten wie Dateioperationen
    - Verfügbarkeit

AP4 - Nachrichtenübertragung via P2P

- Implementierung weitestgehend vorhanden
- noch zu leisten:
  - mehr paralleles senden und empfangen
  - Priorisierung
  - Blockierungsfreiheit
  - Durchsicht-Richtigkeit
    - Zustände: CONNECTED CONNECTING DISCONNECTED DISCONNECTING ERROR
    - Richtige Handlungen im jeweiligen Zustand und Übergang?
  - Durchsicht-Sicherheit
    - Authentizität (wie prüfe ich mein gegenüber?)
    - Unverfälschbarkeit von Aktivitäten wie Dateioperationen
    - Verfügbarkeit

AP5 - Cancelation- und Progress-Support von Dateiübertragungen

- bisher: kein Abbruch der Verbindung durch cancel()

I/O Cancelation via TProgressMonitor [4]

```
1 try { // genauer in den Coderules im Wiki
2 while(! progress.isCanceled()){
3 try {stream.read();}
4 catch {IOException {..}}
5 } finally {stream.close();}
```

[4] Pattern aus 'Concurrent Programming in Java' von Doug Lea

AP6 - Jingle-Übertragung (P2P) durch FW und NAT

- Anwendungsfälle
  1. Entwickler hinter NAT Router
  2. Entwickler hinter einer Firewall (wie hier im MiLan)
- Smack Jingle unterstützt NAT traversal per STUN oder ICE
- bisher werden folgende Wege durch ein NAT erkannt:
  - die zweite Netzverbindung (eth0/eth1)
  - Netzwerkschnittstellen wie VMware
- Anwendungsfälle könnten in der Diplomarbeit untersucht werden

AP6 - Jingle-Übertragung (P2P) durch FW und NAT

- Anwendungsfälle
  1. Entwickler hinter NAT Router
  2. Entwickler hinter einer Firewall (wie hier im MiLan)
- Smack Jingle unterstützt NAT traversal per STUN oder ICE
- bisher werden folgende Wege durch ein NAT erkannt:
  - die zweite Netzverbindung (eth0/eth1)
  - Netzwerkschnittstellen wie VMware
- Anwendungsfälle könnten in der Diplomarbeit untersucht werden

AP7 - Smack-API-Liaison (Community Arbeit)

- Ziel: Beseitigung von Fehlern genutzter Bibliotheken
- Smack-API - schwieriger Zugang
  - E-Mails wurden nicht beantwortet
  - diverse Foreneinträge von 2008 unbeantwortet und ungelöst
  - Verhalten bei Einreichung von Patches ist bisher unbekannt
  - Relevanz: hoch
- limewire (RUDP-Implementerung)
  - einfacher Patch ist seit 3-4 Wochen unbeantwortet
  - Relevanz: niedrig
- Smack forken wäre wahrscheinlich eine Diplomarbeit für sich

AP7 - Smack-API-Liaison (Community Arbeit)

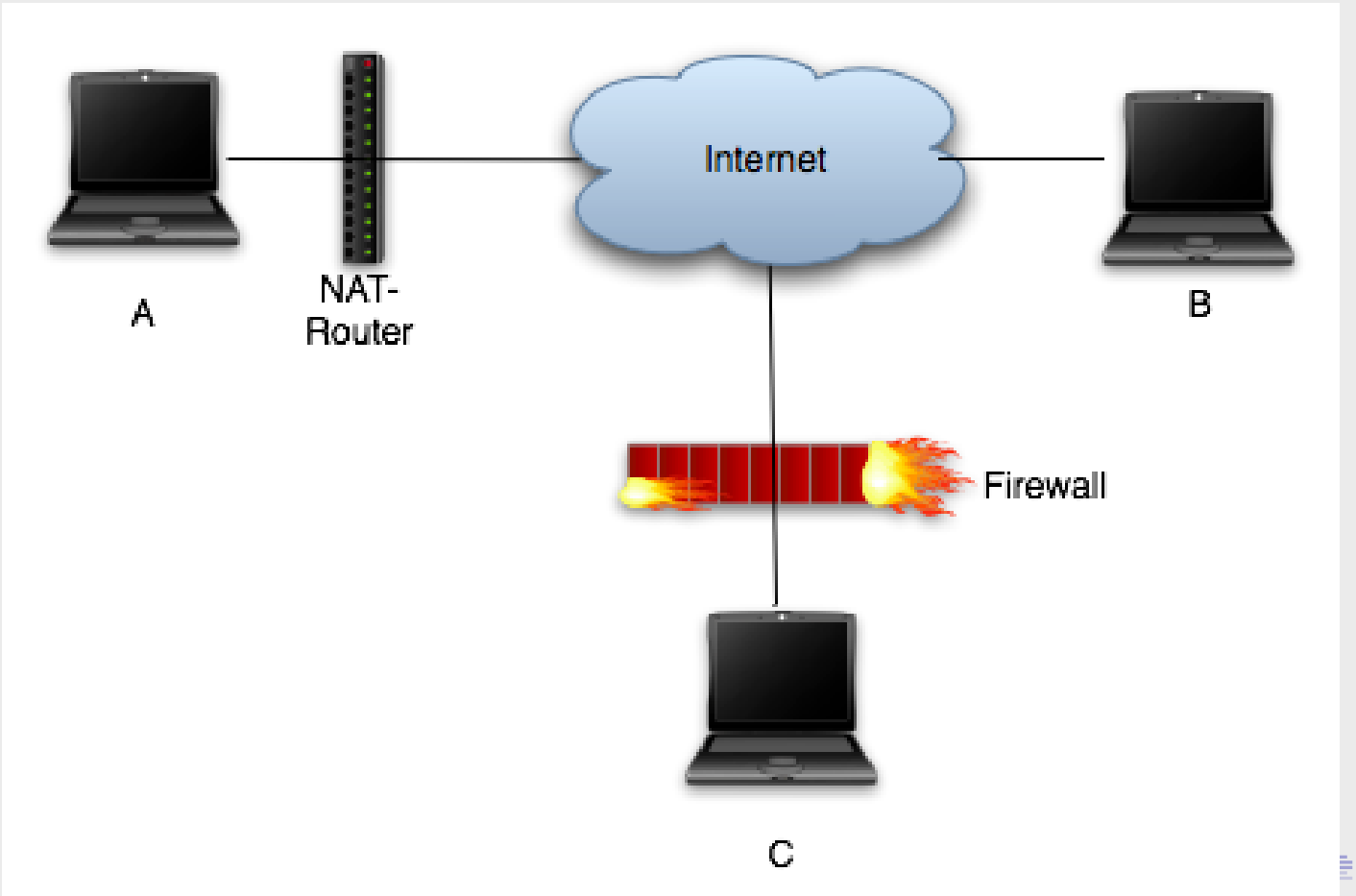
- Ziel: Beseitigung von Fehlern genutzter Bibliotheken
- Smack-API - schwieriger Zugang
  - E-Mails wurden nicht beantwortet
  - diverse Foreneinträge von 2008 unbeantwortet und ungelöst
  - Verhalten bei Einreichung von Patches ist bisher unbekannt
  - Relevanz: hoch
- limewire (RUDP-Implementerung)
  - einfacher Patch ist seit 3-4 Wochen unbeantwortet
  - Relevanz: niedrig
- Smack forken wäre wahrscheinlich eine Diplomarbeit für sich

AP8 - Test und Optimierung von Saros in Netzen mit hoher Latenz

- Simulation notwendig
- einfache Variante: Latenz durch Wartezeit in send()
- Umfrage über Usability in Netzen mit hoher Latenz möglich

AP9 - Testsuite für die Netzkomponente entwickeln

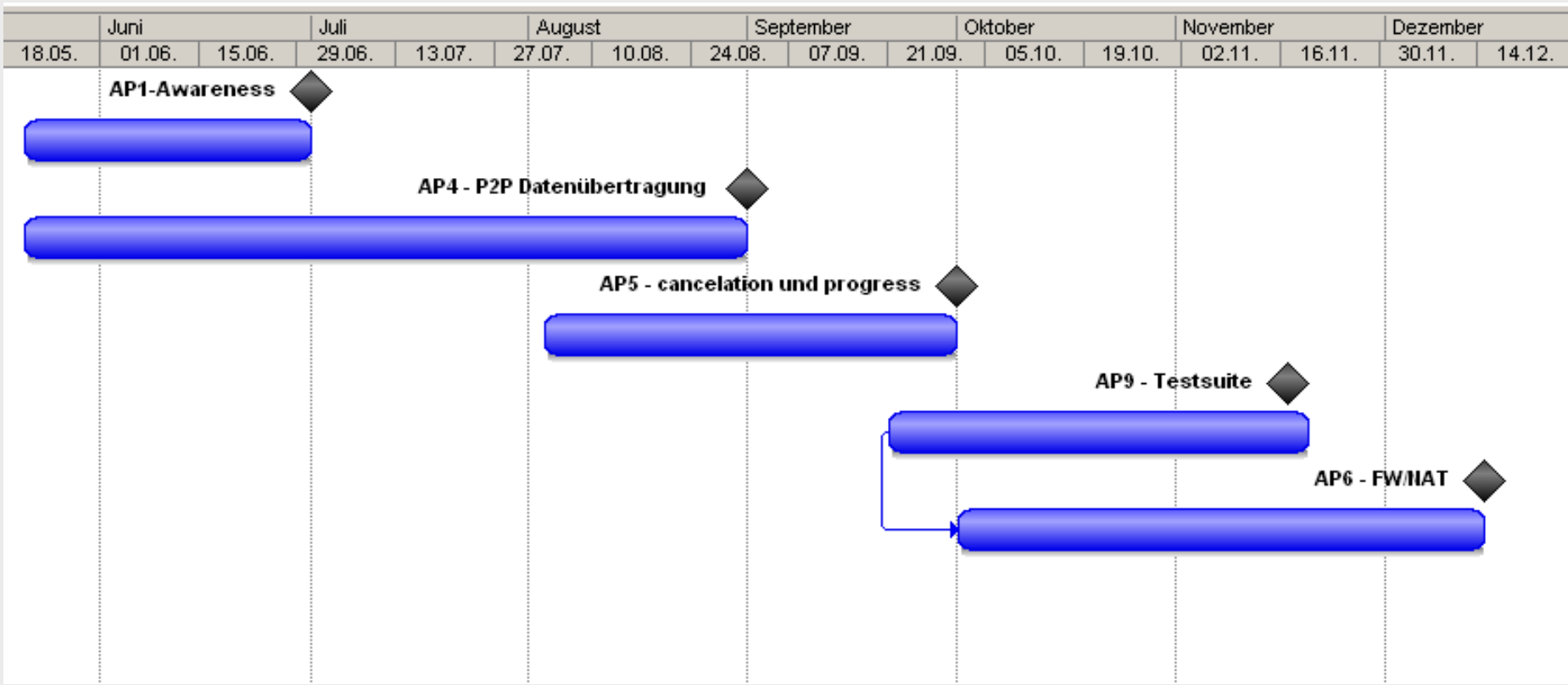
als Test für AP6 (FW/NAT) notwendig



AP - Review

- wahrscheinlich ist nicht alles zeitlich schaffbar, daher Priorisierung notwendig
  1. AP1 - Awareness → Umfrage: Ausreichend? Was fehlt? Was ist hilfreich für eine DPP Sitzung?
  2. AP4 - P2P Nachrichtenübertragung
  3. AP5 - Cancelation und Progress
  4. AP6 - P2P durch FW und NAT (bedingt AP9 Testsuite) → Case Study
  5. AP8 - Optimierung für Netze mit hoher Latenz → Umfrage: benutzbar bei hohen Latenzen?
  6. AP2 - Dateisynchronisation
  7. AP3 - find matching project
- außer Konkurrenz: AP7 - Community Arbeit
  Braucht jemand hier eine X-Thema?

Roadmap


Juni
Juli
August
September
Oktober
November
Dezember
18.05.
01.06.
15.06.
29.06.
13.07.
27.07.
10.08.
24.08.
07.09.
21.09.
05.10.
19.10.
02.11.
16.11.
30.11.
14.12.
AP1-Awareness
AP4 - P2P Datenübertragung
AP5 - cancelation und progress
AP9 - Testsuite
AP6 - FW/NAT

Anregungen oder Fragen?

Danke!