OPNET/Riverbed Modeler:
Transport Layer

Roman Dunaytsev

The Bonch-Bruevich Saint-Petersburg
State University of Telecommunications

roman.dunaytsev@spbgut.ru

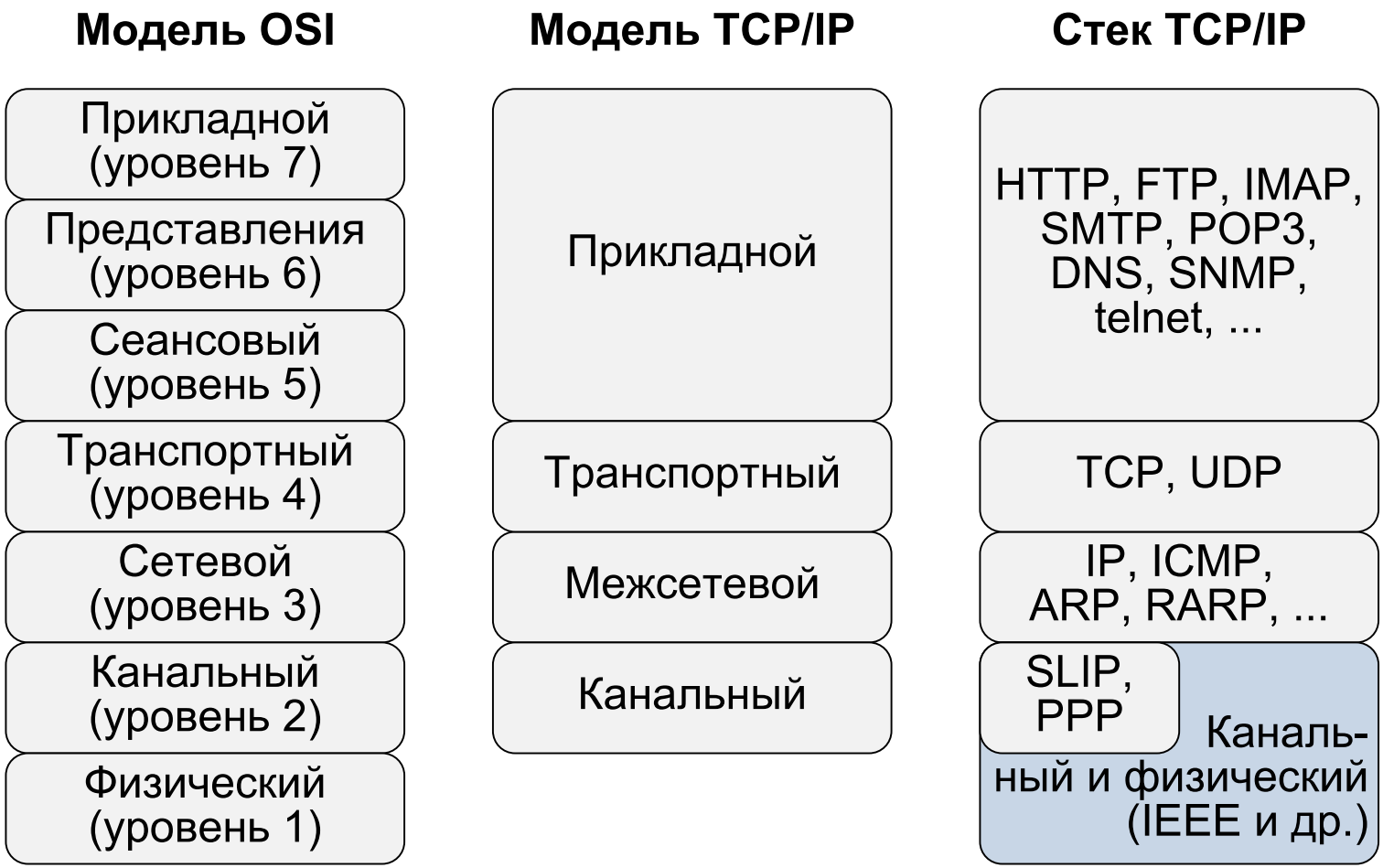
Lecture № 10

Содержание

Содержание

Протоколы транспортного уровня

- Эталонная модель OSI, модель и стек протоколов TCP/IP



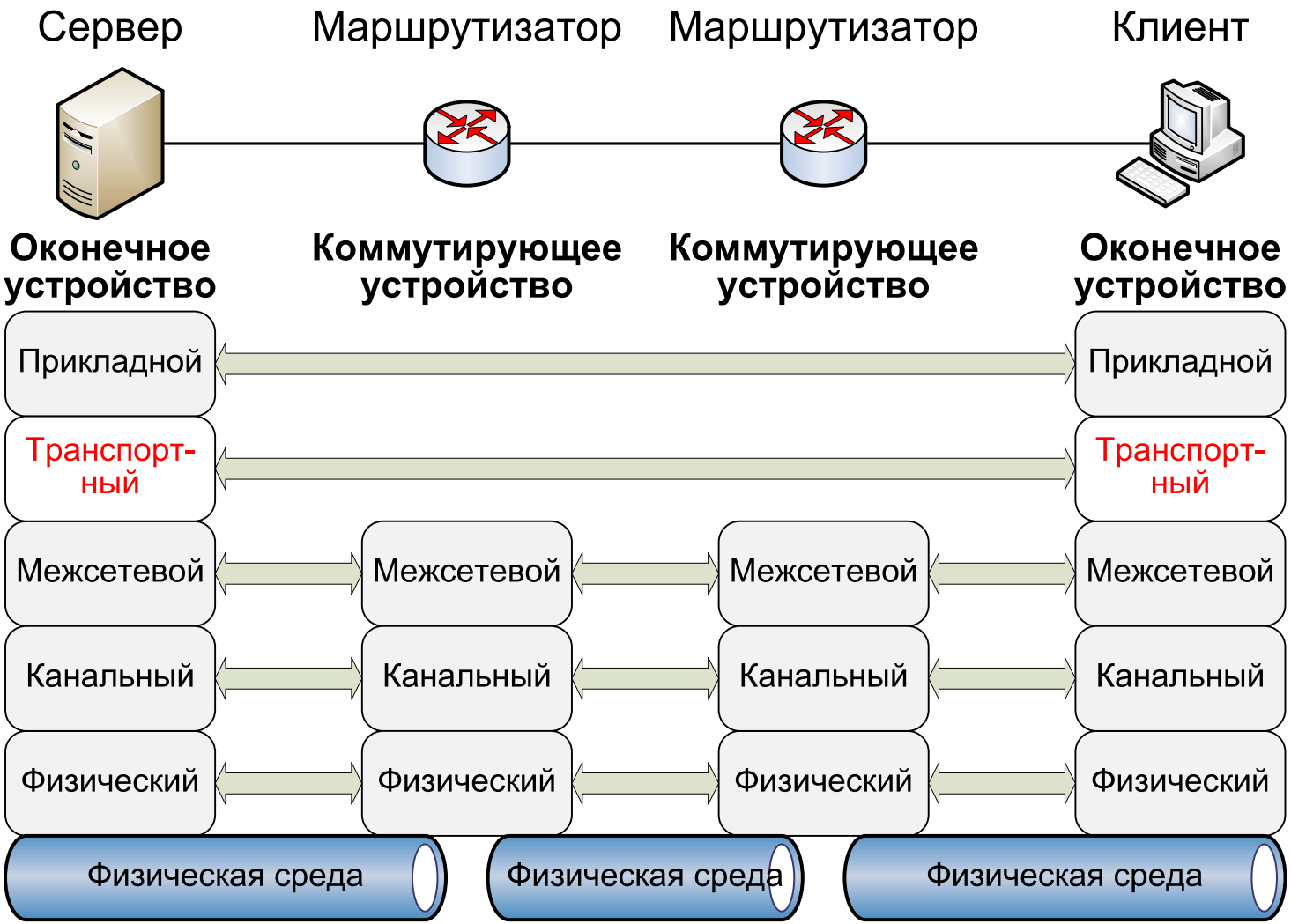
Протоколы транспортного уровня

- User Datagram Protocol vs. Transmission Control Protocol

UDP	TCP
Ориентирован на работу с сообщениями	Ориентирован на работу с потоком байтов
Без установления соединения	С установлением соединения
Без запоминания состояния	С запоминанием состояния
Ненадежный	Надежный
Однонаправленная или групповая передача	Только однонаправленная передача
Используется немногими пользовательскими приложениями (VoIP, потоковое мультимедиа и др.)	Используется многими пользовательскими приложениями (WWW, email, FTP, Telnet и др.)
Используется многими сетевыми службами (RIP, SNMP, DNS и др.)	Используется немногими сетевыми службами (например, зонные передачи DNS)

Протоколы транспортного уровня

- Протоколы UDP и TCP являются **сквозными** (end-to-end)



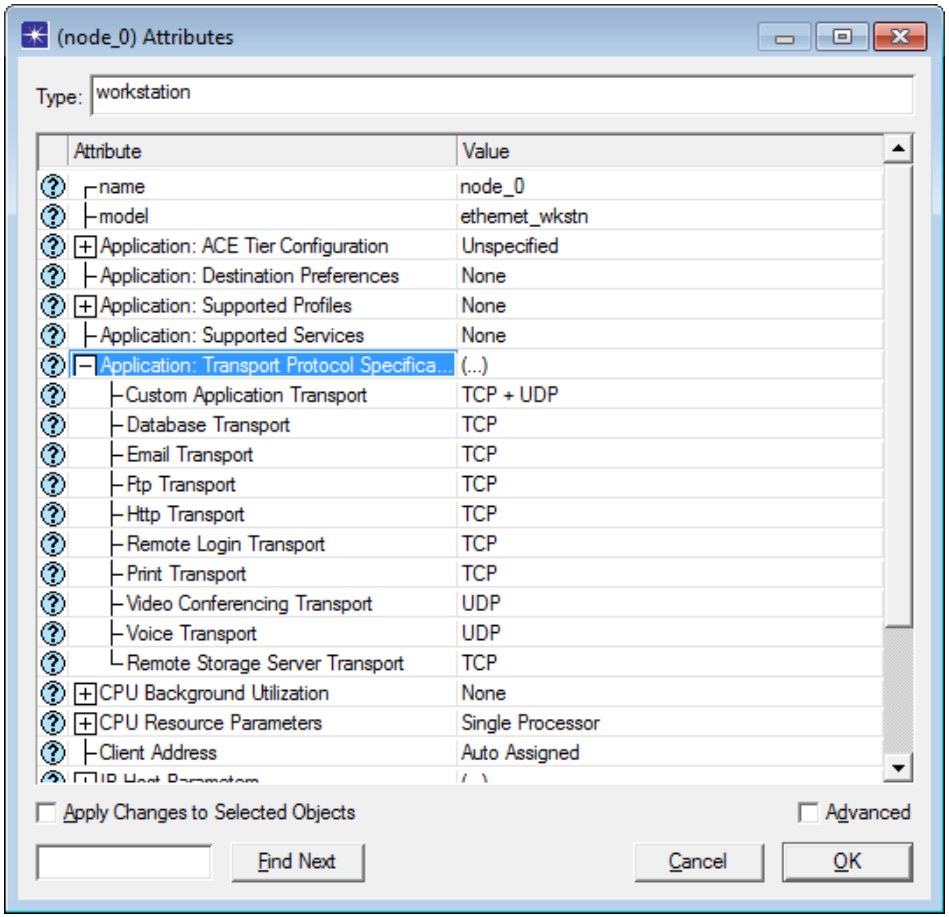
Содержание

Протоколы транспортного уровня в OPNET

- OPNET поддерживает как UDP, так и TCP
- UDP и TCP реализованы лишь в моделях **оконечных** устройств:
  - ***_wkstn** (ethernet_wkstn, ppp_wkstn, . . .)
  - ***_server** (ethernet_server, ppp_server, . . .)
  - ***_LAN** (100BaseT_LAN, FDDI_LAN, . . .)
- TCP используется в большинстве стандартных приложений:
  - Database
  - Email
  - Ftp
  - Http
  - Print
  - Remote Login
  - Remote Storage Server
- UDP используется лишь в:
  - Video Conferencing
  - Voice

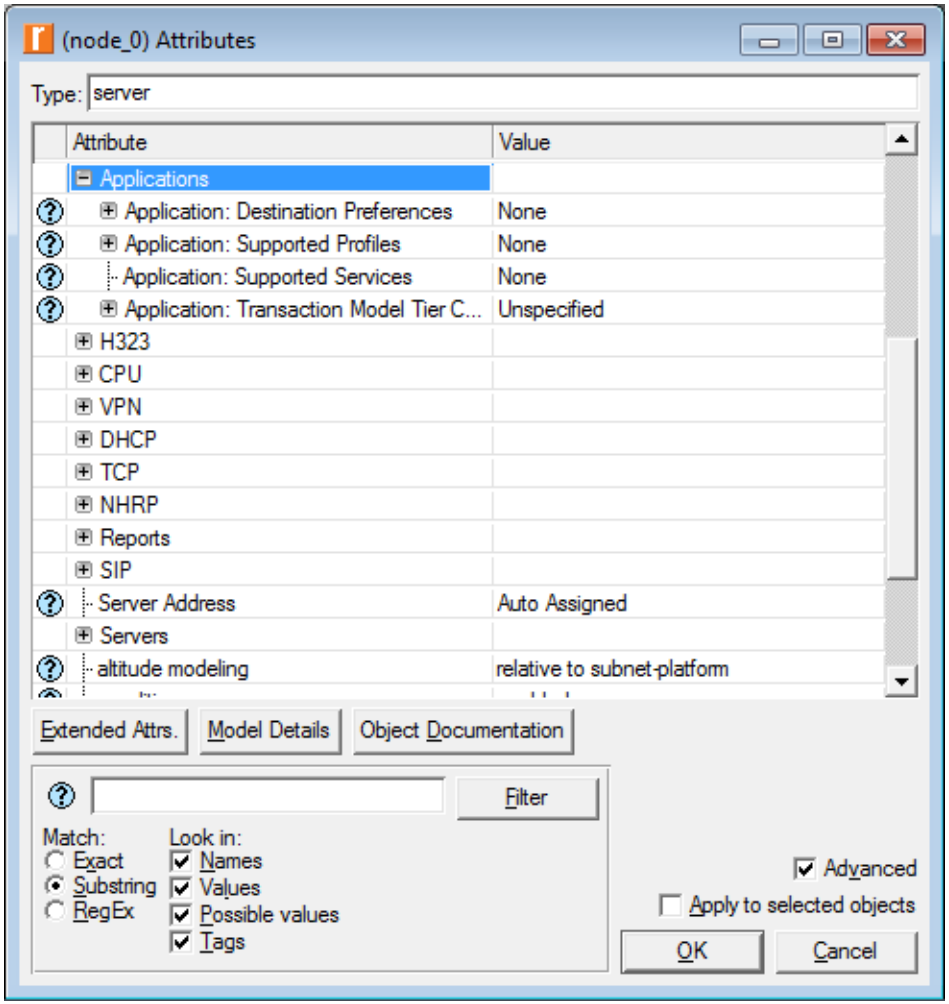
Протоколы транспортного уровня в OPNET

- Используемые протоколы транспортного уровня можно изменить, поменяв значение параметра **Application: Transport Protocol Specification**



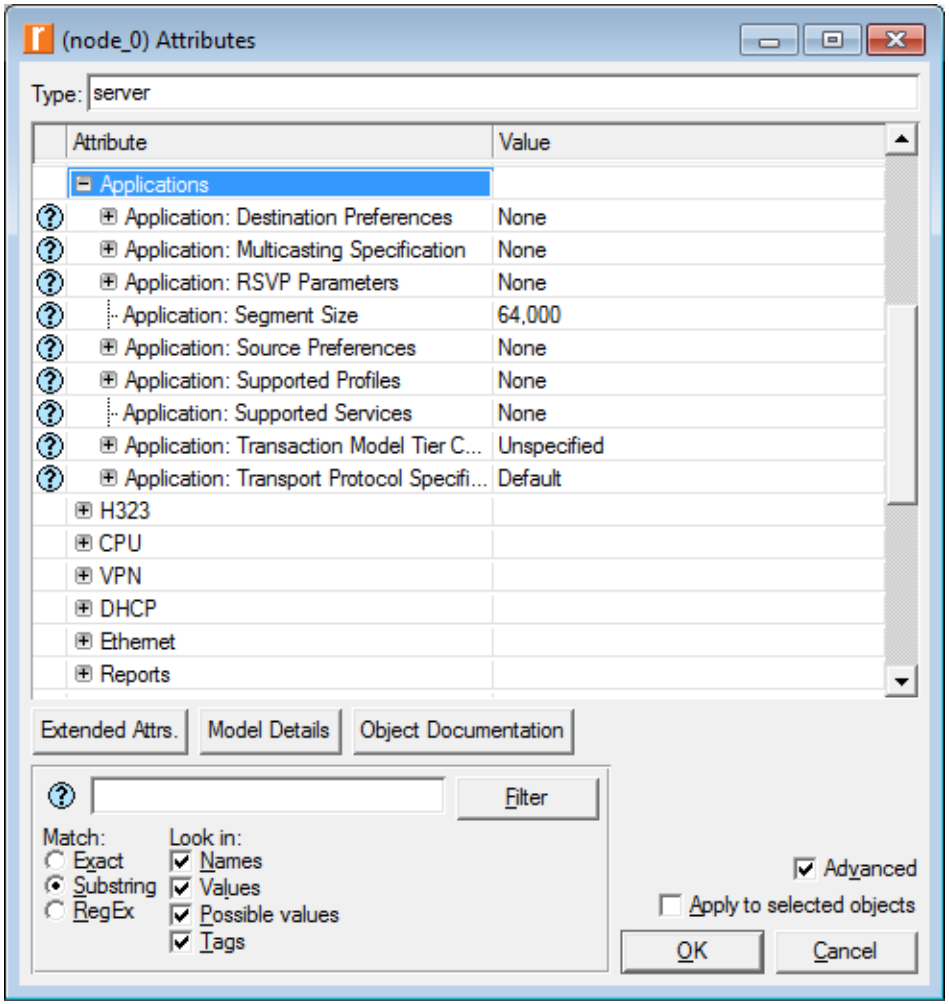
Протоколы транспортного уровня в OPNET

- В серверах этот параметр отсутствует в стандартных моделях (***_server**)



Протоколы транспортного уровня в OPNET

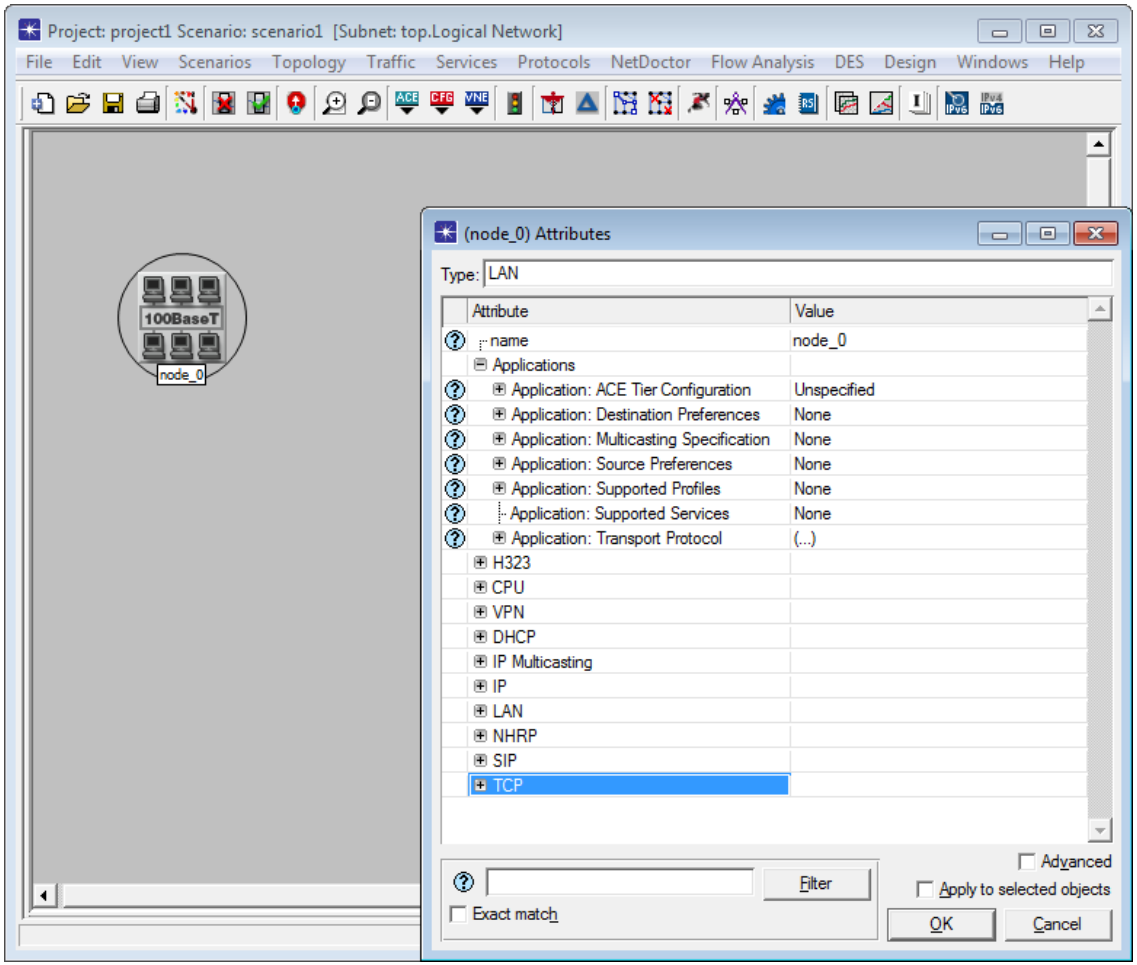
- Лишь модели с дополнительными опциями имеют эту настройку (***_server_adv**)



Содержание

UDP в OPNET

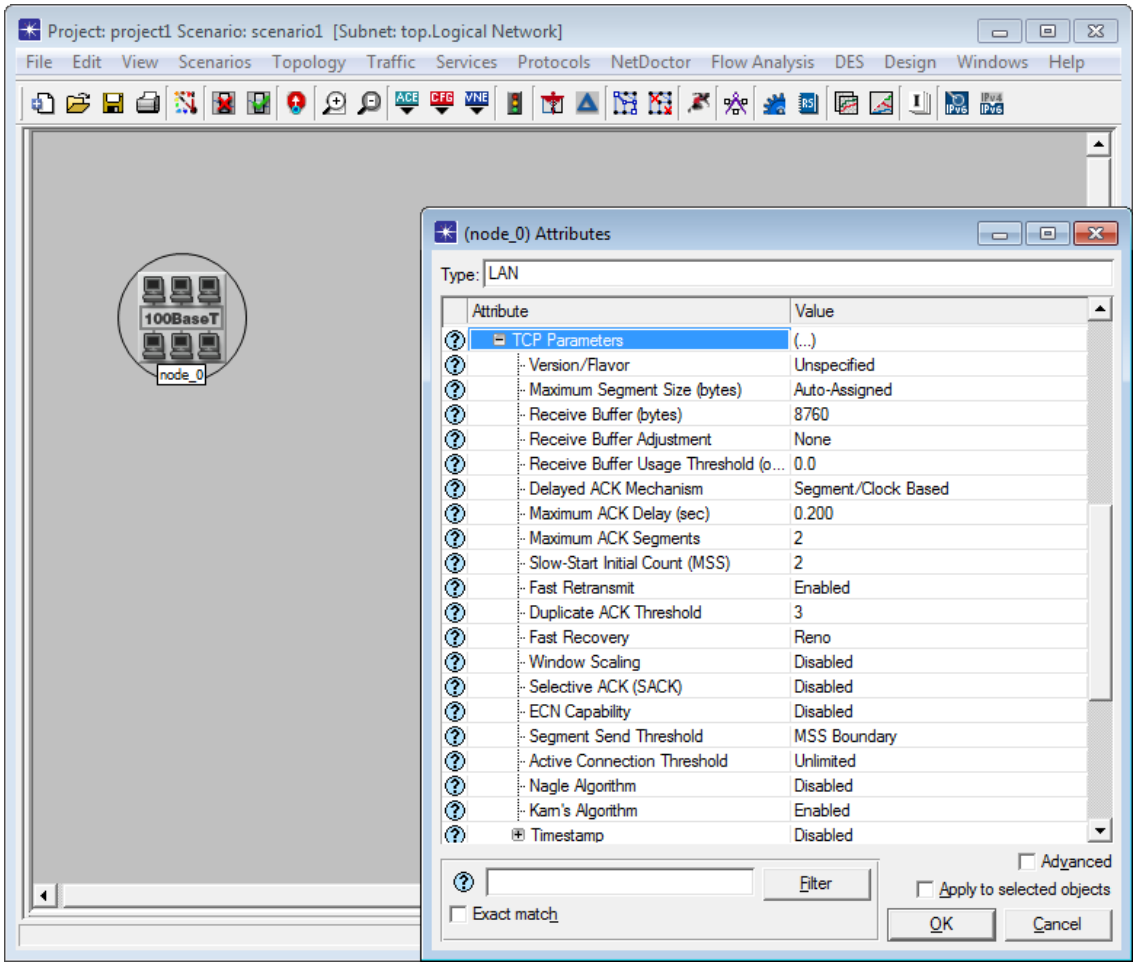
- UDP, будучи достаточно простым протоколом, не имеет особых настроек в OPNET



Содержание

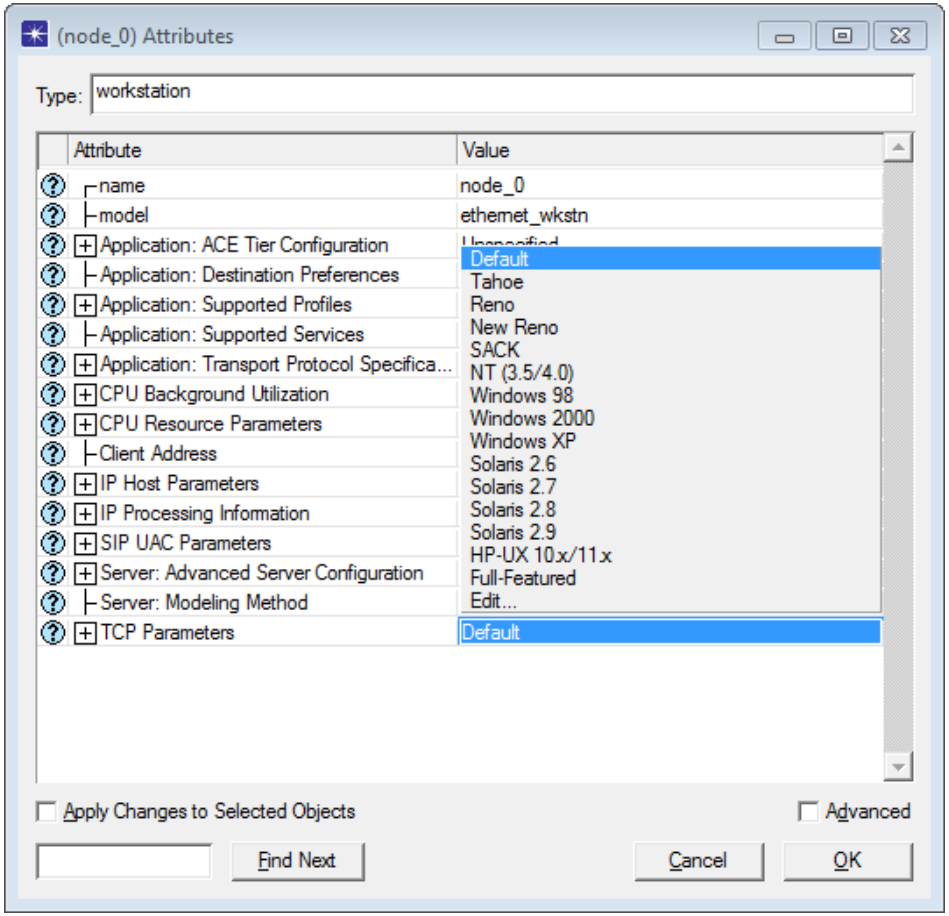
TCP в OPNET

- TCP, будучи достаточно сложным протоколом, имеет множество настроек в OPNET



TCP в OPNET

- **Version/Flavor**
  - По алгоритмам (Tahoe, Reno, New Reno, SACK, Full-Featured)
  - По ОС (NT (3.5/4.0), Windows 98, Windows 2000, . . .)



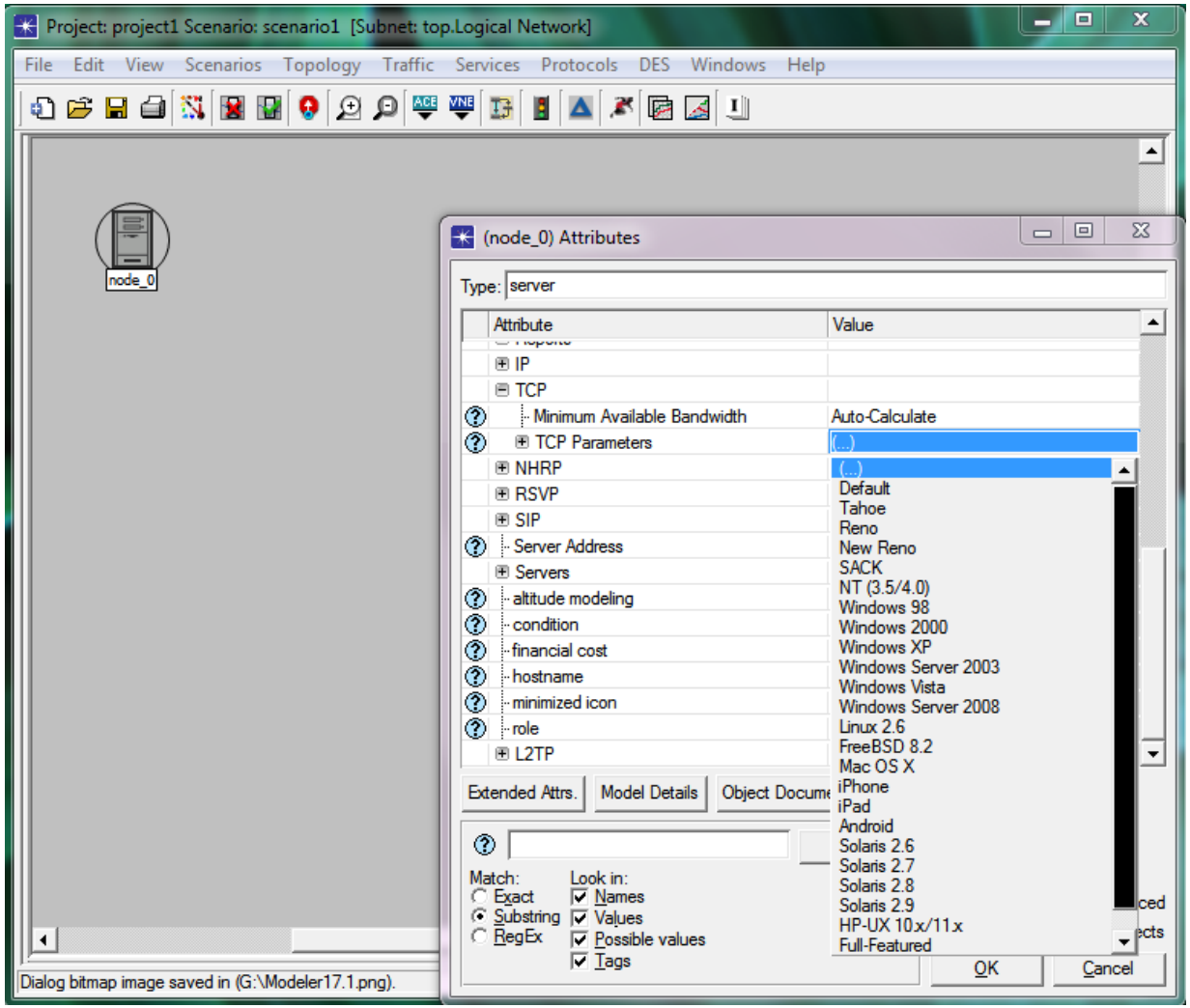
TCP в OPNET

- Стандартные версии TCP (см. Request for Comments, RFC)

Алгоритм	**Версия TCP**			
	Tahoe	**Reno**	**NewReno**	**SACK**
Подтверждения	Кумулятив-ные	Кумулятив-ные	Кумулятив-ные	Кумулятив-ные и выбо-рочные
Алгоритм Карна				
Медленный запуск				
Предотвращение перегрузки				
Быстрая повторная передача				Расширен-ный
Быстрое восстановление	Нет		Модифици-рованный	Расширен-ный

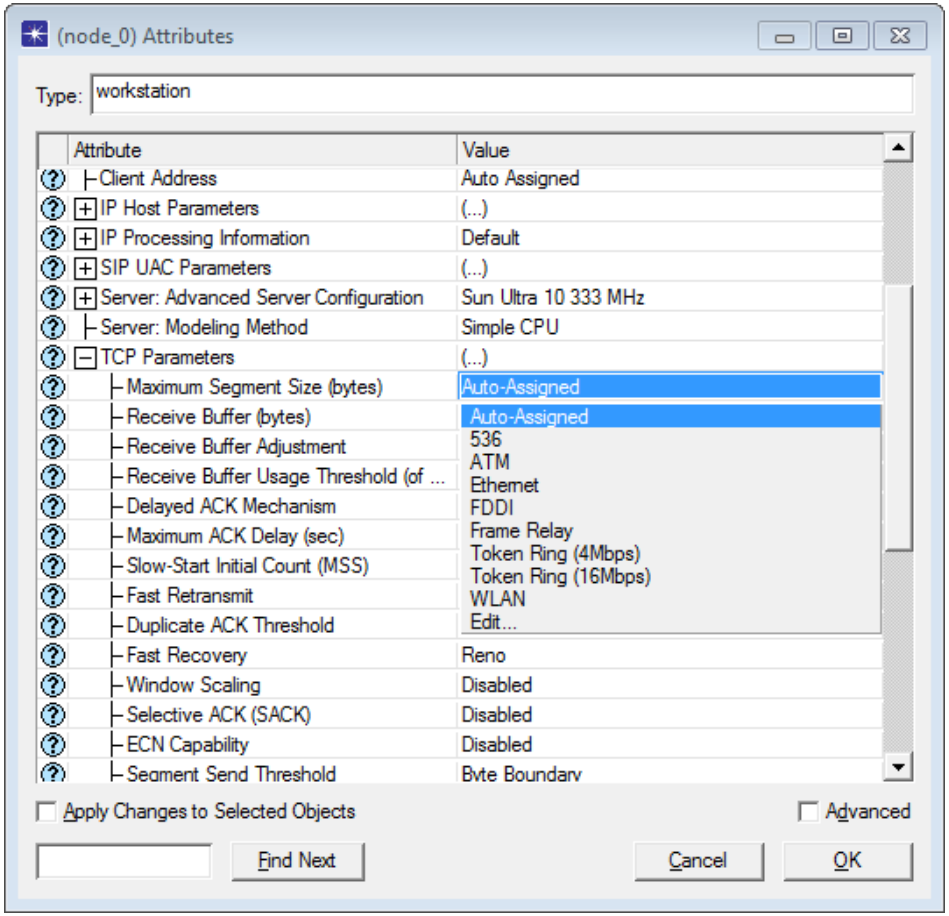
TCP в OPNET

- OPNET Modeler 17.1: новые ОС и новые варианты настроек TCP



TCP в OPNET

- **Maximum Segment Size (MSS)**
  - MSS = MTU – заголовок IP – заголовок TCP
  - Ethernet II: 1500 – 20 – 20 = 1460 байт

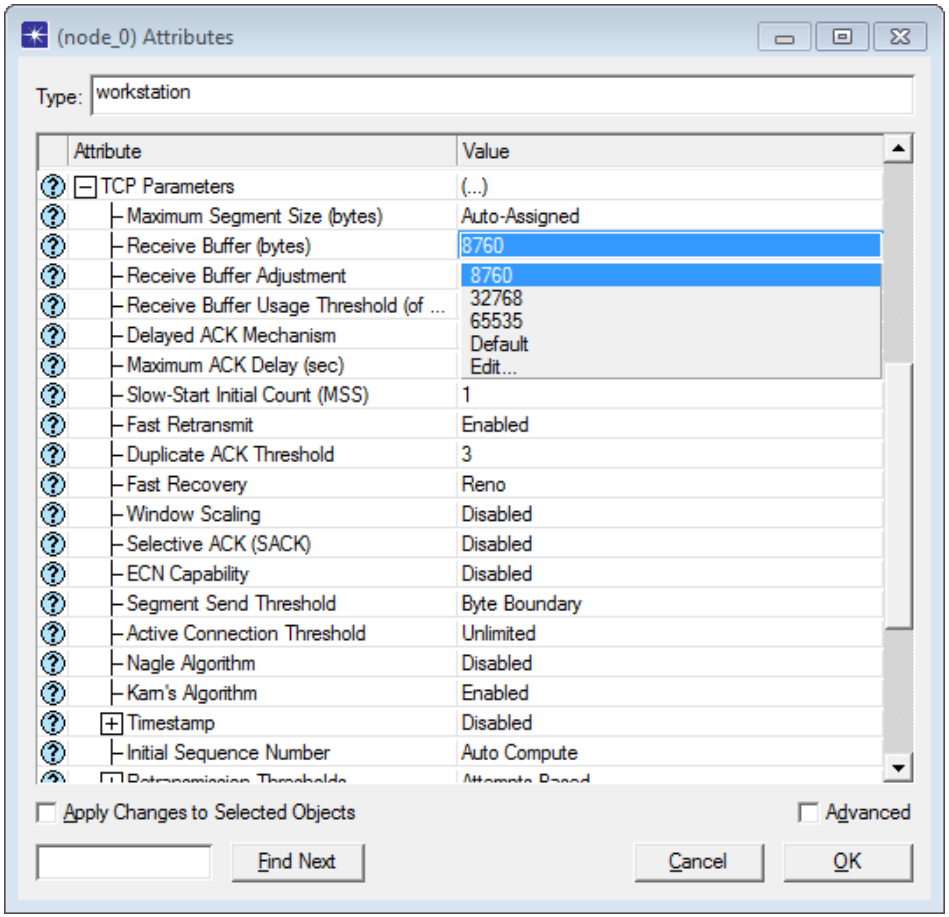


TCP в OPNET

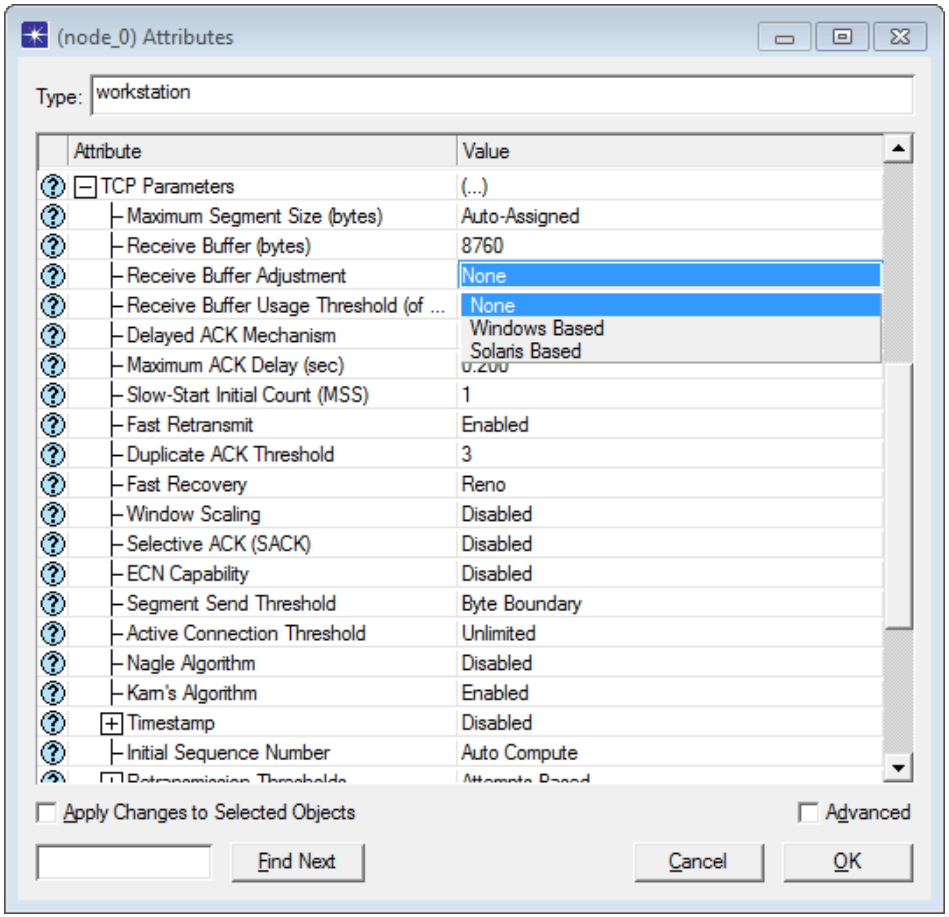
- **Maximum Transmission Unit (MTU)**

RFC	Технология	MTU (байт)
894	Ethernet II (DIX-Ethernet)	1500
1042	Ethernet SNAP	1492
1042	Token Bus (IEEE 802.4)	8166
1042	4 Mbits/s Token Ring (IEEE 802.5)	4464
отсутствует	16 Mbits/s Token Ring (IBM)	17914
1055	Serial Line Internet Protocol (SLIP)	1492
1356	X.25, ISDN	1500
1626	IP over ATM AAL5	9180
1661	Point-to-Point Protocol (PPP)	1500
2019	Fiber Distributed Data Interface (FDDI)	4352
2516	PPP over Ethernet (PPPoE)	1492

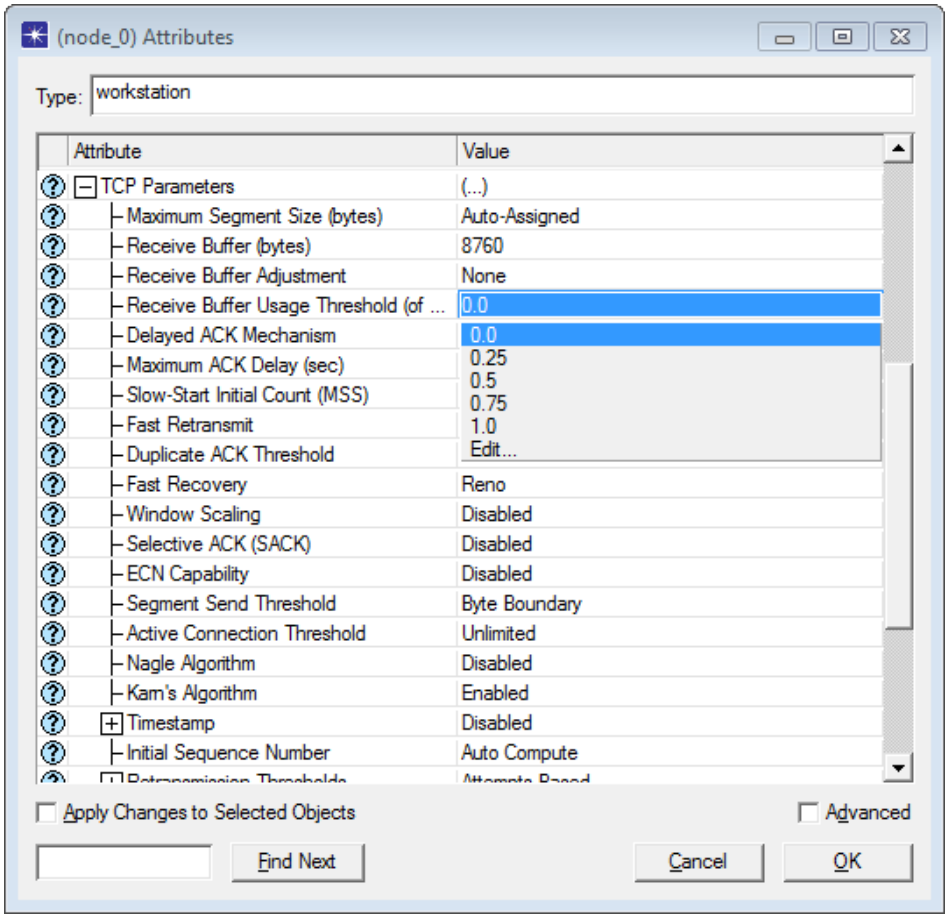
- **Receive Buffer**
  - Размер передаваемого окна = min{cwnd, **rwnd**}



- **Receive Buffer Adjustment**
  - Windows Based: округление rwnd до четного числа MSS
  - Solaris Based: округление rwnd до целого числа MSS



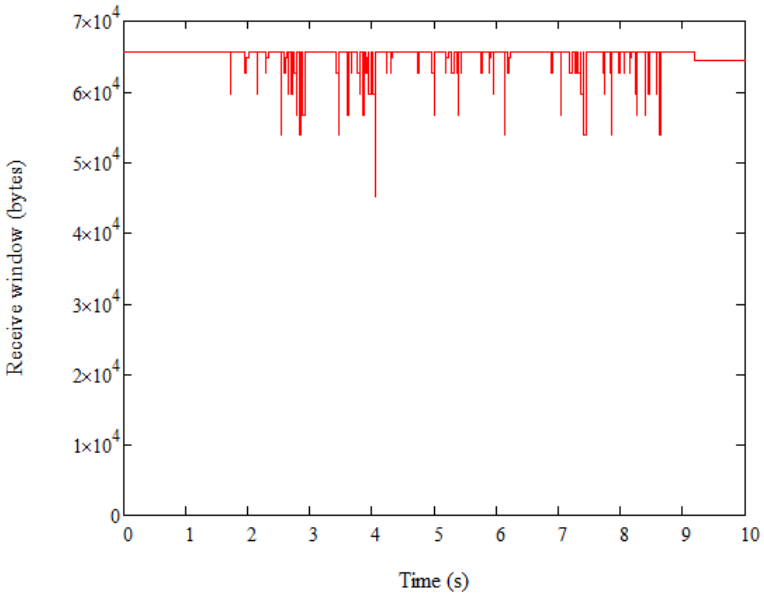
- **Receive Buffer Usage Threshold**
  - Порог занятости буфера, по достижению которого накопленные данные передаются приложению



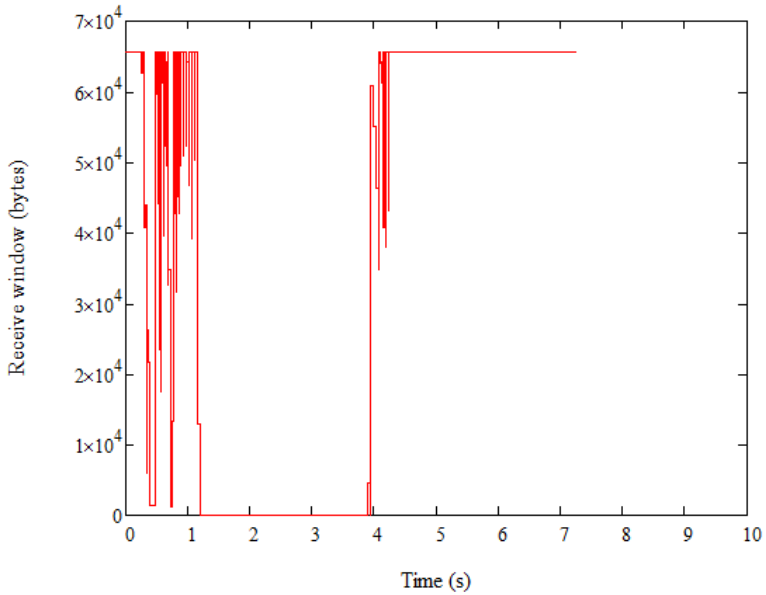
TCP в OPNET

- В реальности размер окна приемника (rwnd) может изменяться динамически

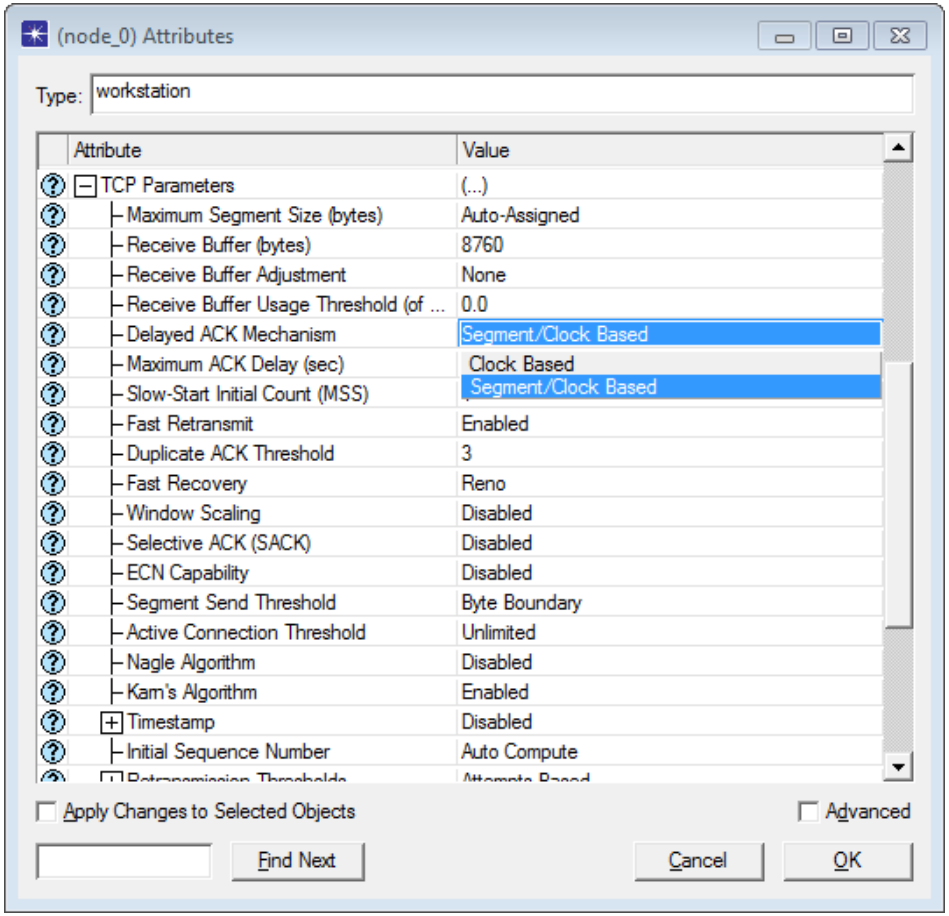
- Ненулевое окно приемника



- Нулевое окно приемника

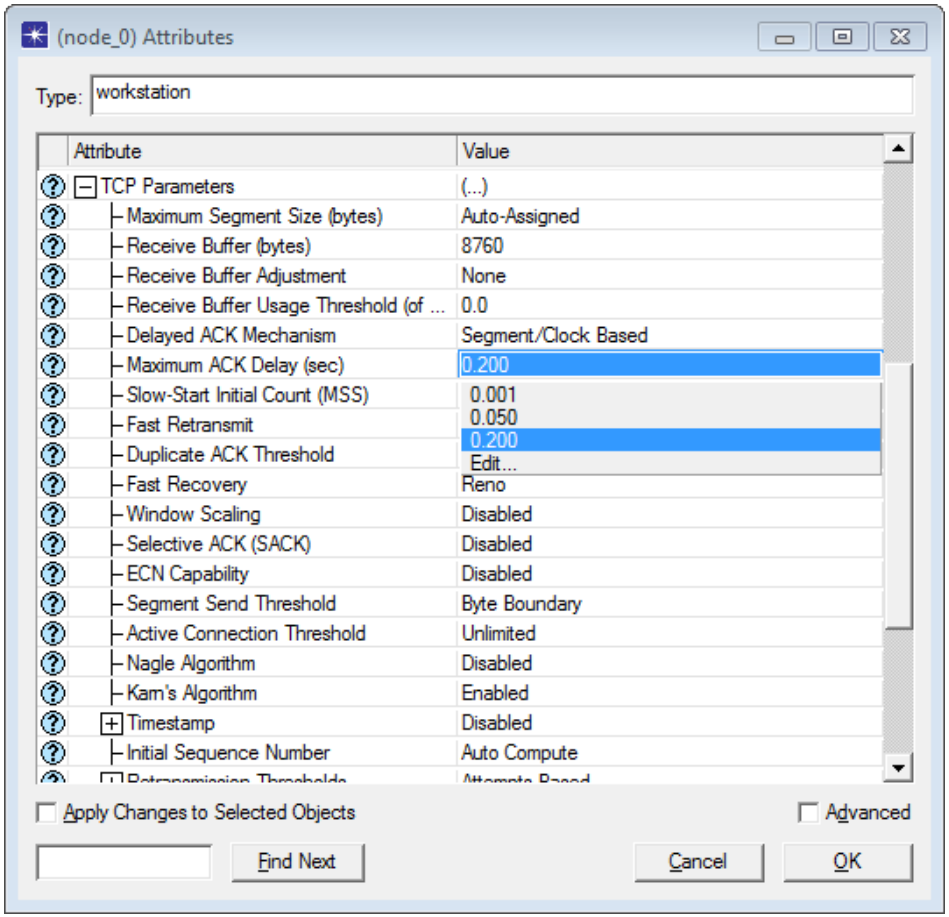


- **Delayed ACK Mechanism**
  - Clock Based: ACK по таймауту
  - Segment/Clock Based: по таймауту или на каждые 2 сегмента



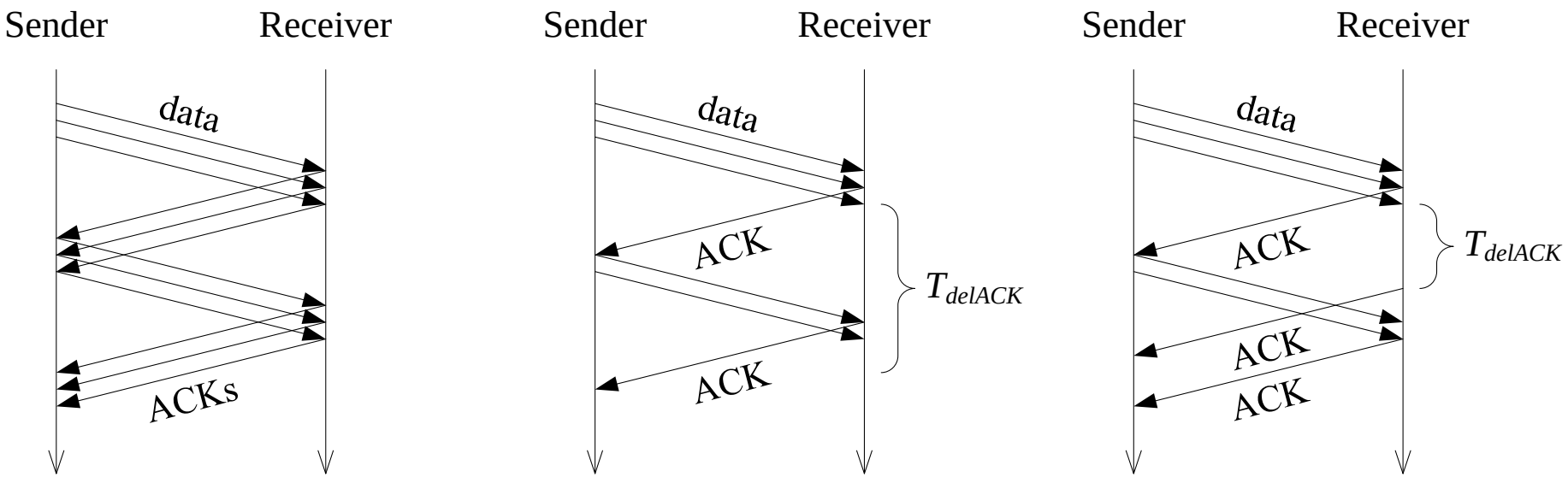
TCP в OPNET

- **Maximum ACK Delay**
  - RFC 1122: не более 500 мс
  - На практике: 200 мс

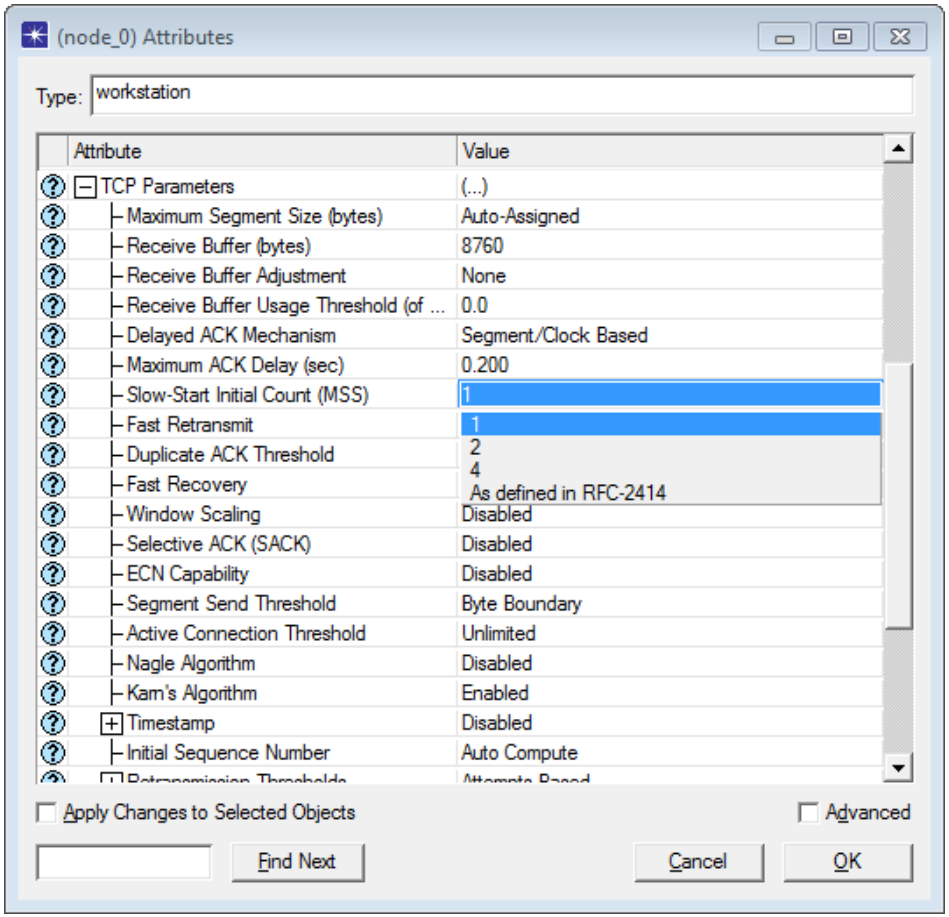


TCP в OPNET

- Алгоритм отправки ACK-сегментов (без данных)
  - Подтверждения посылаются без задержки
  - Второй сегмент приходит раньше, чем истекает таймаут
  - Таймаут истекает раньше, чем приходит второй сегмент

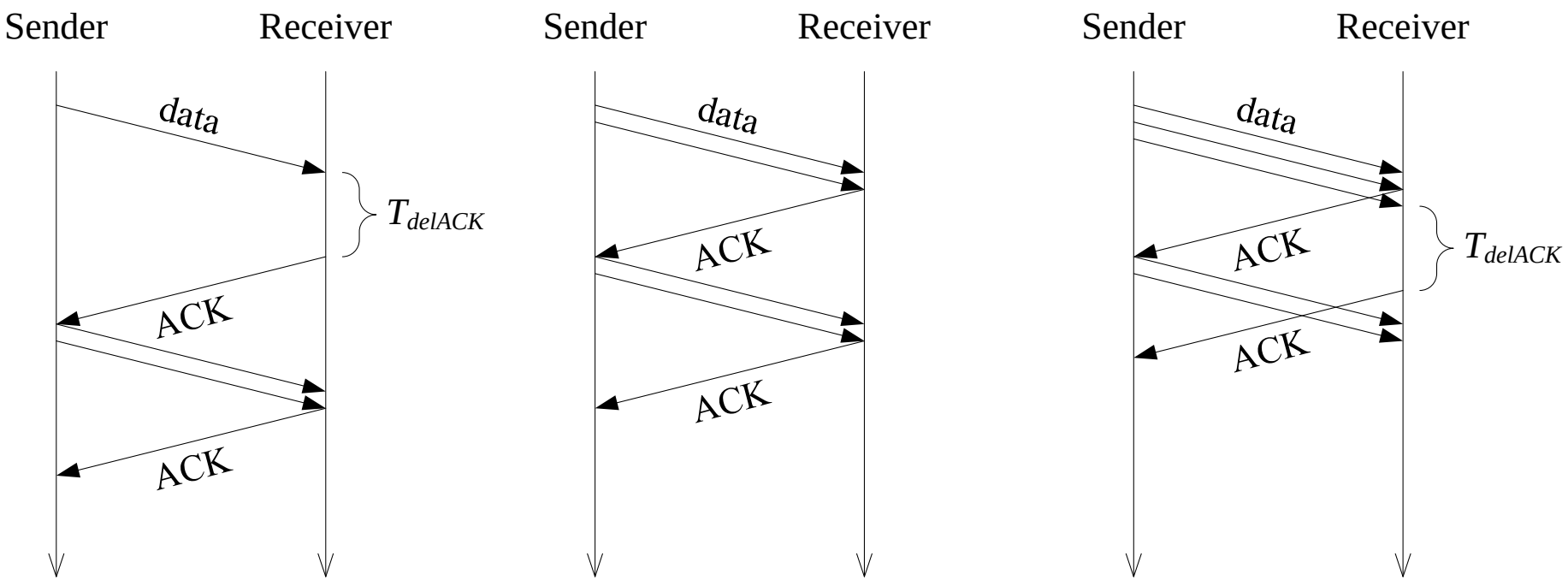


- **Slow-Start Initial Count (MSS)**
  - RFC 3390: Initial Window = min{4MSS, max{2MSS, 4380 байт}}
  - Ethernet II: IW = 1×1460, 2×1460, 3×1460 байт



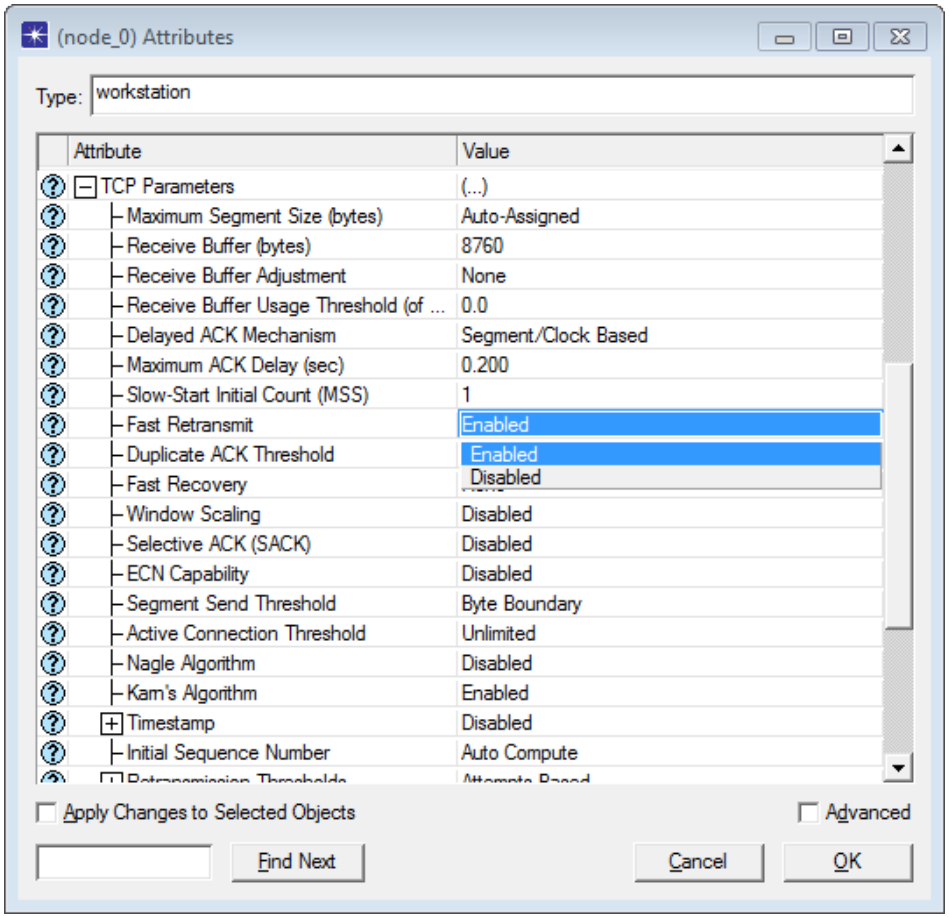
TCP в OPNET

- IW и отправка ACK-сегментов
  - Для передачи небольших объемов данных использование большего значения исходного окна (IW) существенно ускоряет доставку



TCP в OPNET

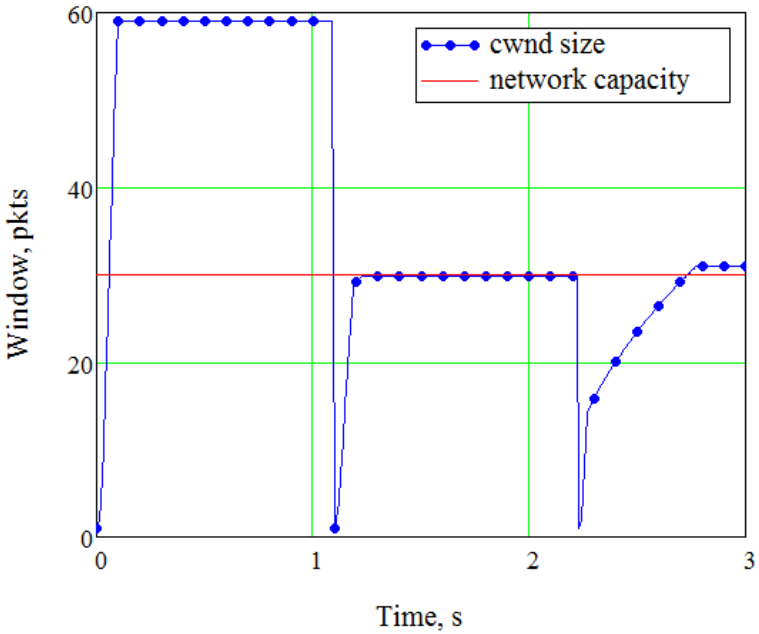
- **Fast Retransmit**
  - Enabled: алгоритм быстрой повторной передачи используется
  - Disabled: обнаружение потерь лишь по таймауту



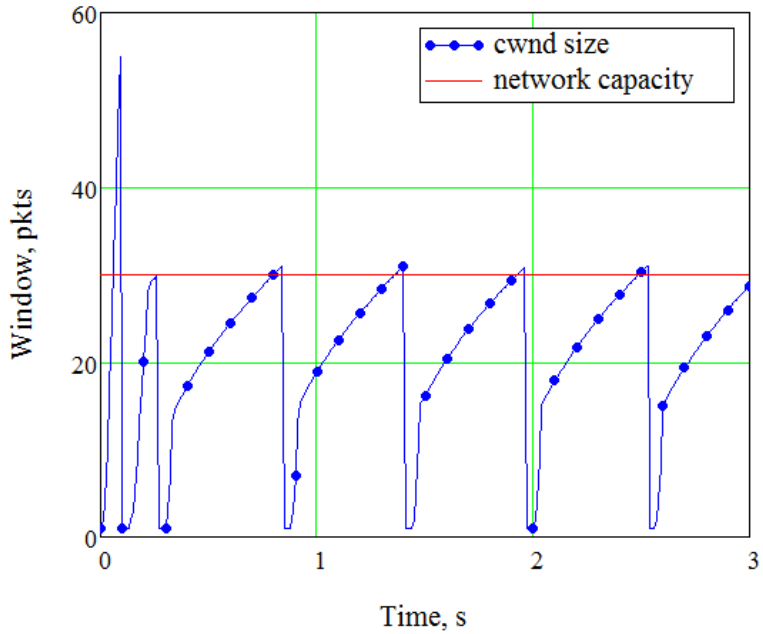
TCP в OPNET

- Использование Fast Retransmit существенно ускоряет обнаружение потери сегмента

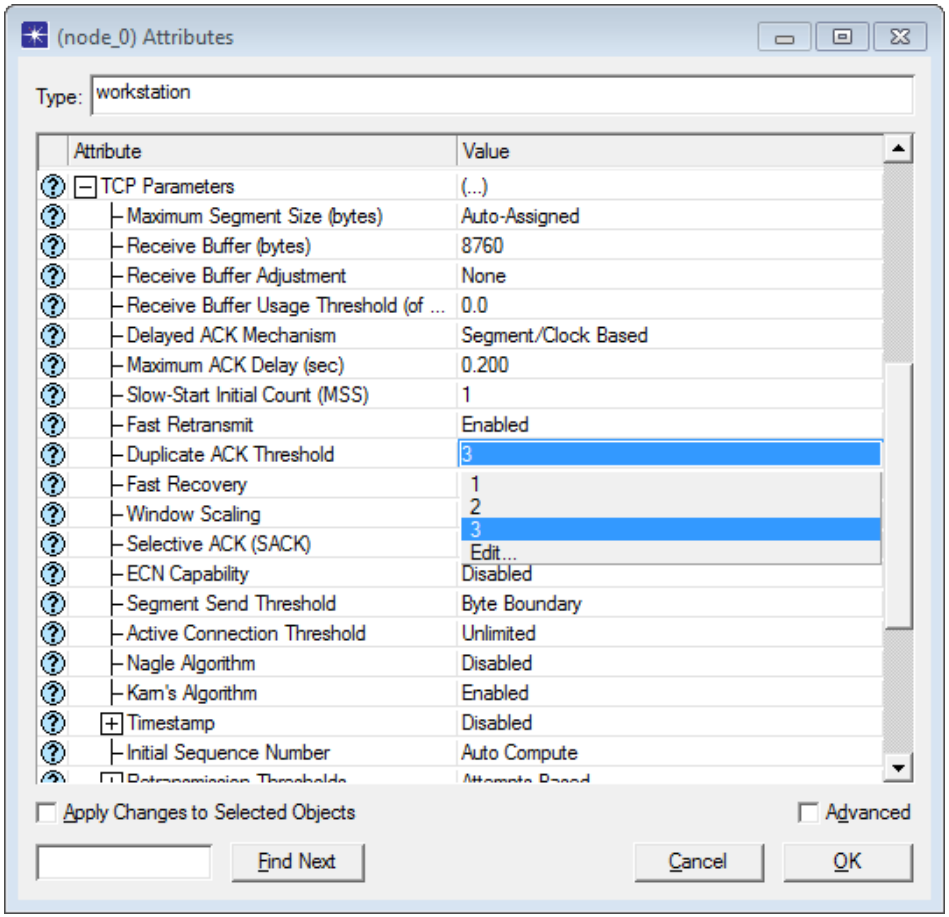
- TCP Tahoe без Fast Retransmit



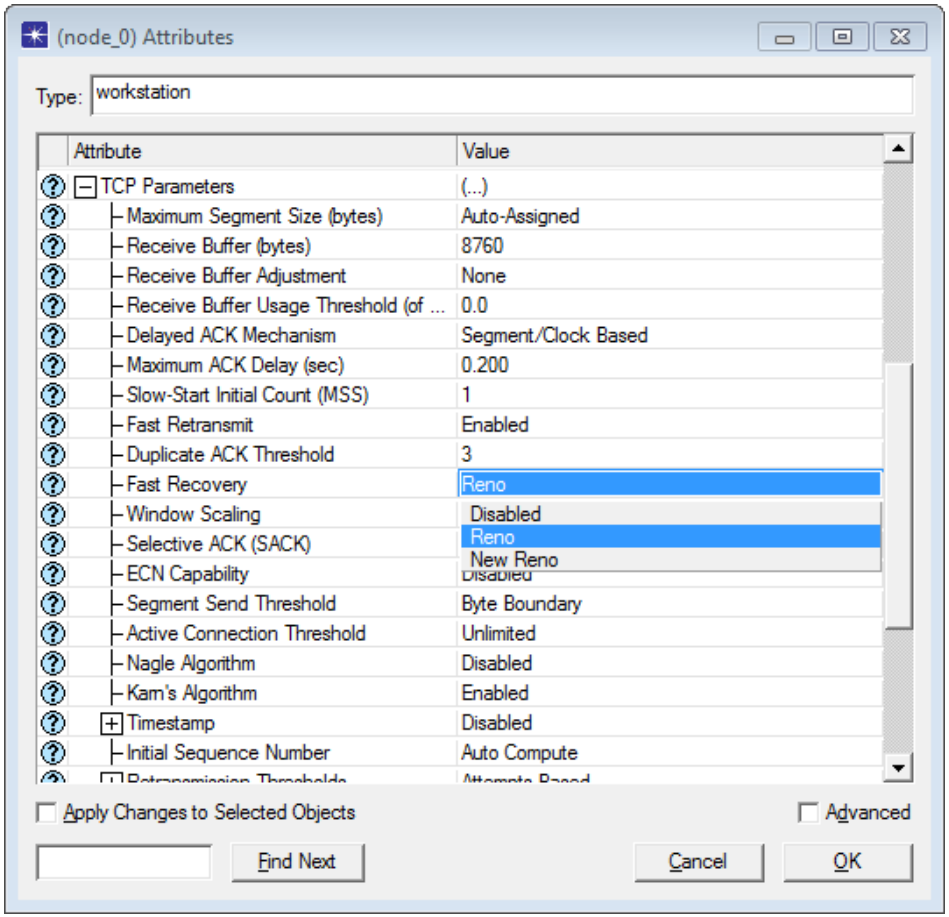
- TCP Tahoe с Fast Retransmit



- **Duplicate ACK Threshold**
  - Число повторных подтверждений, необходимых для запуска алгоритма быстрой повторной передачи

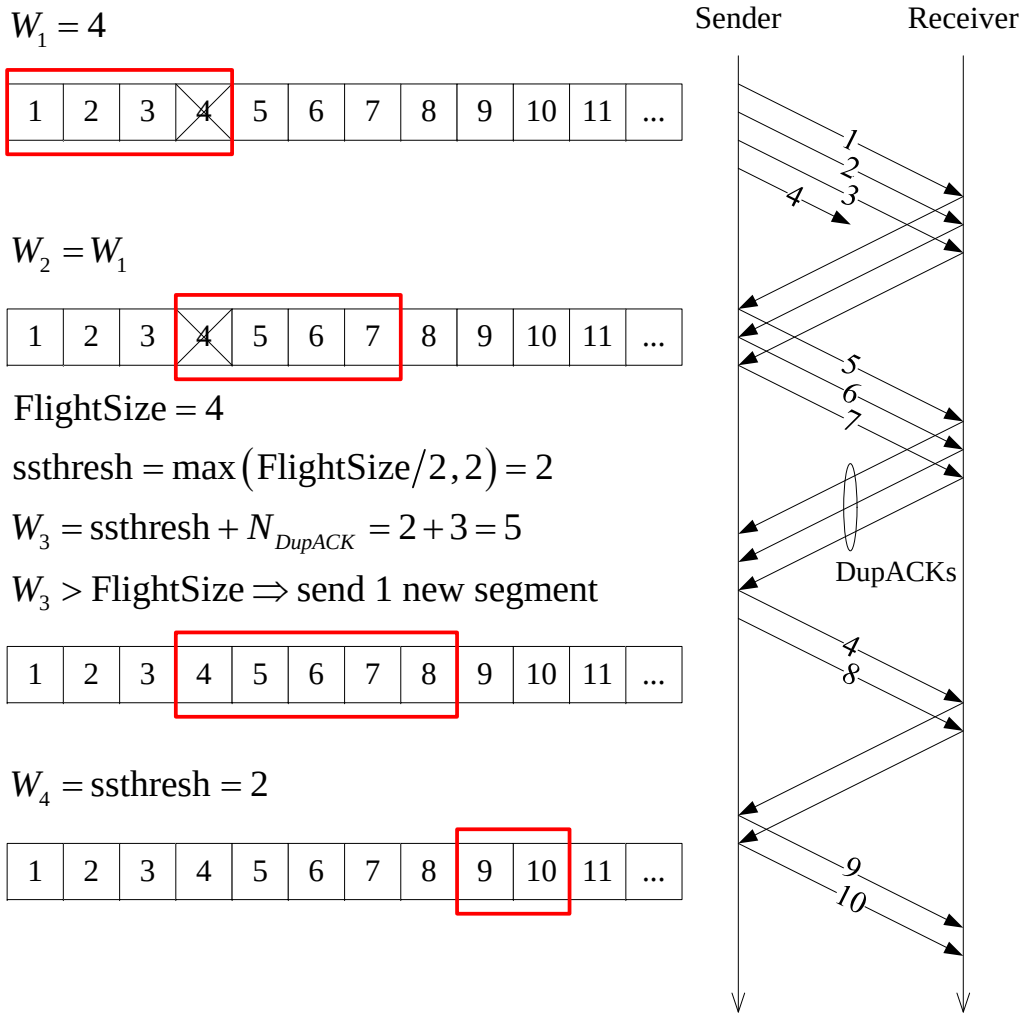


- **Fast Recovery**
  - Использование и модификация (Reno/NewReno) алгоритма быстрого восстановления



TCP в OPNET

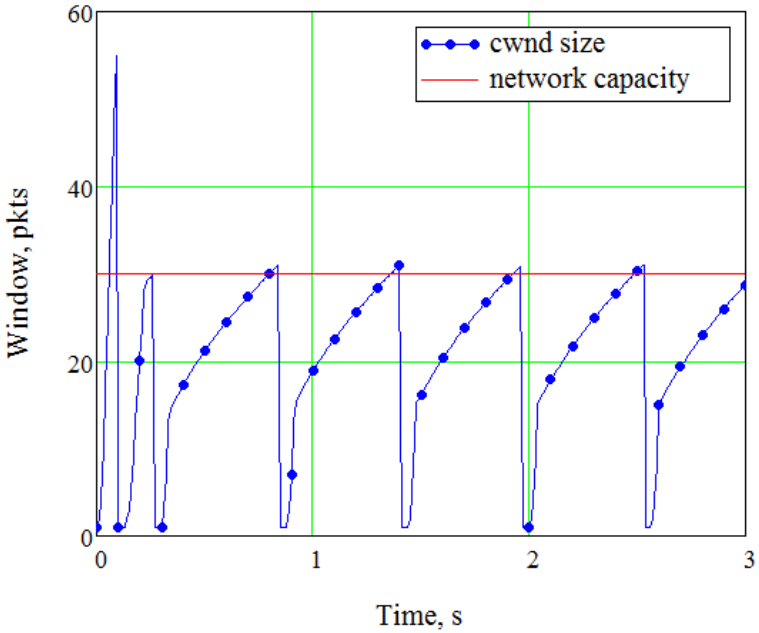
- Совместное использование алгоритмов быстрой повторной передачи и быстрого восстановления (Fast Recovery)



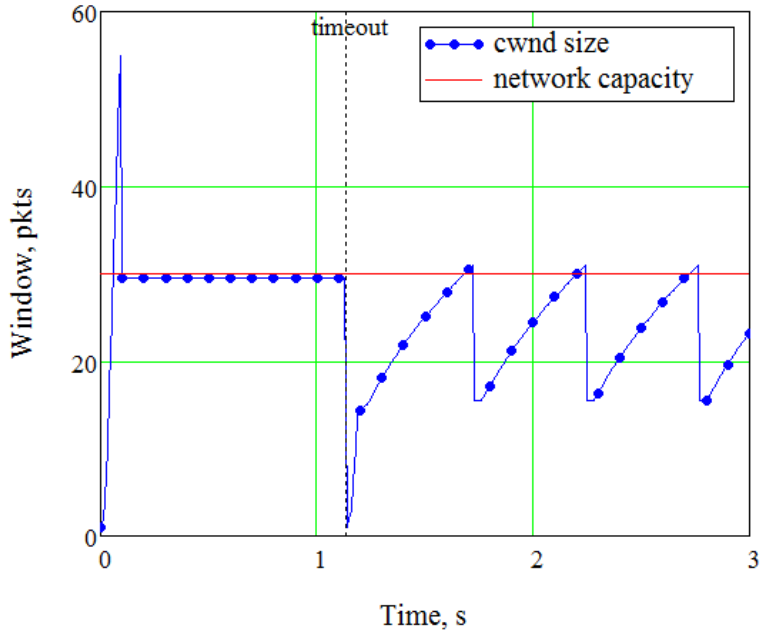
TCP в OPNET

- Сравнение стандартных версий TCP

- Tahoe без Fast Recovery



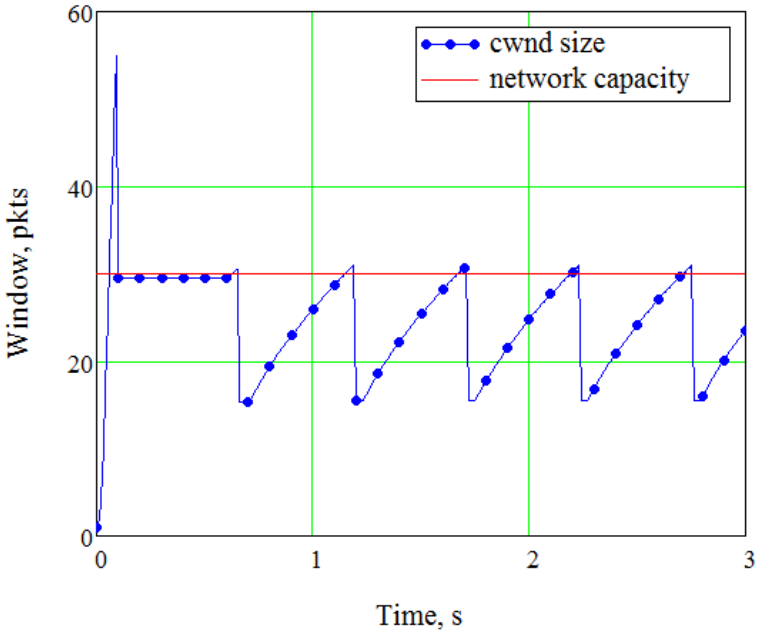
- Reno с Fast Recovery



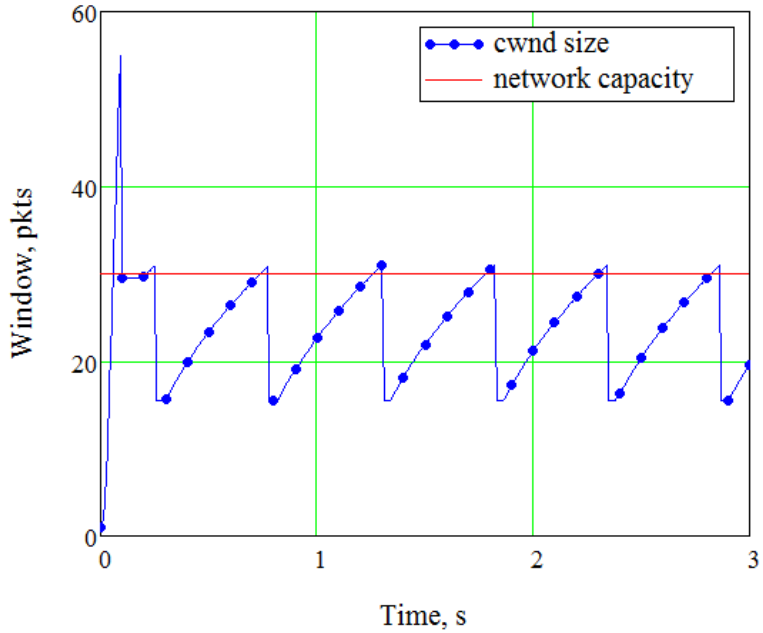
TCP в OPNET

- Сравнение стандартных версий TCP

- NewReno с Fast Recovery



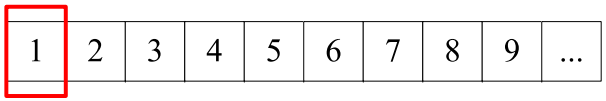
- SACK с Fast Recovery



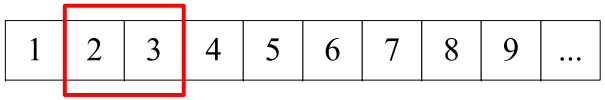
TCP в OPNET

- Использование алгоритма медленного запуска (Slow Start)
  - cwnd растет **экспоненциально**, увеличиваясь в 1,5/2 раза каждый раунд

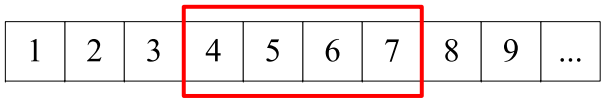
$W_1 = \mathrm{IW} = 1$



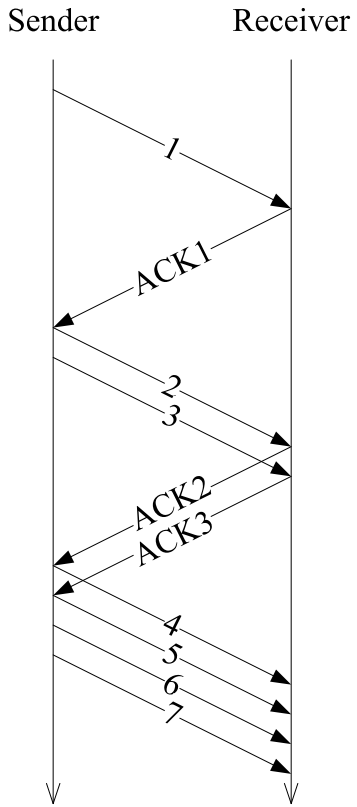
$W_2 = W_1 + 1 = 2$



$W_3 = W_2 + 2 = 4$

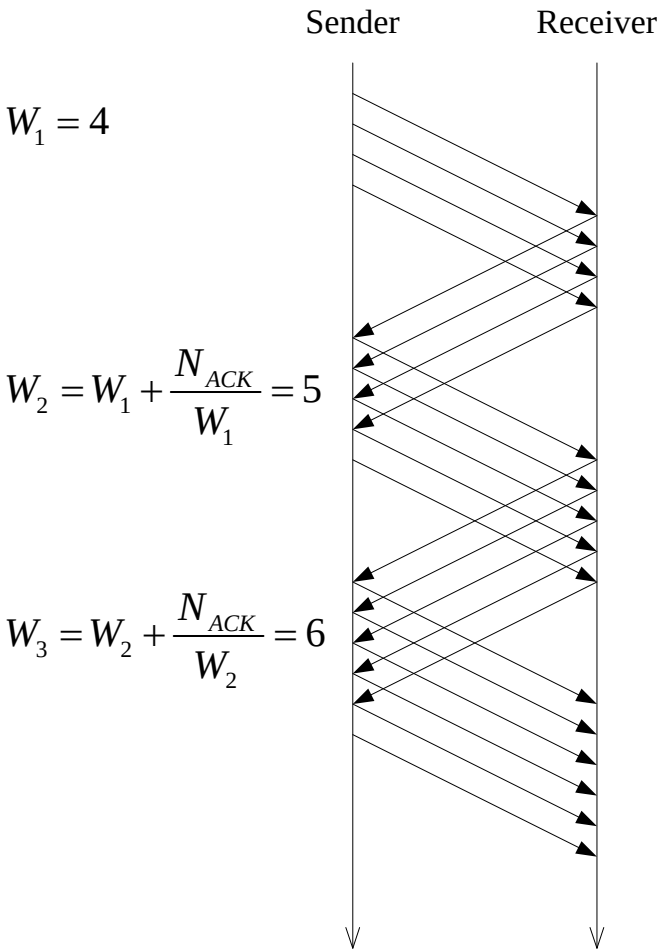


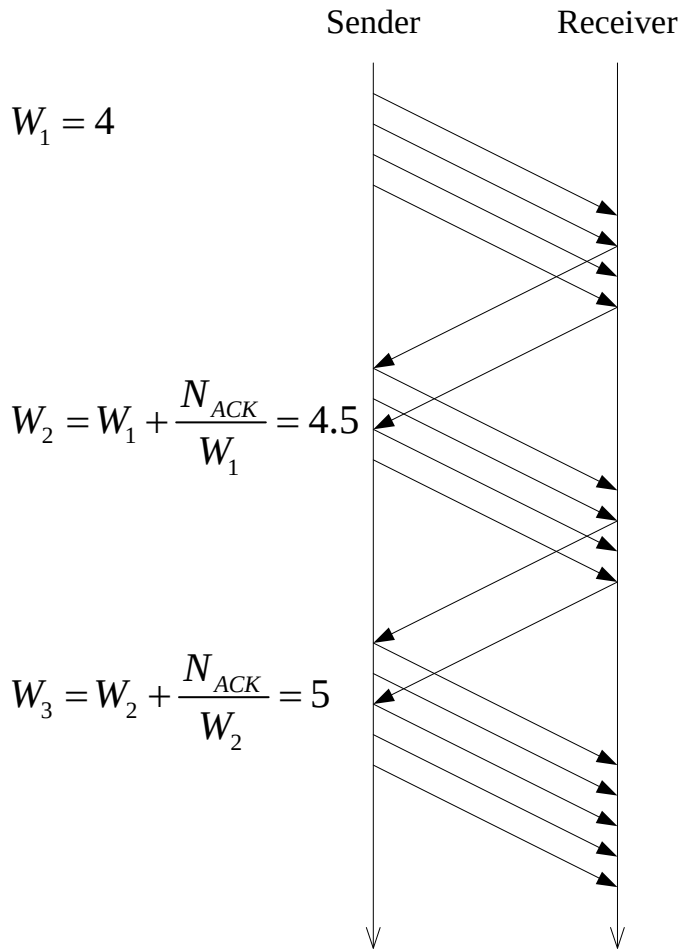
$W_{i+1} = W_i + W_i = 2W_i$



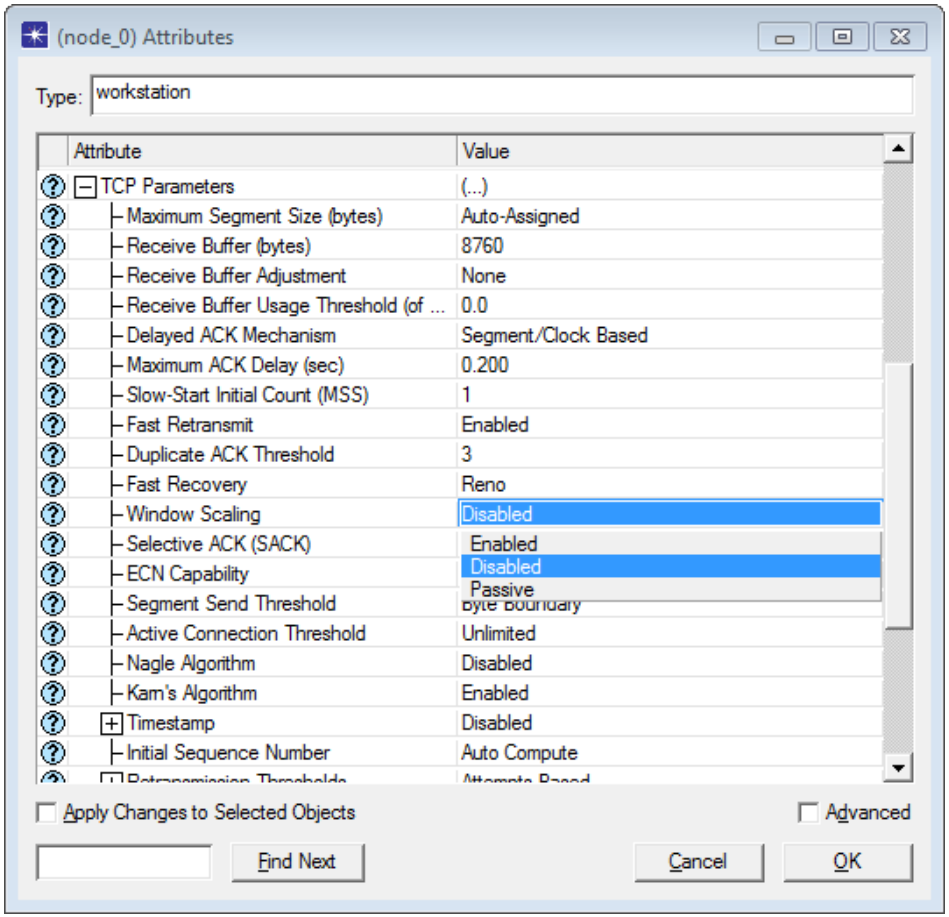
TCP в OPNET

- Использование алгоритма предотвращения перегрузки (Congestion Avoidance)
  - cwnd растет **линейно**, увеличиваясь на 0.5/1 MSS каждый раунд

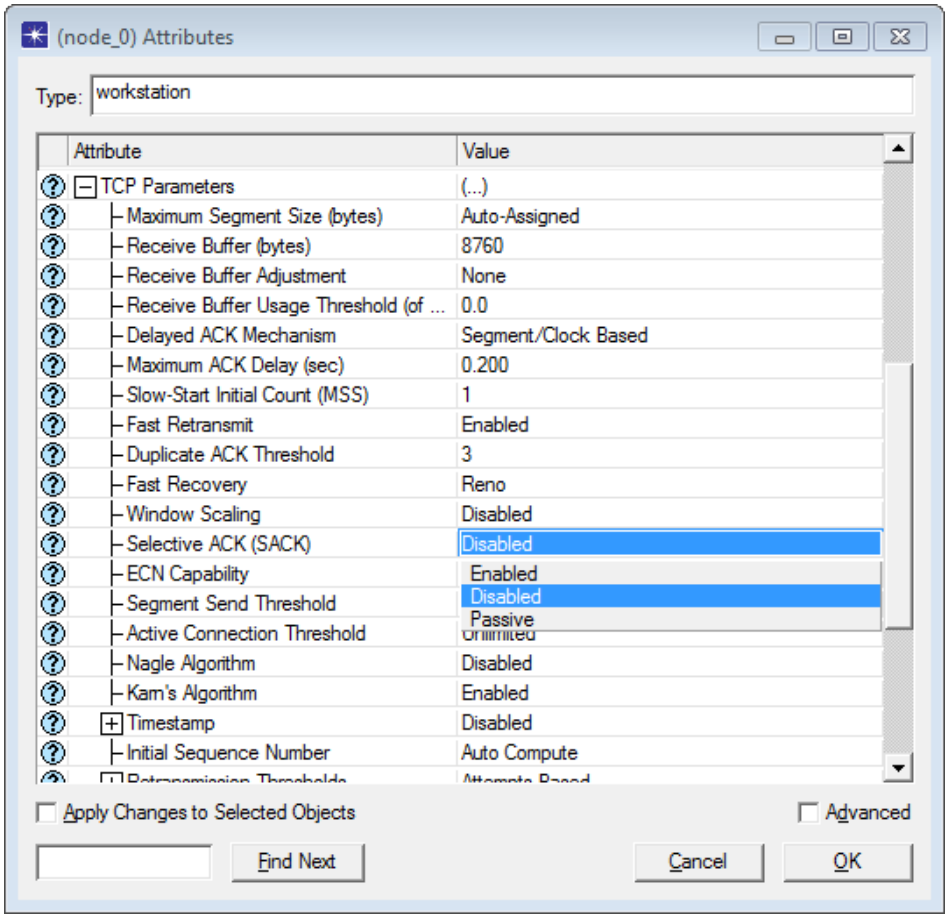




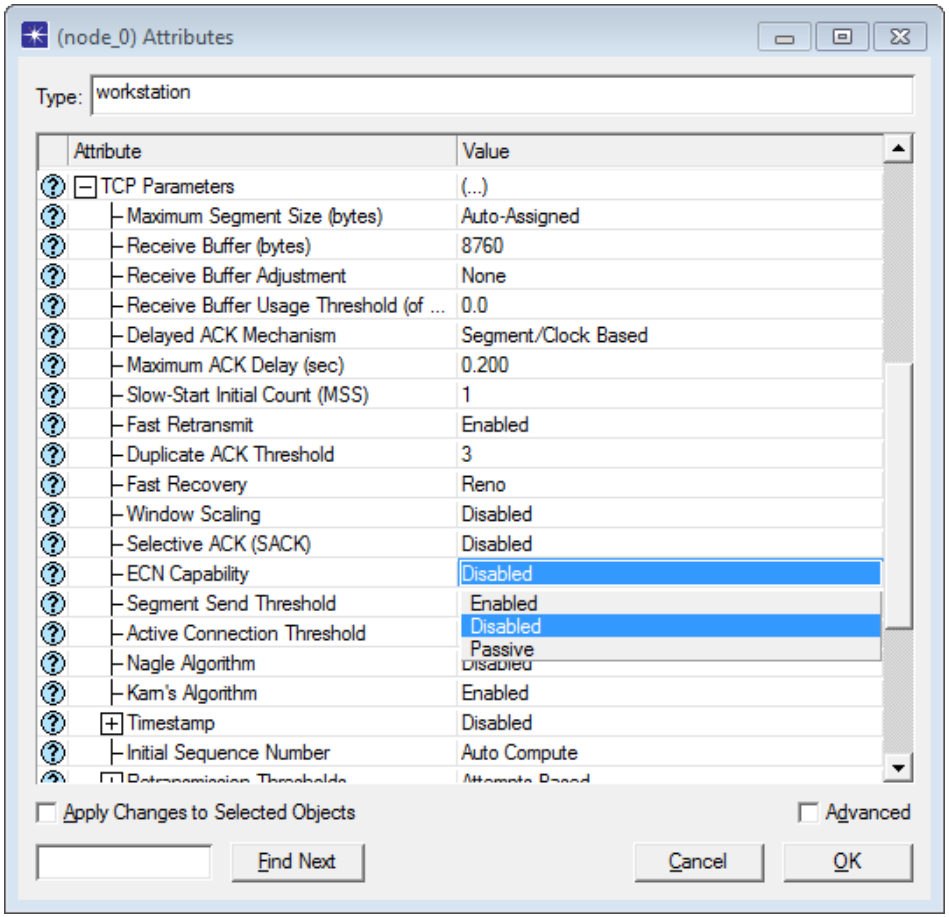
- **Window Scaling**
  - rwnd без масштабирования максимум: $2^{16} - 1 = 65535$ байт
  - rwnd с масштабированием максимум: 65535×2^s, $S = 0, \ldots, 14$



- **Selective ACK (SACK)**
  - Без SACK: только кумулятивные подтверждения
  - С SACK: кумулятивные и выборочные подтверждения

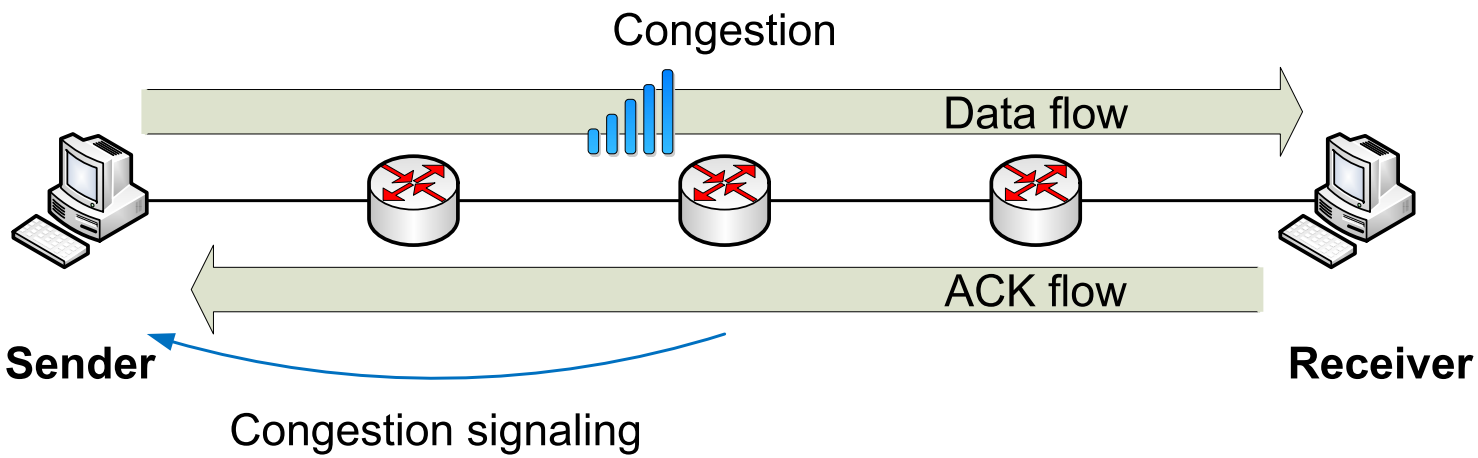


- **ECN Capability**
  - Без ECN: снижение скорости по факту перегрузки и потерь
  - С ECN: снижение скорости по указанию маршрутизатора

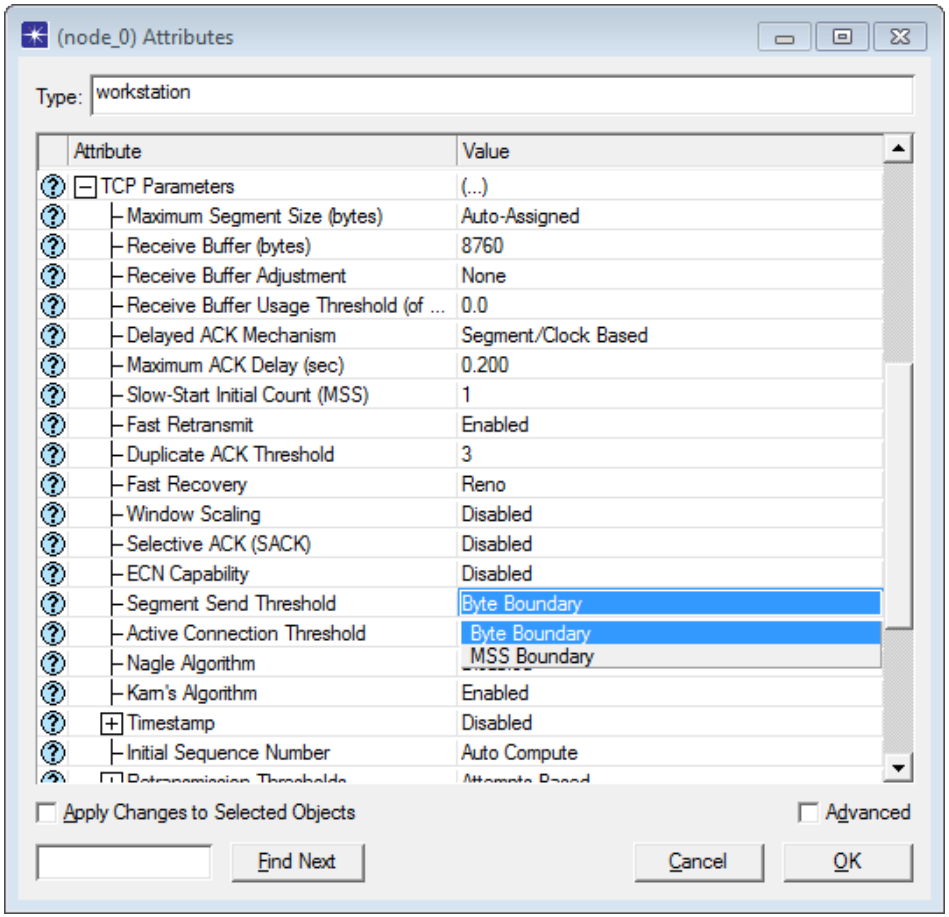


TCP в OPNET

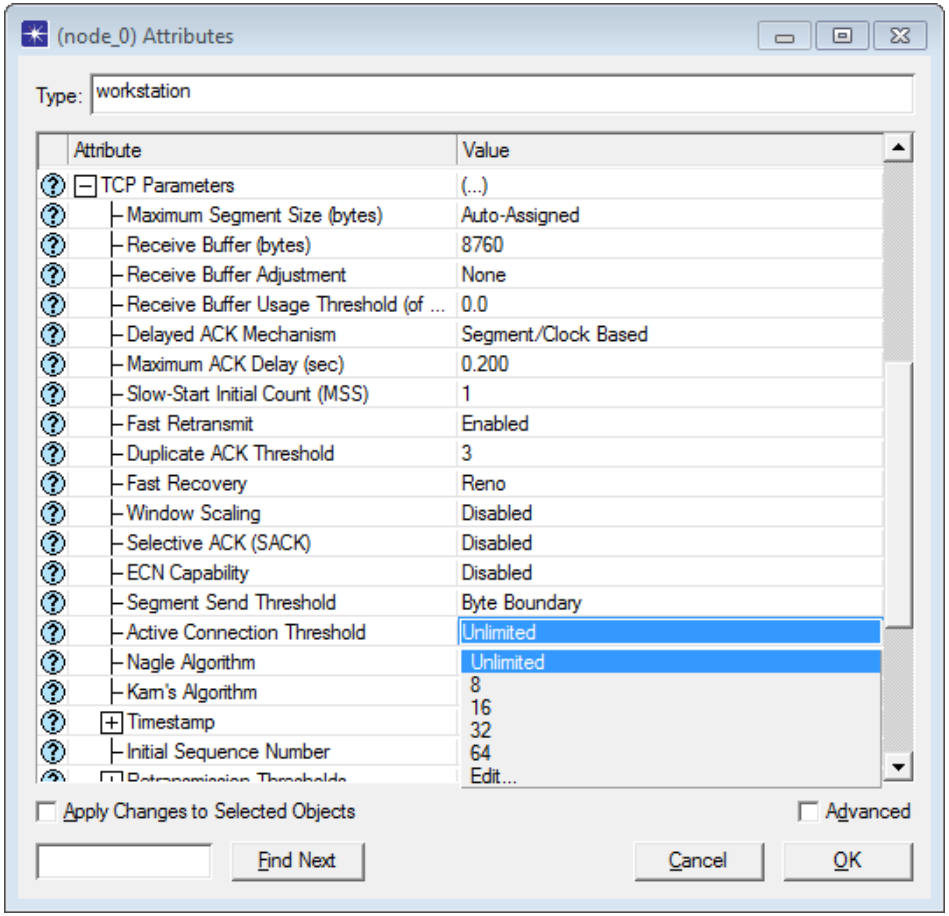
- Явное уведомление о перегрузке (Explicit Congestion Notification, ECN) требует одновременной поддержки на стороне:
  - Передатчика
  - Приемника
  - Маршрутизатора узкого участка



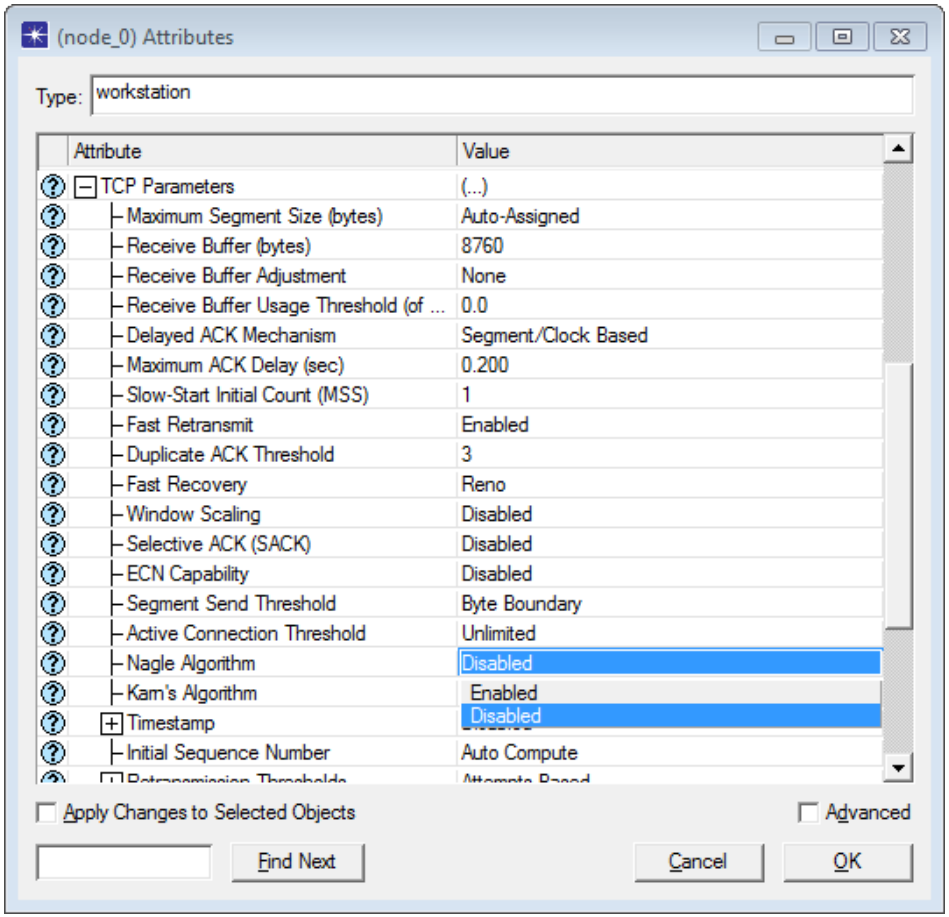
- **Segment Sent Threshold**
  - Byte Boundary: при потере ssthresh $= \frac{cwnd}{2}$
  - MSS Boundary: при потере ssthresh $= \left\lceil \frac{\text{cwnd}}{2} \right\rceil \times \text{MSS}$



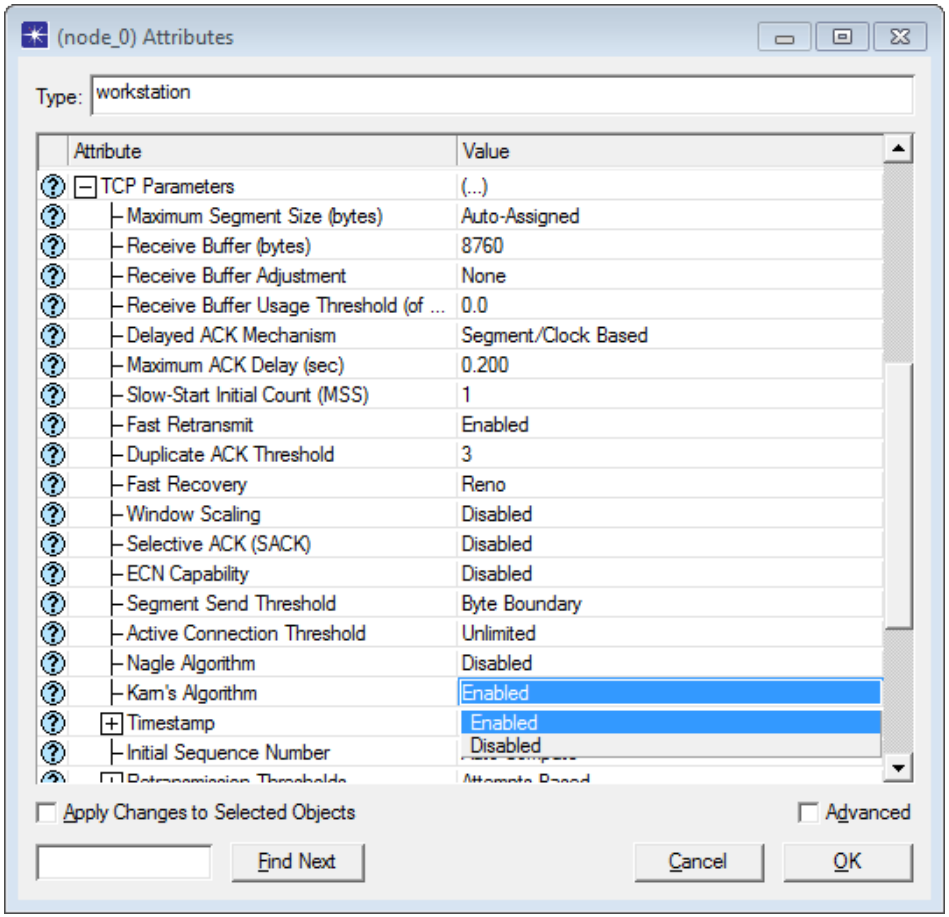
- **Active Connection Threshold**
  - Максимальное число активных TCP-соединений для узла



- **Nagle Algorithm**
  - Enabled: не отправлять сегмент меньше MSS, пока не придет ACK
  - Disabled: не задерживать отправку сегментов при любом объеме

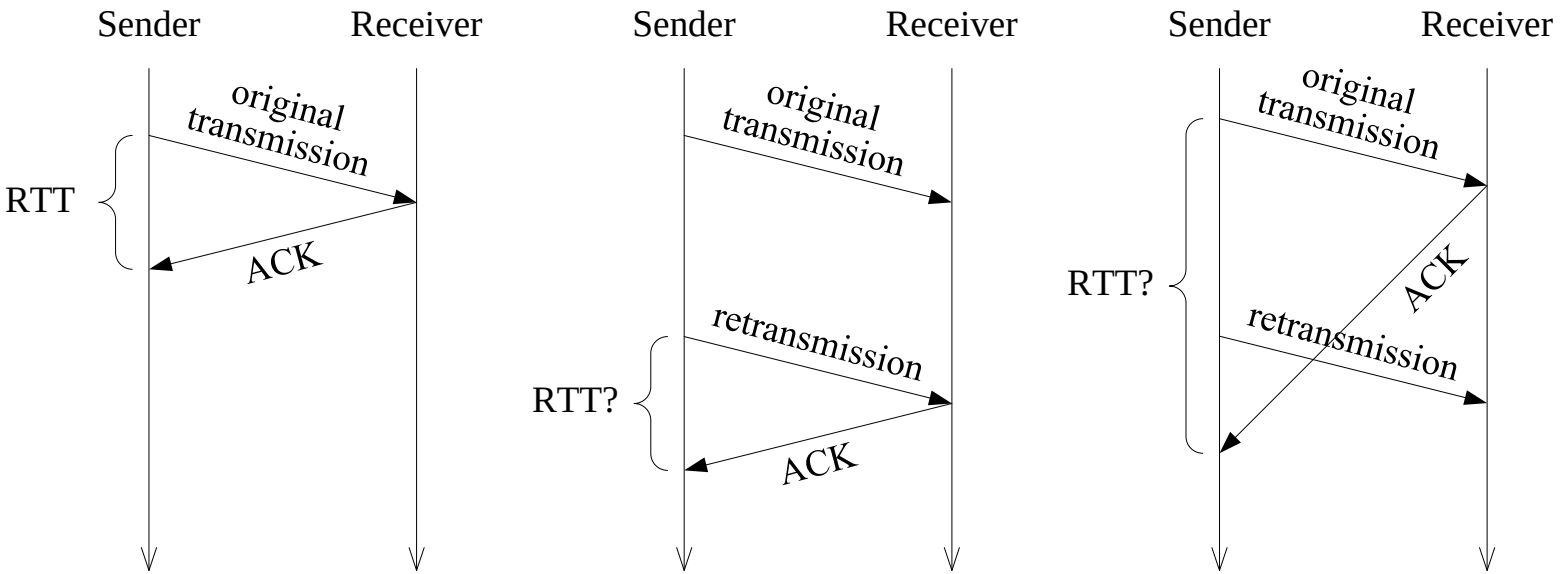


- **Karn's Algorithm**
  - Игнорировать замеры RTT при повторных передачах
  - Вместо этого удваивать прежнее значение таймаута



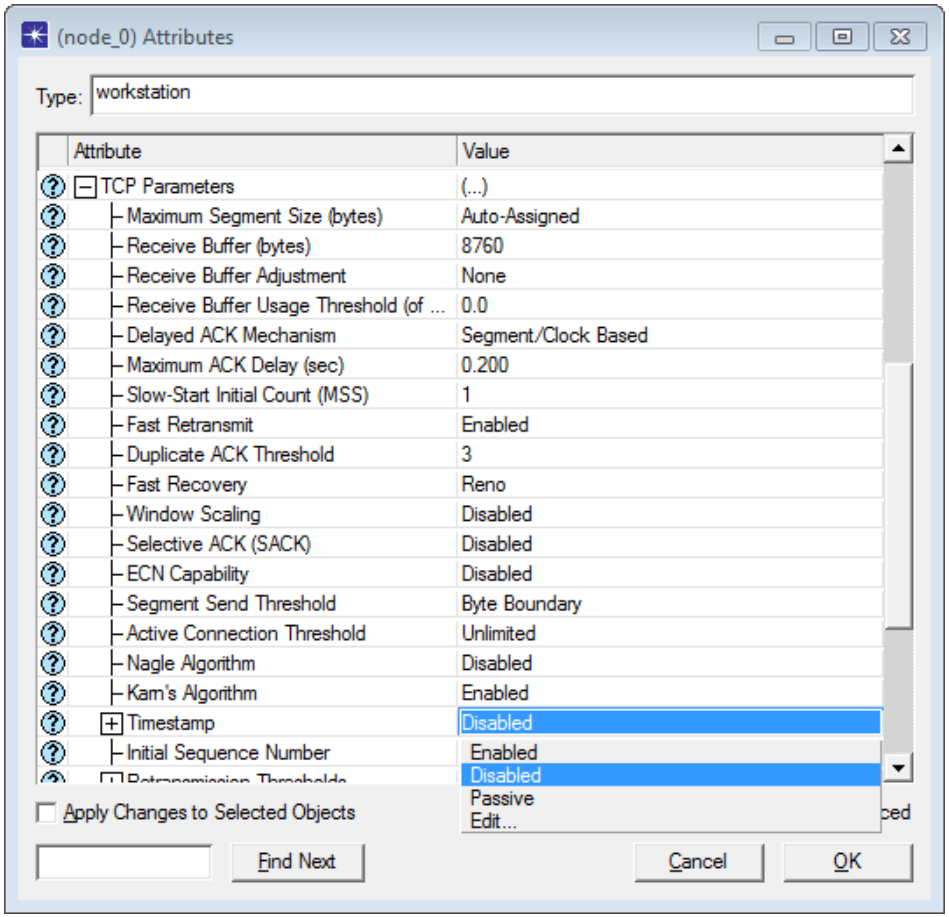
TCP в OPNET

- Как корректно определить время обращения сегмента (Round-Trip Time, RTT) при повторной передаче для расчета таймаута повторной передачи (Retransmission Time-Out, RTO)?
  - Вопрос: к какой передаче относится полученный ACK-сегмент?
  - Ответ: не считать RTT, а просто удвоить прежнее значение RTO!
  - RTO, $2 \times$ RTO, $4 \times$ RTO, $8 \times$ RTO, $16 \times$ RTO, $32 \times$ RTO, $64 \times$ RTO, $64 \times$ RTO, $64 \times$ RTO, ...

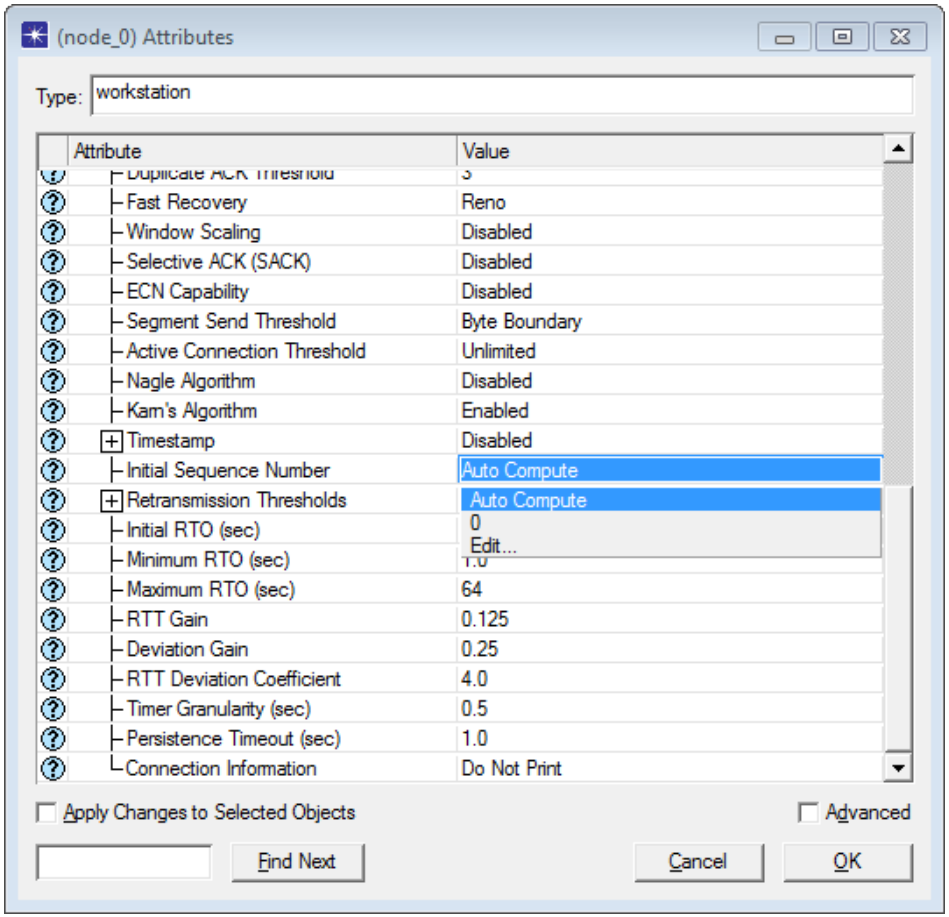


TCP в OPNET

- **Timestamp**
  - Enabled/Passive: использовать временные метки
  - Disabled: не вставлять временные метки в сегменты

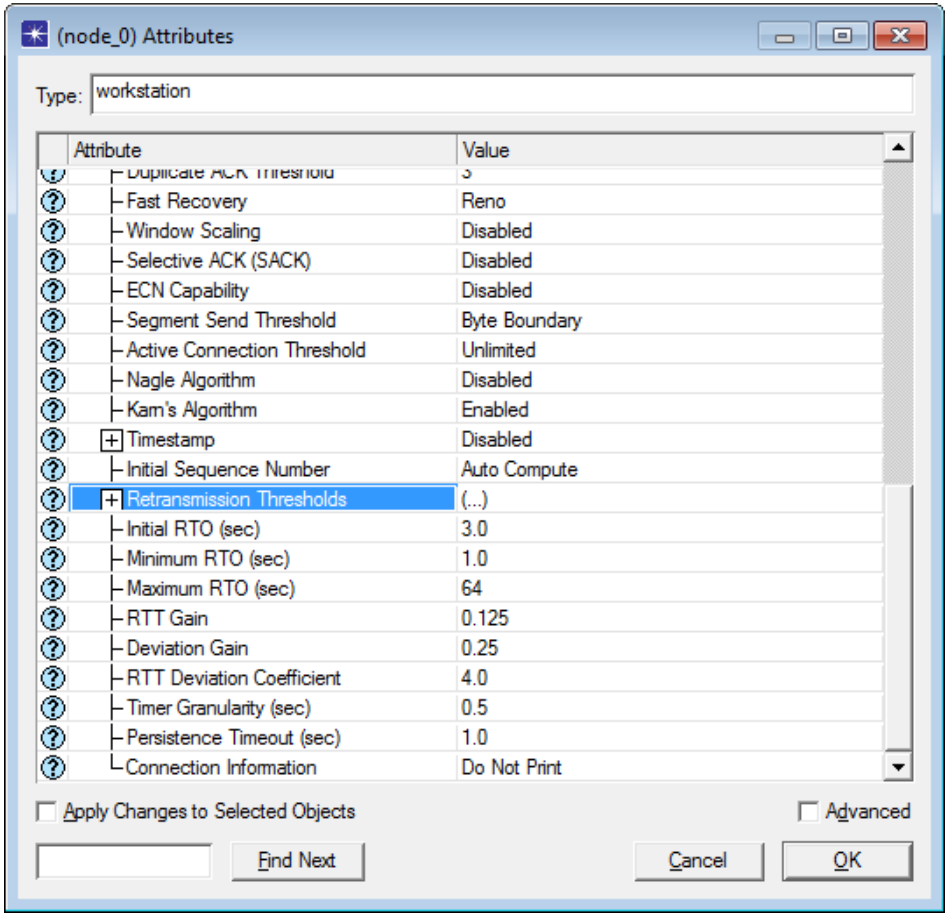


- **Initial Sequence Number**
  - Auto Compute: отсчет с произвольного числа от 0 до $2^{32} - 1$
  - 0: начинать отсчет передаваемых байтов всегда с 0



TCP в OPNET

- **RTO/RTT**
  - RTO = SRTT + max{Timer Granularity, 4 × RTTVAR}
  - Exponentially Weighted Moving Average (EWMA)



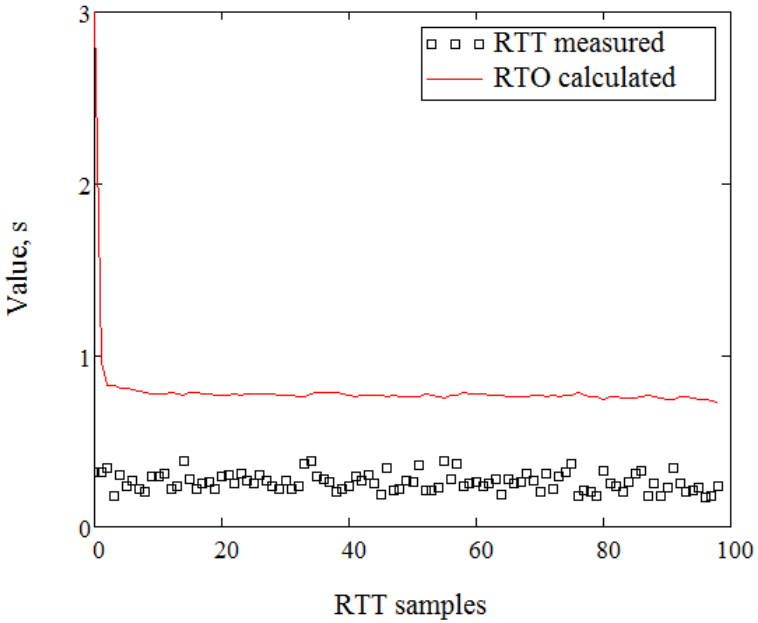
TCP в OPNET

- **Экспоненциально взвешенное скользящее среднее**:
- Round-Trip Time Variation (RTTVAR)
  - **Deviation Gain = 0,25**
- Smoothed Round-Trip Time (SRTT)
  - **RTT Gain = 0,125**
- Retransmission Time-Out (RTO)
  - **RTT Deviation Coefficient = 4,0**
- Первый замер RTT (R):
  - SRTT $\leftarrow$ R
  - RTTVAR $\leftarrow$ R/2
  - RTO $\leftarrow$ SRTT + max{Timer Granularity, **4,0**$\times$RTTVAR}
- Последующие замеры RTT (R'):
  - RTTVAR' $\leftarrow$ (1 – **0,25**)$\times$RTTVAR + **0,25**$\times$|SRTT – R'|
  - SRTT' $\leftarrow$ (1 – **0,125**)$\times$SRTT + **0,125**$\times$R'
  - RTO' $\leftarrow$ SRTT' + max{Timer Granularity, **4,0**$\times$RTTVAR'}

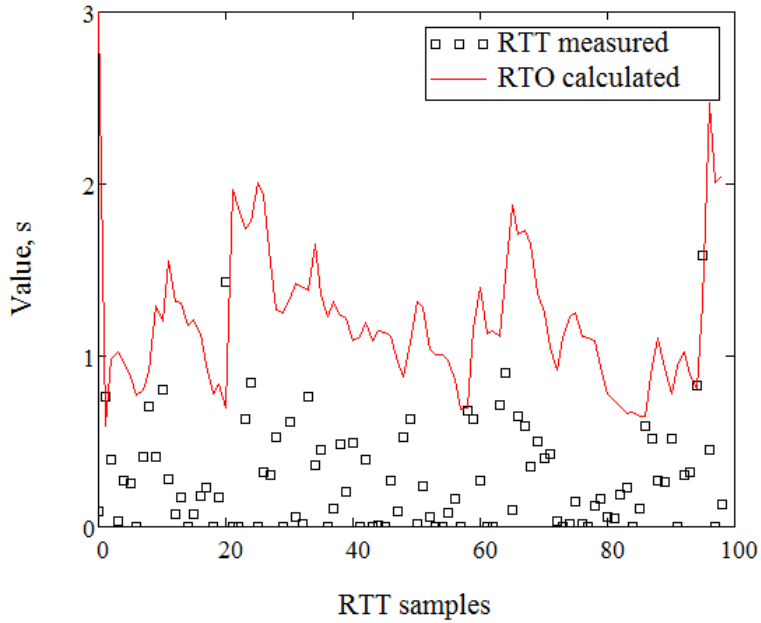
TCP в OPNET

- Без ограничения снизу на значение RTO:

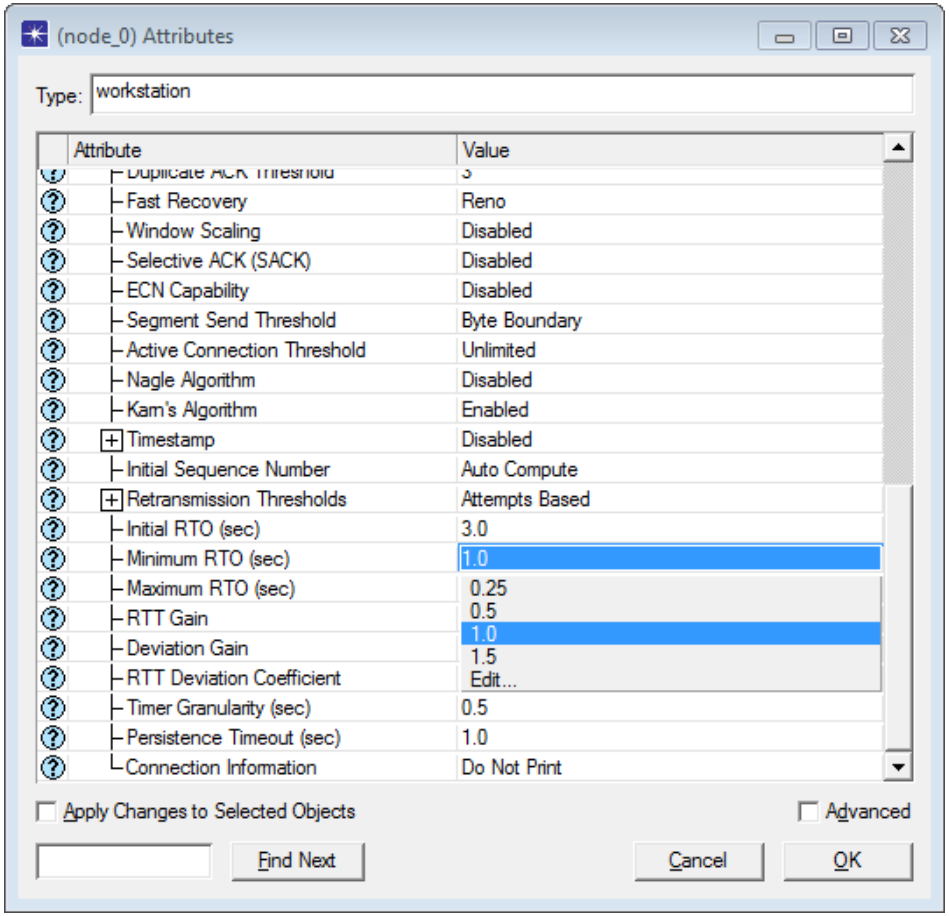
- Малый разброс RTT



- Большой разброс RTT



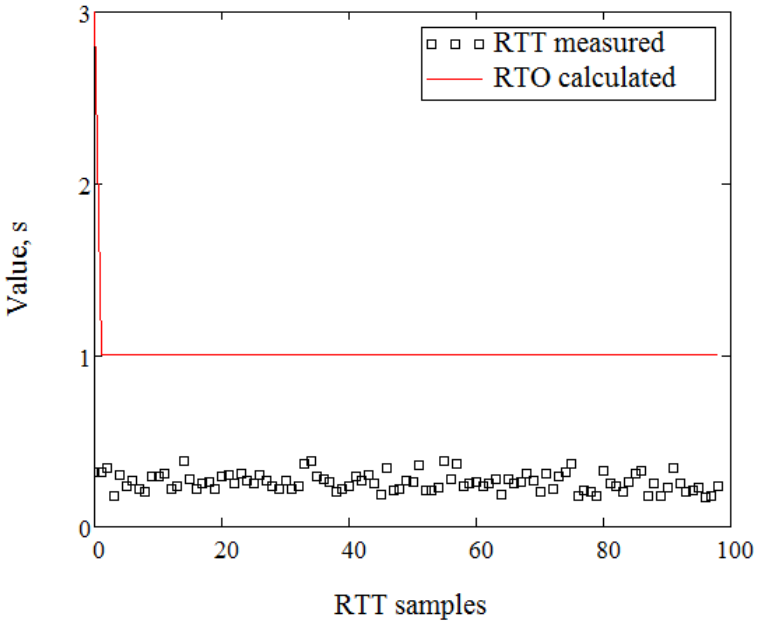
- **Minimum RTO**
  - Минимальное значение RTO



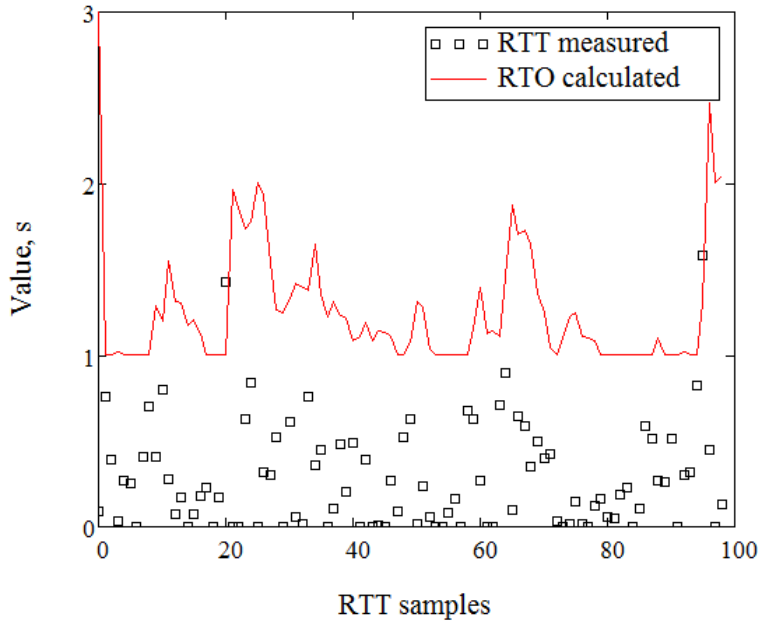
TCP в OPNET

- С ограничением снизу на значение RTO:

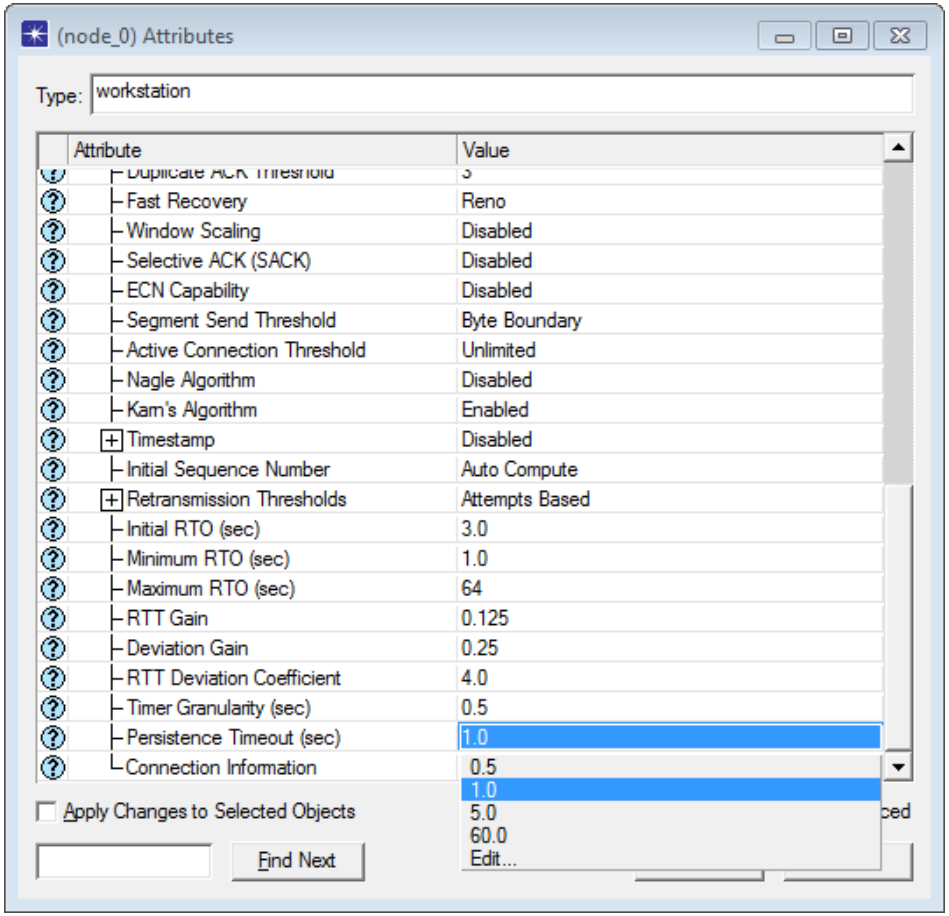
- Малый разброс RTT



- Большой разброс RTT

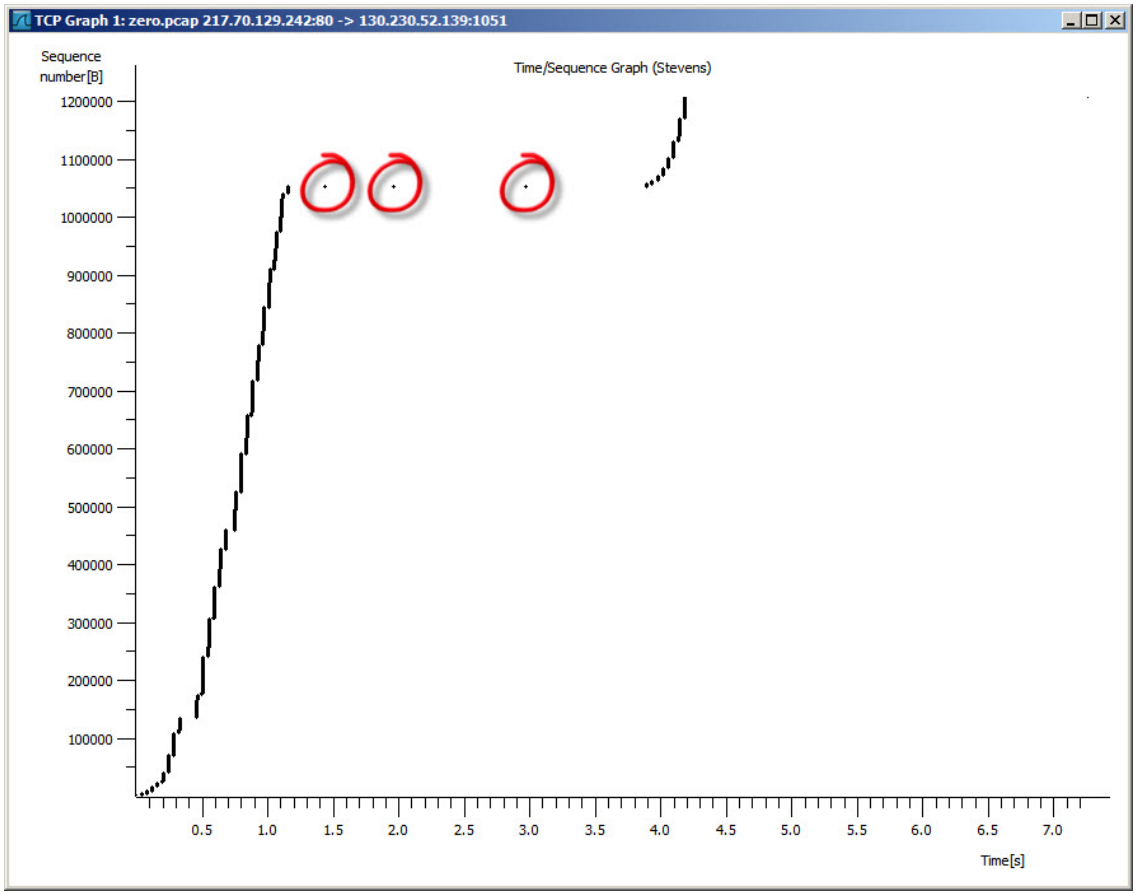


- **Persistence Timeout**
  - Исходное значение таймаута для посылки пробных сегментов при объявлении приемником нулевого окна(?)



TCP в OPNET

- Посылка пробных сегментов, чтобы гарантированно получить новое (ненулевое) значение окна приемника
  - Приемник отвечает на пробные сегменты, указывая текущий объем свободного места в своем буфере



Содержание

Сравнение параметров

- *'Microsoft Windows 2000: TCP/IP Implementation Details,' 2000*
  - Доверяй, но проверяй! ☺

Параметр	Windows 2000			
	OPNET IT Guru Academic Edition	Microsoft White Paper	Windows 2000 Pro SP4 2195	Совпало?
Maximum Segment Size	Auto-Assigned	Auto-Assigned	Auto-Assigned	
Receive Buffer	16384 байт	16384 байт	64240 байт	нет
Delayed ACK Mechanism	Segment/Clock Based	Segment/Clock Based	Segment/Clock Based	
Maximum ACK Delay	200 мс	200 мс		
Slow Start Initial Count	2xMSS	2xMSS	1xMSS	нет
Duplicate ACK Threshold	3	2		нет
Window Scaling	Disabled	do not initiate options but if requested provide them	Disabled	нет
Selective ACK (SACK)	Enabled	Enabled	Enabled	
Nagle Algorithm	Enabled	Enabled		
Initial RTO	3 c	3 c	3 c	